TF: ALGORITMO GENÉTICO APLICADO NO PROBLEMA DO CACHEIRO VIAJANTE.

Carlos Felipe Cacione Alves

INPE, COMPUTAÇÃO APLICADA – CAP-354, 1P_2018
São José dos Campos, SP
carlos.fas@gmail.com

RESUMO

Este trabalho apresenta os resultados na aplicação do Algoritmo Genético para solução do problema do cacheiro viajante. Os resultados indicam a convergência entre as soluções em torno da geração 200 com o erro associado menor que grande parte das distâncias individuais existentes no grafo mostrando assim robustez quanto a convergência para uma solução próxima da ótima global e eficiente varredura no espaço de soluções.

Key words: Expert Systems, Optimization, , Genetic Algorithm.

I. INTRODUÇÃO

O problema do cacheiro viajante (PCV) é um dos problemas mais conhecidos em otimização combinatória. O problema foi definido em 1800 por W. R. Hamilton e T. Kirkman e pode ser posto como: Dada uma lista de cidades e a distância entre elas, qual a menor trajetória que passa por todas as cidades e volta para a cidade de partida?

Matematicamente, esse problema pode ser facilmente convertido como uma busca sobre qual caminho fechado existente em um grafo completo não direcionado e com pesos? Utilizando o grafo como um modelo para o problema as cidades representariam nós, as arestas os diversos caminhos entre as cidades, tal que todos os nós tenhamos conexão com todos e cada ligação seja ponderado pela distância geográfica entre as cidades.

Dado um conjunto finito V ={v1,v2,...,vn} e uma relação E como um subconjunto de pares ordenados tal que seus elementos tenham simetria e reflexividade, um grafo G pode ser definido como um par G=(V,E) onde V eh um conjunto de pontos ou vértices num plano e E uma relação em V, conhecida também como arestas ou ligações.

Um grafo é ponderado é uma quádrupla G=(V,E,W,f) onde W é um conjunto de pesos tal que f é o mapeamento de E em W. Em um grafo não direcionado e não ponderado a distância do menor caminho entre dois pois p e q, d(p,q)=dpq é o caminho contendo o menor número de arestas, definindo assim um espaço métrico em V. A solução então do problema PCV pode ser entendido como uma busca de uma trajetória fechada contendo todos os vértices de tal forma que a soma d(p,q) para que todos os vértices distintos seja mínima. O tamanho do espaço amostral para esse problema é de (n-1)!/2 sendo n o número de nós, além disso devido a complexidade dos algoritmos determinísticos para este tipo de problema, comumente são utilizados meta heurísticas para solução, tal como o Algoritmo genético que foi aplicado nesse trabalho para obtenção da solução.

II. ALGORITMO GENÉTICO

Os Algoritmos Genéticos (AGs) são o ramo mais conhecido da Computação Evolutiva (CE). A CE é um ramo da ciência da computação que propõe um sistema de processamento de dados baseado em mecanismos encontrados na natureza baseado na teoria evolutiva de Darwin. No AG os indivíduos contêm um genótipo formado por cromossomos representados por uma cadeia de bits, de tal forma que através da aplicação sucessiva dos operadores mencionados um computador possa simular o processo de evolução natural de Darwin. Tal processo é simulado através do pseudocódigo da figura 2.

0º - Inicializa População de Indivíduos
Enquanto não terminar faça
1º - Avalie a População
2º - Selecione Pais
3º - Recombinação e mutação
4º - Avalia População
5º - Selecione sobreviventes (nova população)
Fim enquanto

Figura 1 – Pseudocódigo do AG.

Cada indivíduo deve ser representando através do seu genótipo para isso é necessário modelar o problema para que as soluções sejam representadas através de uma cadeia de bits que compões o cromossomo. Feito tal modelo a população de soluções inicial é gerada aleatoriamente e avaliada através da função de avaliação, essa função tem o

objetivo de medir a aptidão de cada um dos indivíduos, ou seja, quantificar o quão melhor é uma solução em relação as demais. Em seguida é gerada uma nova população de solução através da recombinação que simula a troca de material genético entre os pais e o operador de mutação que realiza mudanças aleatórias no material genético. Finalmente é feita a seleção de quais indivíduos serão utilizados para a geração da próxima solução. O critério de parada pode ser tanto através da qualidade dos indivíduos, ou seja, depois da obtenção de um indivíduo com uma pontuação de adaptação desejada, tal pontuação dada pela função avaliação ou uma quantidade predefinida de recombinações.

IV. ENGENHARIA DO CONHECIMENTO E VISUALIZAÇÃO

A implementação do algoritmo genético foi feita na linguagem *Python* e utilizado o pacote *networkX* para a visualização através de grafos. Como definido pelo PCV foi inicialmente criado um grafo completo não direcionado tal que cada aresta foi mapeada com o peso. Foram definidas 25 cidades com coordenadas espaciais aleatórias no plano cartesiano contidas em uma região quadrada de 0 a 200. Os pesos de cada aresta são obtidos pela distância euclidiana entre as cidades, coordenadas x,y de cada nó. Linhas xxx até XXX do código em anexo. Resultando no grafo da figura 2.

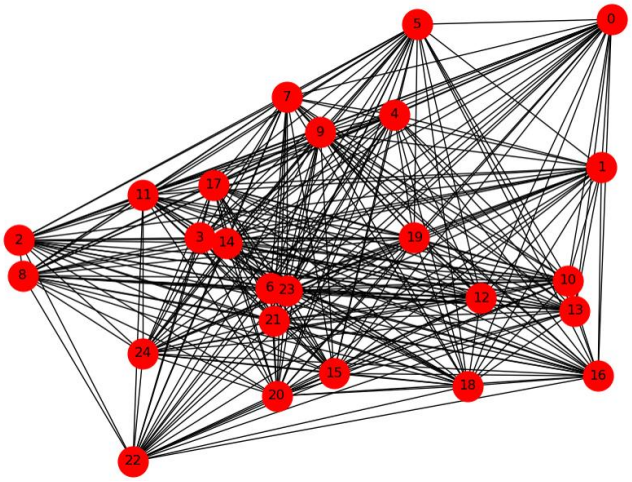

Figura 2 – Rede completa ponderada e não direcionada representando 25 cidades com coordenadas aleatórias

A engenharia do conhecimento pelo Algoritmo Genético foi concebida da seguinte forma. Cada cromossomo ou indivíduo representa uma possível solução do problema, no caso um dos (25-1)!/2= 310.224.200.866.619.719.680.000 caminhos fechados no espaço amostral. Cada gene então é um nó do caminho, tal que a sequencia dos genes define o tamanho do caminho fechado. A população é a quantidade de indivíduos ou cromossomos existentes a cada interação, geração e as melhores soluções ou indivíduos com o melhor genótipo são aqueles com maior valor do inverso da distância euclidiana entre as cidades sequenciais de cada indivíduo. A figura 3 mostra a representação da variável referente a *iésima* população gerada, é possível ver que cada um dos 100 cromossomos, representando os possíveis caminhos fechados é composto por 25 genes, ou seja, 25 combinações de cidade.


iPopulation - Lista (100 elementos)

Índice	Tipo	Tamanho	Valor
0	list	25	[City, City, City, City, City, City, City, City, City, City, ...]
1	list	25	[City, City, City, City, City, City, City, City, City, City, ...]
2	list	25	[City, City, City, City, City, City, City, City, City, City, ...]
3	list	25	[City, City, City, City, City, City, City, City, City, City, ...]
4	list	25	[City, City, City, City, City, City, City, City, City, City, ...]
5	list	25	[City, City, City, City, City, City, City, City, City, City, ...]
6	list	25	[City, City, City, City, City, City, City, City, City, City, ...]
7	list	25	[City, City, City, City, City, City, City, City, City, City, ...]
8	list	25	[City, City, City, City, City, City, City, City, City, City, ...]
9	list	25	[City, City, City, City, City, City, City, City, City, City, ...]
10	list	25	[City, City, City, City, City, City, City, City, City, City, ...]

OK Cancel

Figura 3 – *iésima* população com 100 indivíduos, formados por 25 genes.

Seguindo o pseudocódigo da figura 1, a população inicial é criada inicialmente através das funções `initialPopulationFromCityList(popSize,cityList)` e `createChromosome(cityList)` estas simplesmente criam uma população de tamanho `popSize` com indivíduos compostos por cada um dos 25 vértices em ordem aleatória.

A etapa de avaliação da população é feita pelas funções e `fit(self)` da classe `Fitness`. A função `calcAdaptation(population)` retorna uma lista de índices de indivíduos da população por ordem de mais adaptado ou melhor genótipo e o valor calculado pela função de avaliação, no caso o inverso do tamanho do ciclo. A função `fit(self)` é responsável chama a função `routeDistance(self)`, que por sua vez retorna a soma das distâncias entre os genes ordenados no indivíduo calculada explicitamente através da distância euclidiana das coordenadas das cidades.

Em seguida, os pais são selecionados utilizando as funções `matingpool(currentGen,selectionResults)` e `selection(popRanked,eliteSize)`. A função `selection(popRanked,eliteSize)` seleciona de duas formas, primeiro seleciona os *n* melhores utilizando a métrica definida pela função `fit(self)`, este método é denominado de elitismo. Em seguida utiliza o método denominado, método da Roleta (figura 4), que tem por finalidade selecionar os indivíduos por sorteio com base na sua aptidão. A função calcula uma probabilidade associada a razão da aptidão de cada indivíduo pela soma das aptidões, em seguida sorteia *i* vezes (i = *tamanho da população – eliteSize*) um valor de aptidão entre 0 e 100 e adiciona qualquer indivíduo com valor de aptidão igual ou maior. Por fim, a função `matingpool(currentGen, selectionResults)` apenas recupera os indivíduos de `selection`, pois esta opera sobre uma lista contendo o indicie do indivíduo e sua aptidão e não a população em si.

Indivíduo (S_i)	Aptidão $f(S_i)$	Aptidão Relativa
S_1 10110	2.23	0.14
S_1 11000	7.27	0.47
S_1 11110	1.05	0.07
S_1 01001	3.35	0.21
S_5 00110	1.69	0.11

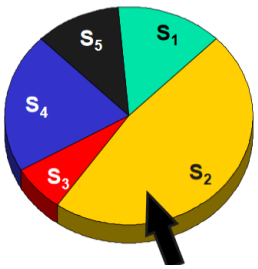

Figura 4 – Representação da metodologia de seleção por Roleta (http://wiki.icmc.usp.br/images/f/fa/Aula_2_CB_AEs.pdf)

O processo de recombinação é composto pelas funções `crossoverPopulation(matingpool, eliteSize)` e `odered_crossover(parent1,parent2)`.A primeira apenas mantem os indivíduos que não sofrerão recombinação devido a elitismo, os demais sofrerão um processo chamado ordered crossover (figura 5). Em muitas implementações é definido um ou mais pontos para recombinação, mas como aqui os indivíduos devem ter genes únicos, pois um caminho não pode ter mais de um vértice igual, o descendente é montado com um trecho de genes de um dos pais e demais de outro parente, tais que esses genes não estejam na porção anterior.

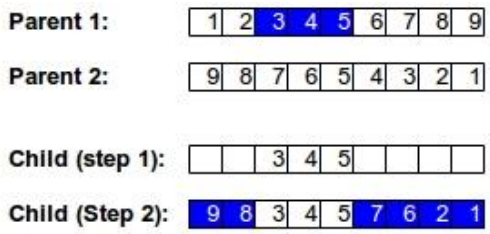

Figura 5 – Ilustração do ordered crossover. http://dtnewman.github.io/finch/genetic_algorithms.html

A mutação de cada um dos indivíduos é tratada pela função `mutate(individual,mutationRate)`. A mutação é uma das peças fundamentais para garantir a varredura ampla do espaço amostral, fazendo com que o algoritmo não convirja rapidamente para um mínimo local. Nesse caso o espaço amostral é definido pelas diversas combinações de nós que formam o caminho e por isso a mutação é implementada apenas como uma troca de posição entre os genes do mesmo indivíduo.

V. RESULTADOS

O grafo apresentado na figura 6 permite visualizar no contexto da teoria de grafos a solução obtida pelo AG após 200 gerações, os vértices estão dispostos no plano utilizando as coordenadas de cada uma das cidades, representadas pelos números em cada vértice. A figura 7 mostra o tamanho do ciclo do mais bem pontuado indivíduo a cada geração, é possível notar a grande diminuição após poucas gerações e uma certa noção de convergência. Tal característica pode indicar que com o passar das gerações, as melhores soluções começam a compartilhar sequências comuns.

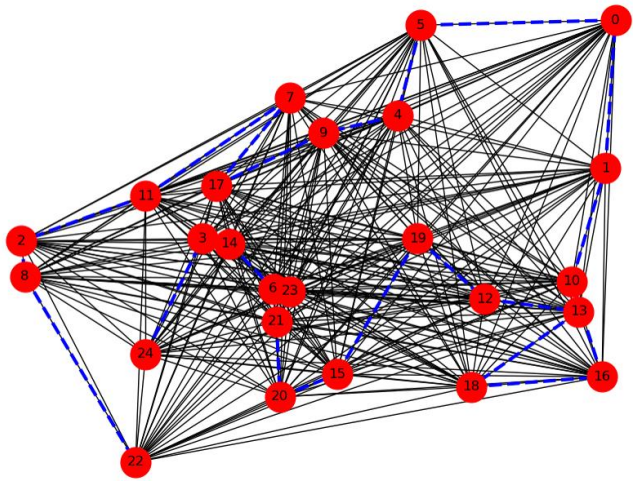

Figura 6 – Visualização em Grafo de solução, em azul, do PCV via AG

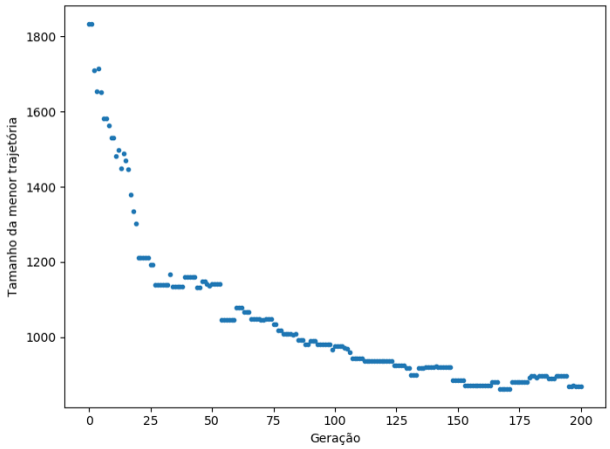

Figura 7 – Tamanho do menor caminho fechado por geração

Com o intuito de verificar a sensibilidade do algoritmo para a mesma rede o algoritmo foi executado 5 vezes e através da figura 8, é possível notar uma certa dispersão entre as respostas resultando no valor médio de 824 com desvio padrão de 18.

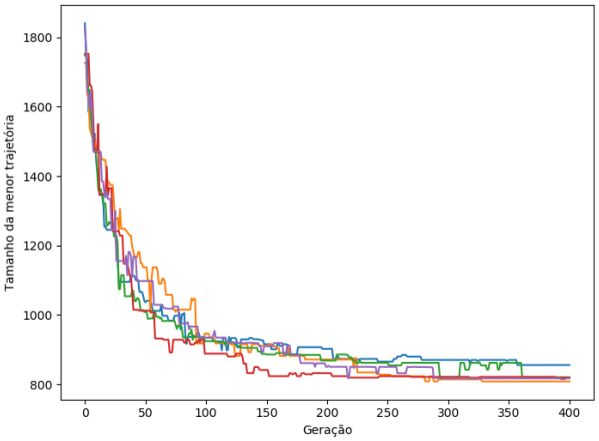

Figura 8 - Tamanho do menor caminho fechado por geração para 5 execuções.

O erro dado pelo desvio padrão entre as execuções se comparado com o histograma das distâncias associados a todas as arestas mostra que ele em si é muito menor do que grande parte das distâncias de cada uma das arestas individualmente, mostrando assim a qualidade da resposta.

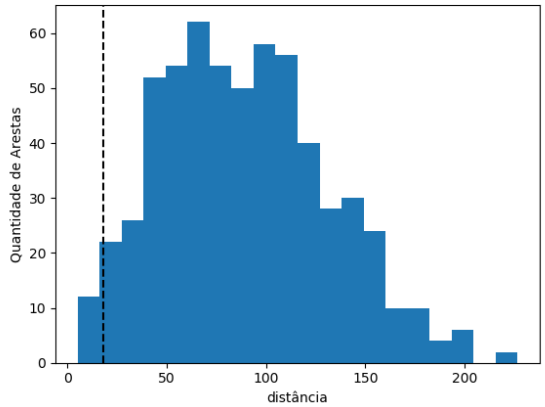

Figura 9 – Histograma das distâncias associadas a cada uma das arestas do grafo.

REFERÊNCIAS

BITTENCOURT, Guilherme. **Inteligência artificial: ferramentas e teorias**. 1998.

Studer, Rudi, V. Richard Benjamins, and Dieter Fensel. **Knowledge engineering: principles and methods.** Data and knowledge engineering 25.1 (1998): 161-198.

W.J. Clancey, **The Epistemology of a Rule-Based Expert System** - a Framework for Explanation, Artificial Intelligence 20 (1983), 215-251

R. Davis, B. Buchanan, and E.H. Shortcliffe, **Production Rules as a Representation for a Knowledge-base Consultation Program**, Artificial Intelligence 8 (1977), 15-45.