Escola Politécnica da Universidade de São Paulo
Departamento de Engenharia de Computação e Sistemas Digitais
PCS 3438 – Inteligência Artificial

Isabela Comegna dos Santos - 8994136

Relatório
EP1 - Problema do Caixeiro Viajante

São Paulo, setembro de 2018

Índice

Introdução

Este trabalho estuda uma solução para o problema do caixeiro viajante a partir de uma abordagem de solução com um algoritmo genético.

Objetivos

Os objetivos do trabalho são a aplicação de um algoritmo de inteligência artificial para encontrar uma solução (subótima, na maior parte dos casos) para o problema do caixeiro viajante. O algoritmo escolhido será o algoritmo genético.

1. Problema

O problema do caixeiro viajante, também conhecido como TSP (Travel Salesman Problem) é tal que se deseja encontrar a menor rota entre cidades de um dado conjunto, de forma que cada cidade só deve ser visitada uma única vez. Cada cidade, tem, então, uma distância conhecida até as outras cidades e a distância total percorrida deve ser minimizada.

Este problema é conhecido por ser um problema do tipo NP complexo na área de otimização combinatorial.

2. Especificações do projeto

O projeto deve ser desenvolvido na linguagem de programação R. Deve ser criado um script para a solução do problema utilizando-se apenas funções definidas pelo usuário ou disponíveis em bibliotecas do CRAN.

A função EP1_8994136, no caso deste trabalho, deve ser executada recebendo duas strings: a primeira sendo a rota para o arquivo de entrada e a segunda a rota para o arquivo de saída do programa.

O arquivo de entrada contém, na primeira linha, o nome de cada vértice, separados por vírgulas. As linhas seguintes contém o nome de um dado vértice e sua distância para cada um dos outros vértices e para ele próprio (0), separadas por vírgulas.

As distâncias possíveis são ou um número inteiro ou “Inf”, para infinita, ou seja, quando não há caminho entre dois vértices em questão.

A solução do programa deve ser escrita no arquivo de saída e impressa na tela.

3. Solução proposta

3.1 Algoritmo de busca escolhido

O algoritmo de busca escolhido é o algoritmo genético, sendo ele um algoritmo de busca local. Eles são implementados como uma simulação de computador em que uma população de representações abstratas da solução de um dado problema é selecionada para a busca de soluções ótimas.

Os algoritmos do tipo genéticos operam sobre uma população de candidatos em paralelo, buscando indivíduos em diferentes áreas do espaço de soluções. Este tipo de algoritmo é um algoritmo heurístico inspirado na teoria da evolução natural de Charles Darwin, em que os animais mais adaptados ao seu ambiente sobrevivem.

Este algoritmo funciona em cinco fases. Sendo elas:

1) população inicial;
2) função de fitness;
3) seleção;
4) crossing-over;

5) mutação.

O processo do algoritmo se inicia com um conjunto de indivíduos, chamado de população. Cada indivíduo é uma solução para o problema que se deseja resolver, sendo eles caracterizados por alguns parâmetros, conhecidos como genes. Os genes são unidos em strings a fim de formarem cromossomos.

Nos algoritmos genéticos, um conjunto de genes de um indivíduo é representado por uma string.

A função de fitness determina o quão “apto” é um indivíduo, ou seja, a habilidade deste indivíduo de competir com os outros. Essa função fornece uma pontuação de fitness para indivíduo, que é ligada à probabilidade dele ser escolhido para reprodução ou não dos próximos indivíduos. Em outras palavras, o quanto uma dada solução tem uma probabilidade maior de ser escolhida que outras soluções.

Na etapa de seleção, escolhe-se os indivíduos (soluções) mais aptos (com maior pontuação provinda da função de fitness) e faz-se a “reprodução” com eles. A reprodução dos indivíduos, ou seja, a geração de novas soluções a partir da combinação de soluções já encontradas, conta, também, com o crossing-over e a mutação.

No processo de crossing-over as soluções descendentes recebem em seus genes parte do código genético de um dos pais e parte do outro. Essa recombinação garante variedade na geração de novos indivíduos, o que faz com que possam ser mais aptos a sobreviver e assim gerar descendentes ainda mais aptos. As mutações são feitas com probabilidade mais baixa possível permitem maior variabilidade nos genes, tentando impedir que as soluções fiquem restritas a um máximo local no espaço das soluções.

3.2 Implementação do algoritmo de busca escolhido

A implementação do algoritmo genético para a solução foi baseada em um pacote disponibilizado no conjunto CRAN da linguagem R. Este pacote fornece uma

função de busca genética, para a qual são dados parâmetros com base na solução esperada.

Por tratar-se de um problema combinatório, o parâmetro “tipo” desta função foi passado como “permutation”, ou seja, permutação. Além disso, foram fornecidos, por meio dos parâmetros “lower” e “upper” os limites do espaço de soluções em que poderiam ser feitas buscas (quantidade de cidades).

A população neste problema é composta pelos indivíduos que são as possíveis rotas a serem percorridas, tendo elas como características o custo e ordem das cidades. Optou-se por não gerar aleatoriamente uma população inicial para o problema, e sim utilizar-se a matriz de entrada fornecida por simplicidade na implementação e pelo pressuposto de que a matriz de cidades fornecida para o problema será sempre escrita de forma aleatória.

Além disso, a probabilidade de mutação escolhida foi de 20%, para garantir que máximos locais da função de fitness no espaço das soluções fossem um problema para encontrar-se a solução ótima. Isso porque, com uma probabilidade de mutação relativamente alta garante-se que vários pontos do espaço de soluções sejam visitados (vários indivíduos diferentes gerados). Porém a probabilidade não é alta demais a fim de fazer com que a escolha de indivíduos para reprodução não seja significativamente importante. Assim, garantindo que a premissa de que dois indivíduos muito aptos geram outro muito apto a partir deles não seja deixada de lado. Os valores de referência para a escolha da probabilidade, na literatura (MIRANDA, 2007), são ditos entre 10% e 50%, ou seja, o valor escolhido está dentro deste intervalo, sendo ele um valor considerado bom.

O número máximo de iterações permitido foi de 5000, uma vez que é necessário equilibrar uma rota relativamente minimizada com tempo de execução relativamente baixo do algoritmo. Assim, estabeleceu-se que, a partir de 5000 iterações, o algoritmo demoraria muito para ser executado, perdendo sua função e equilíbrio entre execução e resultado.

A população de cada execução do algoritmo foi fixada em 50 indivíduos, pelo mesmo motivo descrito acima.

Para a função de fitness, ou seja, a função que calcula um resultado de avaliação heurística para a rota encontrada, definiu-se uma função própria chamada de “fitness_func”. Essa função simplesmente retorna o inverso da distância total a ser percorrida em uma dada rota que está sendo avaliada, devendo ela ser maximizada, ou seja, quanto menor a distância maior o resultado da avaliação da solução e melhor esta solução será.

A função escolhida de fitness tem como vantagem a sua simplicidade de cálculo. Sendo para sua execução necessário apenas calcular a distância total a ser percorrida em uma dada rota e seu inverso. Uma desvantagem, porém, é que para valores muito próximos uns aos outros pode ocorrer uma grande imprecisão devido ao número de casas significativas utilizadas pela linguagem R (arredondamentos e truncamentos podem fazer com que uma solução seja avaliada igual a outra mesmo que a outra seja pior).

Como saída da função do pacote utilizado, obtém-se o espaço das soluções finais ou a melhor solução, dependendo dos valores retornados da função de fitness (se forem muito próximos teremos várias soluções finais). Assim, para obter-se a melhor solução entre o conjunto final, criou-se uma função que retorna a rota cuja função de fitness é a maior possível (para fitness iguais, será retornada a primeira rota no conjunto das soluções).

3.3 Pacotes necessários

Para que seja possível executar o script da solução proposta é necessário instalar o pacote “GA”.

4. Conclusões

Neste trabalho foi possível implementar um programa para cálculo da melhor rota possível dado

um tempo de execução razoavelmente baixo para um conjunto de cidades e distâncias relativas entre elas. A implementação feita tem uma acurácia melhor ou

pior para problemas com um número de cidades diferentes, bem como para problemas com distâncias relativas muito parecidas (solução menos confiável) ou muito distintas(solução mais confiável, devido a menos imprecisões de truncamentos em cálculos e comparações matemáticas).

Nem sempre é possível que a solução ótima seja encontrada com pouco esforço computacional (tanto do hardware quanto do software).

4. Referências

MIRANDA, Márcio Nunes de. Algoritmos Genéticos: Fundamentos e Aplicações. 2007. Disponível em: . Acesso em 04 de abril de 2010.