Instituto Tecnológico de Aeronáutica

Divisão de Engenharia de Computação

CTC-17: Inteligência Artificial

Projeto II - Buscas de Melhoria Iterativa e Satisfação de Restrição

---

Aluno: Luiz Angel R. Rafael

Profº. responsável:
Paulo André Castro

19 de Setembro de 2016

1 Resultados Obtidos

1.1 Melhoria Iterativa

Ao executar o arquivo Main.py, no diretório source/n-queens, são impressos os resultados dos tabuleiros finais e o tempo de execução, para cada caso. Na *array* do estado final, cada i-ésimo elemento representa a linha em que a i-ésima rainha, situada na i-ésima coluna, se localiza. Foi utilizado o algoritmo Hill Climbing:

```
************************ 10 X 10 BOARD ************************
Final state: [4, 1, 5, 2, 9, 6, 8, 3, 0, 7]
Final cost: 0
Execution time: 0.119999885559

************************ 15 X 15 BOARD ************************
Final state: [5, 1, 12, 4, 0, 8, 13, 2, 10, 6, 14, 9, 11, 3, 7]
Final cost: 0
Execution time: 2.83400011063

************************ 20 X 20 BOARD ************************
Final state: [17, 6, 18, 7, 2, 14, 12, 15, 19, 16, 3, 10, 4, 1, 9, 0, 13, 11, 8, 5]
Final cost: 0
Execution time: 5.88800001144

************************ 25 X 25 BOARD ************************
Final state: [12, 6, 19, 2, 22, 14, 17, 9, 21, 4, 7, 3, 16, 23, 20, 15, 24, 8, 0, 13, 18, 10, 5, 1, 11]
Final cost: 0
Execution time: 2.39800000191
```

Ao executar o arquivo Main.py, no diretório source/global_maximum, são impressos o máximo global e o tempo de execução do algoritmo (foi utilizada a Têmpera Simulada):

```
********************** Simulated Annealing **********************
Calculating maximum value of a function
Maximum value:  4.0
Execution time:  18.2850000858
```

1.2 Satisfação de Restrições

Para a resolução do problema da Zebra, duas formas de modelar o problema poderiam ser utilizadas:

1. Implementação de um ou vários grafos de restrições e, a partir deles, atribuir valores para estados possíveis, até se achar o estado correto;

2. Implementação de uma árvore de possibilidades que, ao ser percorrida em profundidade, verifica cada restrição do problema, chegando à uma folha que satisfaça todas as restrições.

O modelo utilizado foi o segundo, pois achei a codificação do problema mais intuitiva e simples que no primeiro caso.

Ao executar o arquivo Main.py do diretório source/backtracking, é impressa uma tabela com o resultado final do problema:

```
********************************** Zebra's Problem Solution **********************************
 Nationalities:            ingles         espanhol        noruegues        ucraniano          japones
HousePositions:              Meio         Direita1        Esquerda1        Esquerda2         Direita2
   HouseColors:          vermelha           marfim          amarela             azul            verde
        Drinks:             leite  suco_de_laranja             agua              cha             cafe
    Cigarretes:          Winston     Lucky_Strike             Kool     Chesterfield       Parliament
          Pets:         caramujos         cachorro           raposa           cavalo            zebra
```

Pela solução acima, a zebra mora com o japonês, na última casa à direita, cuja cor é verde. E bebe-se água na primeira casa à esquerda, que é amarela e onde mora o norueguês.

2 Conclusões

A codificação do problema das n-rainhas foi interessante porque foi possível estabelecer uma comparação da implementação deste problema clássico entre Python e Prolog (disciplina de CTC-11): em Python, o código acaba sendo bem maior, porém mais claro, legível, enquanto que em Prolog, o código é bem enxuto e mais complexo de se entender.

Para o problema do máximo da função, não foi possível achar valores de máximo local diferentes do máximo global, pois partindo de vários pontos iniciais (o ponto inicial utilizado foi [20, -20]), no algoritmo implementado, variou-se o eixo x e y entre (-50, 50), então foram calculados e testados muitos valores de máximo, chegando-se sempre no valor de 4. Para reduzir o tempo de execução, esse intervalo de variação do x e do y poderia ser reduzido, porém o resultado final poderia ser próximo de 4, não exatamente 4.

Para o problema da Zebra, preferiu-se utilizar apenas uma classe genérica com a qual foram criados os objetos necessários e uma lista de permutações que significava todos os casos possíveis (ou seja, percorrendo a árvore de possibilidades até achar a solução correta).

3 Descrição

Na pasta do projeto, encontra-se a seguinte composição:

```
....
source
    n-queens
        ....
    global_maximum
        ....
    backtracking
        ....
```

Os arquivos principais são aqueles cujos nomes começam com Main. Os arquivos de classes (que especificam uma classe em Python) são todos exceto os principais e os arquivos Helper.py. Estes são arquivos com funções auxiliares utilizadas por arquivos principais e/ou de classe.

Para obter resultado de algum algoritmo, basta executar python <NomeArquivoPrincial>:

```
> python Main.py
```

4 Apêndice

O repositório *Git* que possui todo o código do projeto pode ser encontrado em

`https://github.com/Luizangel50/CTC-17_Lab2`