Relatório Trabalho Prático 3 – Problema das N Rainhas – Laboratório de Inteligência Artificial

Diego Ascânio Santos

1. Introdução

O problema das N-Rainhas consiste em dispor N rainhas (do jogo de xadrez) em um tabuleiro de casas negras e claras de dimensão NxN de forma que as rainhas não se ataquem em nenhum momento, respeitadas as movimentações da peça (Horizontal, Vertical e Diagonal). Existem várias abordagens para se resolver esse problema, tais como a abordagem back track (um tipo de abordagem por força bruta) e abordagens utilizando algorítmos genéticos e algorítmos de subida de encosta avaliando funções de fitness, sendo as últimas duas as que foram implementadas para solucionar esse problema.

2. Conceitos:

Função de Fitness:

Uma função matemática que mapeia um determinado valor de entrada (ex: disposição das linhas que as rainhas ocupam em cada coluna do tabuleiro – [0, 7, 2, 1, 4, 5, 3, 6]) e retorna um valor numérico para essa entrada. A finalidade desse valor numérico gerado por essa função é a de definir o quão útil ou ótima é uma entrada para a solução de um problema e ela é usada pelos algoritmos genético e de subida de encosta para encontrar a solução ótima (local ou global) de um problema.

Algoritmos Genéticos (AG):

São algoritmos baseados em mecanismos de seleção natural (Darwin) e estruturas genéticas (Mendel) utilizados para resolução de problemas de otimização e busca. Foram propostos inicialmente em 1975 pelo professor John Holland, da universidade de Michigan.

Algoritmos de Subida de Encosta:

É um algoritmo de busca local que não armazena a es utilizado para encontar o estado objetivo de um problema de otimização / busca sem se importar com a sequência de passos tomada. Esse algoritmo consiste em uma repetição que percorre o espaço de estados no sentido do valor que atinge o objetivo da função (Crescente – maximização, Decrescente – minimização) e termina quando um pico (Maximização) ou vale (Minimização) são encontrados – pontos onde a derivada da função objetivo é igual a 0. Pelo fato do algoritmo não armazenar a estrutura de dados de busca, duas analogias bem simplórias podem ser utilizadas para explicar o conceito do algoritmo. A primeira, oriunda da transparência da disciplina compara o algoritmo ao ato de “Subir o monte everest em meio a uma crise de amnésia”, podendo ser aplicada aos problemas de maximização. A segunda definição, oriunda da experiência de eventos não tão agradáveis vivenciados em estádios de futebol, compara o algoritmo a “Descer um barranco do mineirão antigo correndo de um arrastão da torcida do flamengo sem nem olhar pra trás e sem nem saber o caminho tomado para tal.”, podendo ser aplicada a problemas de minimização.

3. Experimento

O experimento consistiu em implementar:

1. Função Fitness
2. Algoritmo de Subida de Encosta
3. Algoritmo Genético
4. Junção híbrida entre AG e Subida de Encosta

E após a implementação da função fitness, dos algorítmos e da junção híbrida entre os dois, testou-se a execução dos algoritmos 10 vezes para tabuleiros nxn de tamanhos: n = (6, 8, 16, 32, 64, 128, 256, 512) para pegar o melhor indivíduo encontrado, o pior indivíduo, a média dos indivíduos e o desvio padrão da amostra.

4. Estratégias de implementação.

1. **Função Fitness**
   Dado o objetivo do problema das n-rainhas, o ideal é que em um tabuleiro de casas negras e brancas, o número de rainhas que não estejam se atacando seja máximo. Pode-se calcular o número de ataques entre rainhas em um tabuleiro à partir das coordenadas que elas ocupam em um tabuleiro nxn, usando duas abordagens: A distância entre um par de rainhas ou a inclinação da reta existente entre as coordenadas de um par de rainhas. A primeira abordagem, utilizada inicialmente, consistiu em calcular as distâncias que cada rainha em um tabuleiro mantinha para as outras e averiguar se essa distância era um número inteiro (se o resto da divisão da distância por 1 dava 0 – configurando um ataque horizontal ou vertical) ou se essa distância era um múltiplo de $\sqrt{2}$ (se o resto da divisão da distância por $\sqrt{2}$ dava 0 – configurando um ataque diagonal). Se as condições supracitadas fossem verdadeiras, configurava-se um ataque e portanto a função fitness diminuia em 1 o valor da variável interna pares (inicialmente valendo n**2 e tendo valor ótimo de n*(n-1), uma vez que cada rainha atacava a si própria, já que a distância para ela mesma era um número inteiro). Conceitualmente, a primeira abordagem era correta e válida, mas na prática isso não aconteceu por conta da implementação de operações de ponto flutuante da linguagem python que considerou como ataques alguns movimentos que não eram de ataque, por problemas na precisão de ponto flutuante que independente da linguagem de programação irão acontecer, por conta do padrão IEEE 754.
   Como a primeira abordagem não ofereceu valores concisos, foi necessária pensar em outra abordagem mais simples para servir com solidez o propósito da função fitness. À partir disso, pensou-se em utilizar o cálculo da inclinação das retas entre duas rainhas dispostas em um tabuleiro para averiguar se elas se atacam ou não. Essa abordagem mostrou-se muito mais simplificada e sólida do que a primeira e independentemente dos valores das coordenadas das rainhas em um tabuleiro, o valor da inclinação das retas é sempre a tangente do Ângulo de inclinação delas. Os angulos que configuram ocasiões de ataque entre as rainhas são somente 3: 0°, 45° e 90° e os valores da tangente de cada um desses ângulos são respectivamente 0, 1 e infinito e esses valores foram considerados para a função de fitness avaliar se uma situação de ataque exisitia ou não. Quando essas condicções de ataque eram verdadeiras, assim como na primeira abordagem, o valor da variável interna pares diminuia em 1.

2. **Algorítmo de subida de encosta**

   O algoritmo de subida de encosta foi implementado tal qual o exemplo que encontra-se na transparência utilizada em sala de aula. Sendo uma função que recebe como parâmetros: s – um indivíduo a ser avaliado inicialmente, N – conjunto de vizinhos de s que podem ser úteis para adentrarem a vizinhança e f – um objeto apontando para uma função de fitness a ser avaliada, apontando por padrão para a função fitness. Quando o algoritmo de subida de encosta é chamado, ele avalia o indivíduo inicial s e todos os possíveis vizinhos que podem ser úteis para encontar um pico da função, ou seja, todos aqueles cujo o valor de fitness seja maior ou igual ao valor do fitness de s. À partir daí, ele passa a explorar a vizinhança de s e atualiza a variável s para o indivíduo de maior valor na vizinhança. E esse processo é repetido até não existir mais vizinhos úteis (que tenham valor maior ou igual a s). No final, o indivíduo s de maior fitness é retornado e a execução finaliza.

3. **Algoritmo Genético**
   O algoritmo genético também foi implementado como os exemplos estudados em sala de aula, porém, com algumas alterações, como a adição do parâmetro x para indicar a quantidade de valores mais aptos a serem retornados para a junção hibrida do AG e do subida de encosta e tal como o algoritmo de subida de encosta, também existe um objeto f para apontar para uma função de fitness a ser avaliada – por padrão a fitness já implementada. O processo de funcionamento do algoritmo genético é exatamente o mesmo que o visto em sala, onde, um indivíduo inicial é gerado como uma disposição de rainhas nas colunas sem repetir o número das linhas (ex.: [0,1,2,3,4,5,6,7]) e uma população inicial do tamanho de população definido pelo usuário é gerado à partir de seleções aleatórias de permutações desse indivíduo. À partir daí, ocorre o processo de seleção e pareamento dos indivíduos a serem reproduzidos (de maneira aleatória) para a partir de então ocorrer o processo de cross-over com ponto de corte de 1 coluna – modificado para corte ox após consulta ao professor, para garantir a permutação. Depois, se a probabilidade mutação for atingida, ocorre uma mutação. Após essa mutação uma nova população é gerada e então, os indivíduos para fazer parte da nova população são selecionados pelo método da roleta. O algoritmo repete esses passos até o numero máximo de gerações definido pelo usuário ou até encontrar o valor máximo de função objetivo, que é (n-1)*n rainhas que não se atacam no tabuleiro.

4. **Junção Hibrida.**
   A junção hibrida nada mais é que rodar o AG, selecionar os x indivíduos mais aptos gerados pelo AG, construir vizinhanças para esses indivíduos através de permutações e colocar esses indivíduos como entradas do algorítmo de subida de encosta para ele, a partir de uma vizinhança mais seleta, contribuir na busca pelo indivíduo ótimo para a resolução do problema.

5. Execuções e resultados obtidos

Para se efetuar os testes necessários, foi criado um bash script chamado bateria_testes.sh que recebe como parâmetro o tipo de solução a ser feita pelo programa n_rainhas.py (1 – Subida de Encosta, 2 – AG, 3 – Hibrido) bem como o numero de indivíduos mais aptos a serem considerados no AG para a aplicação do Hibrido.

Esse script consistiu em rodar o programa em python para os seguintes valores de n: (6, 8, 16, 32, 64, 128, 256 e 512) por uma quantidade de execuções de y vezes (no caso 10 pelo enunciado do exercício) para validar as soluções implementadas. Alguns parâmetros eram fixos como por exemplo: tamanho da população (100), gerações (30) e probabilidade de mutação (0.25). A saída padrão do script foi direcionada para arquivos de texto que armazenaram os resultados, os resultados foram processados, analisados e encontrados seus máximos, mínimos, médias e desvios padrões para cada valor de n.

Abaixo a tabela comparativa mostrando os resultados obtidos:

	N = 6				N = 8				N =16				N = 32			
Solução	Melhor Caso	Pior Caso	Media	Desvio Padrão	Melhor Caso	Pior Caso	Media	Desvio Padrão	Melhor Caso	Pior Caso	Media	Desvio Padrão	Melhor Caso	Pior Caso	Media	Desvio Padrão
Subida de Encosta	30	26	28	0,943	54	46	50,6	2,12	208	162	192,6	16,25	526	438	506,6	28,88
Algoritmo Genético	30	28	29.2	1.03	56	54	55.4	0,967	220	218	218.8	1,03	806	800	804,4	2,63
Solução Hibrida	30	28	28,4	0,8	54	54	54	0	230	224	227	2,72	854	820	845,62	10,13

	N = 64				N = 128				N =256				N = 512			
Solução	Melhor Caso	Pior Caso	Media	Desvio Padrão	Melhor Caso	Pior Caso	Media	Desvio Padrão	Melhor Caso	Pior Caso	Media	Desvio Padrão	Melhor Caso	Pior Caso	Media	Desvio Padrão
Subida de Encosta	1268	948	1171,8	104,87	2676	1970	2509,8	270,49	5494	4998	5393	206,09	11128	9122	10922,4	632,62
Algoritmo Genético	2846	2724	2831	40,99	10032	9790	9980,6	75,03	36654	36162	36600	146,0411	139056	139042	139049,2	4,58
Solução Hibrida	2966	2878	2929,12	22,37	10268	9992	10227,5	70,48	37152	36440	36903,9	263,37	140062	139018	139947,7	302,34

Conclusões:

O algoritmo de subida de encosta por si só, para valores muito grande de n fica bem abaixo da solução do problema n*(n-1), algo que o genético já se aproxima com mais facilidade. Porém, quando as duas soluções são combinadas, o de subida de encosta contribui para a partir dos melhores indivíduos obtidos no algoritmo genético, pesquisar vizinhanças deles não exploradas no AG que podem retornar indivíduos mais aptos.

Referências Bibliográficas:

1. Slides da disciplina
2. Documentação oficial do Python 2.7: https://docs.python.org/2.7/