Simulated Annealing para o problema das N rainhas.

Vinicius Oliverio[1], Lucas P. Simões[1], Rafael S. Parpinelli[1], Claudio C. de Sá[1]

[1]Departamento de Ciência da Computação – Universidade do Estado de Santa Catarina (UDESC) – Campus Universitário Prof. Avelino Marcante s/n - Bairro Bom Retiro – CEP 89223-100 – Joinville – SC – Brasil.

viniciusoliverio6@hotmail.com, lucas_tiotuts@yahoo.com, {dcc2rsp,claudio}@joinville.udesc.br

Abstract. *This paper has as objective to propose an approach on solving the problem of the N queens using the Simulated Annealing algorithm, proposed by S. Kirkpatrick et al, that is a probabilistic algorithm based on the annealing process. In chess, a queen can move as far as it wants horizontally, vertically and diagonally. The problem consists in putting N queens in a NxN chess board in a way that none of the queens will be able to attack another one with only one movement. This problem was modeled and the algorithm was implemented so that is possible to analyze its performance with the proposed problem.*

Resumo. *Este artigo tem como objetivo propor uma abordagem para a resolução do problema das N rainhas utilizando o algoritmo Simulated Annealing, proposto por S. Kirkpatrick et al, que é um algoritmo probabilístico baseado no processo de anelamento. No xadrez, uma rainha pode se mover o quão longe quiser horizontalmente, verticalmente ou diagonalmente. O problema consiste em alocar N rainhas em um tabuleiro NxN de modo que nenhuma das rainhas possa atacar uma outra com somente um movimento. Esse problema foi modelado e o algoritmo foi implementado de modo que é possível analisar seu desempenho para o problema proposto.*

Palavras-Chave: simulated annealing, rainhas, anelamento, combinatorial.

1. Introdução

No xadrez, uma rainha pode se mover o quão longe quiser, horizontalmente, verticalmente ou diagonalmente. Um tabuleiro de xadrez possui 8 linhas e 8 colunas. No problema original das 8 rainhas, deve-se alocar 8 rainhas em um tabuleiro comum de xadrez de modo que nenhuma das rainhas possa atacar uma outra com somente um movimento. Neste artigo, é utilizada uma variação deste problema, que consiste em alocar N rainhas em um tabuleiro NxN (N por N, tal que N > 3) de modo que nenhuma das rainhas possa atacar uma outra com somente um movimento.

O objetivo desse artigo é propor uma solução para esse problema combinatorial utilizando o *simulated annealing*, que é uma meta-heurística para resolução de problemas de otimização combinatorial, proposto por [Kirkpatrick et al. 1983] e baseado no processo de recozimento que é o processo pelo qual o metal é aquecido a uma determinada temperatura, então o mesmo é deixado para resfriar lentamente,

suavizando o metal, o que significa que ele pode ser cortado e moldado mais facilmente. A idéia principal do algoritmo é de permitir movimentos que resultem em soluções de pior qualidade que a solução atual de modo a escapar de mínimos/máximos locais. A probabilidade de realizar tal movimento é decrescida ao decorrer da busca.

O algoritmo começa por criar uma solução inicial (aleatoriamente ou heuristicamente) e associar a esta solução, uma temperatura (T) inicializada com um valor aleatório. Então o seguinte é repetido até o critério de parada ser satisfeito: Uma solução vizinha a solução atual é criada (com base na solução atual), então ela é avaliada por uma função de energia, que quanto menor o seu valor melhor a solução, para depois ser comparada a energia da solução atual, se ela for melhor, a solução é aceita como nova solução atual, se ela for pior, essa solução tem uma probabilidade de ser aceita. O pseudo-algoritmo para este algoritmo é:

```
s := GeraSoluçãoInicial( )  /* uma solução inicial qualquer para o problema */
T := T_0                    /* uma temperatura aletoriamente inicial */
ENQUANTO condições de termino não são satisfeitas (s != sfim)
        s' := EscolheAleatoriamente(N(s))
                SE f(s') < f(s)
                        s := s'
                SENAO
                        Aceita como nova solução com a probabilidade p(T,s',s)
                FIM SE
        Atualiza(T)
FIM ENQUANTO
```

Como pode ser observado, esse algoritmo é de fácil implementação e compreensão pois trabalha com uma única solução, que é estocasticamente avaliada segundo uma função de avaliação do problema. Para o caso do problema proposto, a função escolhida foi a contabilização da quantidade de rainhas em condição de ataque.

2. Abordagem/Modelagem do Problema

O primeiro passo a ser dado é modelar o problema, e para esse problema, foram pensadas duas opções de modelagem, uma onde se têm N rainhas, sendo que cada uma teria sua posição X e sua posição Y que representariam respectivamente coluna e linha, ou, outra onde se têm um vetor de inteiros com N posições, onde cada posição desse vetor seria uma linha e o valor dessa posição do vetor seria uma coluna, representando assim a posição de uma rainha. A modelagem escolhida foi a segunda, pois por meio desta pode-se restringir o espaço de busca, eliminando a possibilidade de qualquer rainha ser posicionada em uma coluna que já está ocupada, sendo assim mais simples de ser adaptada para o *simulated annealing*.

Depois de modelado o problema, deve-se partir para o algoritmo. As partes fundamentais desse algoritmo são, a geração da solução inicial, o método de avaliação da solução (calculo da energia), o método de perturbação da solução atual (que gera a nova solução) e o método de decremento da temperatura T.

Na abordagem tomada, a solução inicial (o vetor de inteiros) é gerada contendo cada posição do vetor o número da posição, ou seja, a posição 1 do vetor tem valor igual a 1, a posição 2, tem valor igual a 2, assim por diante até N. A solução sendo gerada dessa forma são eliminadas algumas possibilidades de colisões e permite facilmente ser explorado todo o espaço de busca desse problema com apenas a troca dos valores das posições do vetor, que é o método escolhido para causar a perturbação na solução atual. Esse método escolhe aleatoriamente duas posições dentre as N posições do vetor, e faz a troca de valores entre elas, a primeira posição escolhida recebe o valor da segunda e vice-versa. Depois de efetuado esse passo, deve-se avaliar a nova solução, e para essa avaliação o método escolhido foi verificar quantos ataques ocorrem na nova configuração do ambiente, sabe-se que as rainhas podem se mover o quão longe quiser, horizontalmente, verticalmente e diagonalmente, então se alguma rainha estiver na diagonal de outra, está ocorrendo uma colisão, não é necessário verificar se elas estão na mesma linha ou na mesma coluna, pois pela modelagem tomada isso é impossível. E por fim, o método de decremento da temperatura que neste caso foi utilizado o valor da temperatura da solução atual, assim sendo, quando a solução atual tiver energia igual a zero nenhuma colisão ocorre, portanto essa solução é uma solução ótima, então o algoritmo pode parar.

3. Resultados

Efetuada a modelagem do problema e do algoritmo, eles foram implementados na linguagem de programação Java. O programa desenvolvido, ao fim da computação, imprime a posição das rainhas, como pode ser visto na figura 1 e o gráfico da oscilação de temperatura por número de iterações, como mostra a figura 2, com esses dados pode-se chegar as conclusões de desempenho desse algoritmo.

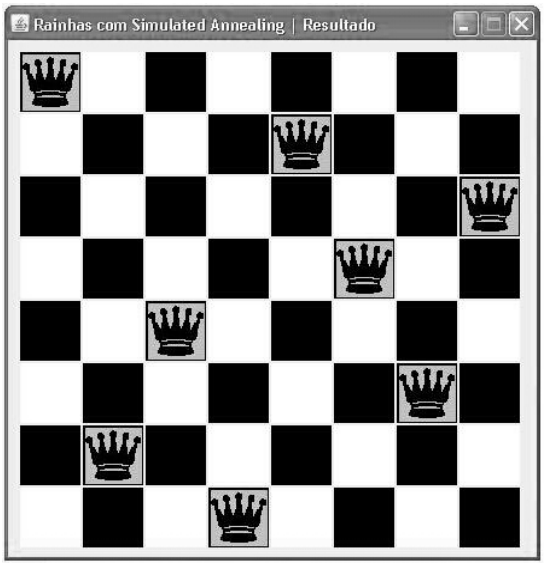

Figura 1. Tabuleiro resolvido

Como pode ser visto na figura 1, o programa resolve o tabuleiro e após demonstra para o usuário de uma forma clara, posteriormente ele mostra o gráfico escolhido pelo usuário como pode ser visto na figura 2.

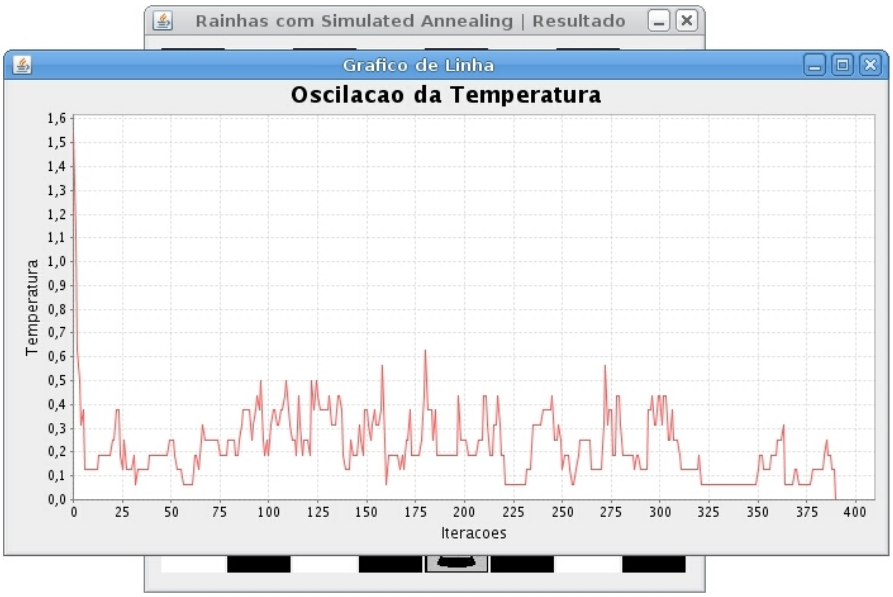

Figura 2. Gráfico de oscilação da temperatura.

Foram realizados testes com N variando entre 4 e 16. Para cada determinado número de rainhas, o programa foi processado 50 vezes. Ao final desse processamento foi feito a média de iterações necessárias para a resolução do problema. Uma vez realizado os testes, os dados são mostrados na tabela 1.

Tabela 1. Numero de Rainhas e Iterações

Rainhas	Iterações
4	14
5	10
6	156
7	132
8	236
9	326
10	1603
11	3461
12	10896
13	23589
14	41399
15	93856
16	91977

A figura 3 demonstra o gráfico resultante da tabela 1, onde pode-se ver mais claramente a curva exponencial de iterações realizadas pelo algoritmo para resolver os problemas de 4 a 16 rainhas.

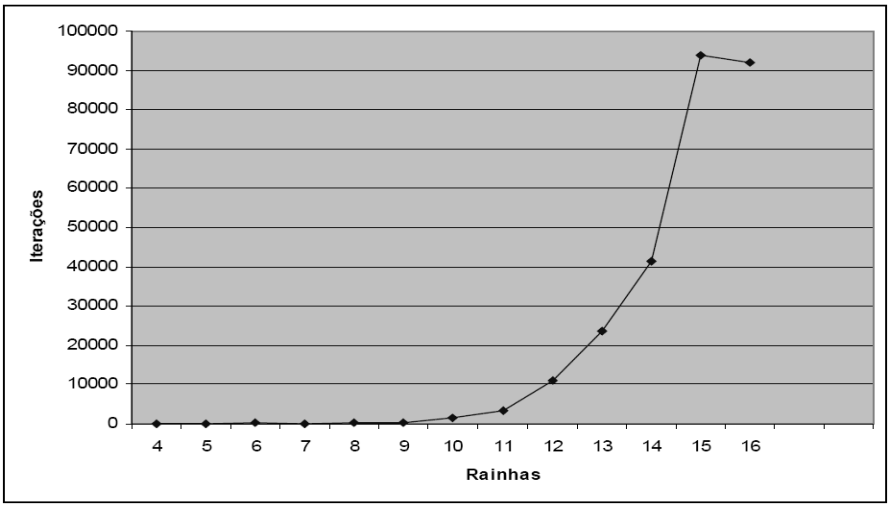

Figura 3. Número de iterações x Número de Rainhas.

Como se pode observar o algoritmo é muito eficiente para este caso e resolve o problema em poucas iterações se comparado a outros métodos, como por exemplo, em (Peter,1997).

4. Conclusões

O problema das N rainhas é um problema conhecido, de fácil entendimento mas de difícil resolução manualmente a medida que cresce o número de rainhas, e por isso um programa para resolver esse problema em alguns casos é necessário. Ele sendo um problema combinatorial, é facilmente modelado para o *simulated annealing* que, como visto, é um algoritmo de fácil implementação e eficiente para problemas combinatoriais, como é o caso do problema das N rainhas, ele resolveu o problema em poucas iterações se comparado a outros métodos.

Referências

Alfed Peter. (1997) “The N by N Queens Problem”. Departament of Mathematics, College of Science, University of Utah. http://www.math.utah.edu/~alfeld/queens/queens.html, Julho.

Blum, Christian, IRIDIA and Roli Andrea. “Simulated Annealing”. Università di Bologna. http://iridia.ulb.ac.be/~meta/newsite/index.php?main=3&sub=34#Problem, Julho.

Carr, Roger. "Simulated Annealing.". From MathWorld--A Wolfram Web Resource, created by Eric W. Weisstein. http://mathworld.wolfram.com/SimulatedAnnealing.html, Julho.

França, Paulo Morelato. “Simulated Annealing”. Unicamp. www.densis.fee.unicamp.br/~franca/EA043/Transpa-sa.pdf, Julho.

Kirkpatrick, S., C. D. Gelatt Jr., M. P. Vecchi, (1983) "Optimization by Simulated Annealing", Science, 220, 4598, 671-680.

Ryan V. (2005) “Annealing Metals”. http://www.technologystudent.com/equip1/heat3.htm, Julho.