Proposta de solução do Problema de Dominação de Rainhas Utilizando ILS

Proposed solution to the Minimum Dominating Set of Queens Problem using ILS

Maria Edoarda Vallim Fonseca Thales Athayde Santos

2018, v-1.0.0

Resumo

Neste artigo, propomos uma solução para o Problema de Dominação de Rainhas usando Busca Iterativa Local e comparamos nossos resultados com uma solução prévia utilizando Algoritmo Genético.

Palavras-chave: Problema de Dominação de Rainhas. Busca Iterativa Local. Metaheurística.

Abstract

In this paper, we propose a solution to the Dominating Set of Queens Problem using Iterative Local Search and compare our results with a previous solution using Genetic Algorithm.

Keywords: Dominating Queen Problem. ILS. Metaheuristic.

Data de submissão e aprovação: elemento obrigatório. Indicar dia, mês e ano

Identificação e disponibilidade: elemento opcional. Pode ser indicado o endereço eletrônico, DOI, suportes e outras informações relativas ao acesso.

1 Introdução

O problema de dominação de rainhas é muito bem conhecido dentre os problemas de xadrez. Nele, dado um tabuleiro $n \times n$, temos n quadrados dispostos nas linhas e n quadrados nas colunas. Quando uma rainha Q é disposta no tabuleiro, ela domina a linha, a coluna, e as diagonais que passam pela sua posição. **??** O objetivo do problema é descobrir a disposição da menor quantidade de rainhas possível de forma à dominar todo o tabuleiro.

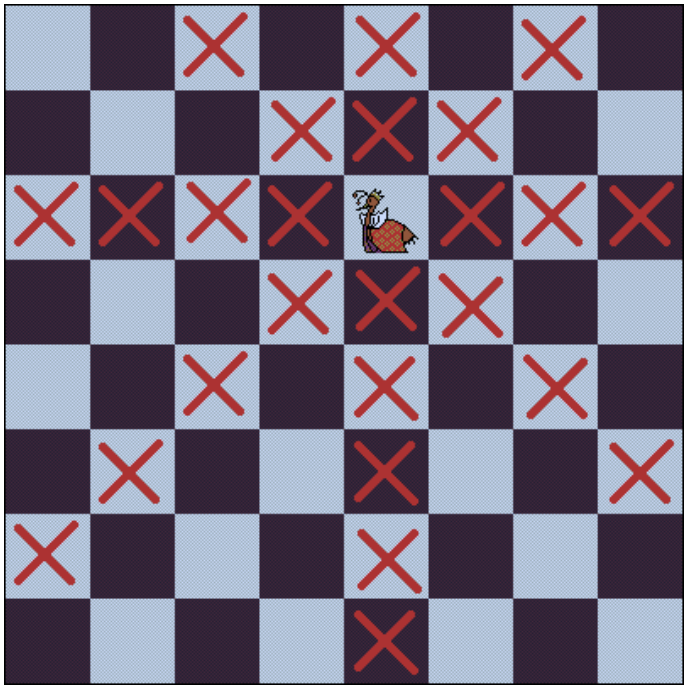

Figura 1 – Exemplo da linha de dominação de uma rainha em um tabuleiro de xadrez 8×8

Algoritmos evolucionários provaram ter sucesso para resolver e otimizar uma grande variedade de problemas complexos, incluindo problemas combinatórios, como o estudado aqui, em um tempo computacional razoavelmente aceitável. (**??**)

Local Search, ou Busca Local, é um método heurístico para resolver problemas computacionalmente difíceis. Busca local pode ser usada em problemas que possam ser formulados como achar a solução maximizando um critério entre várias soluções possíveis. Algoritmos de busca local movem de solução à solução no espaço de soluções possíveis aplicando mudanças locais, até uma solução dita ótima ser encontrada. (**??**)

Um dos problemas da Busca Local é que ela pode ficar presa em um mínimo local, sem conseguir melhorar seu resultado. Para contornar esse problema, utiliza-se *Iterative Local Search*, ou Busca Local Iterativa. Essa modificação consiste em iterar sobre chamadas da busca local, cada vez começando de um ponto diferente do conjunto de solução perturbando o mínimo local atual de modo que faça a solução chegar em outro ótimo local. Esta perturbação não pode ser muito forte nem muito fraca, pois corre o risco dela acabar encontrando o mesmo mínimo local ou servir como uma inicialização aleatória. (**??**)

Neste artigo, propomos uma solução baseada em *Iterative Local Search* para o Problema de Dominação de Rainhas. Nossos resultados serão comparados diretamente com o método descrito em (**??**), que utiliza Algoritmo Genético e é o mais comumente encontrado para solucionar este problema. Depois disso, iremos debater os resultados alcançados.

Figura 2 – Exemplo de um tabuleiro de tamanho 8×8 sendo completamente dominado por quatro rainhas

2 Motivação

Decidimos escolher o *Dominating Set of Queens Problem* por ter sido um dos temas abordados por um dos membros do grupo durante as apresentações de trabalho da disciplina, o que nos deu um certo grau de familiaridade com o assunto. Pesquisando mais à fundo, vimos que as soluções mais comuns para a solução do Problema de Dominação de Rainhas eram com *Backtracking* e Algoritmo Genético. Além disso, de acordo com (**??**), existem muitos artigos procurando os limites superior e inferior do problema, mas não existe muito esforço de pesquisa na busca de soluções práticas para o problema.

Visto essa situação, decidimos propor uma solução baseada em *Iterative Local Search* para o problema e comparar os resultados com uma solução em Algoritmo Guloso.

3 Trabalhos Relacionados

Um problema semelhante, proposto em 1850, conhecido como problema das n-rainhas teve muitos esforços focados nele para sua solução. O problema das n-rainhas é descrito como: dado um tabuleiro $n \times n$, qual seria a disposição das rainhas de modo que nenhuma rainha consiga atacar a outra. Houve muita pesquisa em torno deste problema, e suas soluções utilizam desde teoria matemática até teoria dos grafos. O estudo desse tipo de problema pode beneficiar várias áreas como controle de tráfego, prevenção de *deadlocks* e armazenamento de memória paralela. (**??**)

Muitas soluções para o problema das n rainhas foram propostas, dentre elas backtracking, redes neurais e algoritmos evolucionários.

Em contrapartida, o problema de dominação de rainhas não recebeu tanta atenção

dentre os pesquisadores das ciências da computação. Na realidade, várias pesquisas se preocuparam em pesquisar o problema porém só adotaram modelos matemáticos. (**??**) Burger e Mynhart apontaram que o problema de dominação das rainhas é um dos mais difíceis problemas de xadrez. (**??**) Esse problema é comumente definido como achar o menor número possível de rainhas em um tabuleiro $n \times n$ que consigam a dominação total do tabuleiro. Esse número é conhecido como número de dominação e sua notação é $\gamma(\mathbf{Qn})$. Muitos pesquisadores focaram em descobrir os limites superior e inferior do problema desde a década de 70, e o número mínimo possível de rainhas para solucionar o problema ($\gamma(\mathbf{Qn})$) foi calculado para diversos tamanhos de tabuleiro (vide tabela **??**) (**????????**).

Tabela 1 – Número mínimo de rainhas para dominação total $\gamma(\mathbf{Qn})$ de um tabuleiro de tamanho n.

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\gamma(\mathbf{Qn})$	1	1	1	3	3	4	4	5	5	5	5	7	7	$\leq$8	$\leq$9	$\leq$9	9	9

4 Formulação do Problema

Para este problema ser usado num computador tivemos que fazer as seguintes coisas.

Cada casa do tabuleiro é representada por uma tupla (x, y) onde obviamente x é o eixo x e y é o eixo y. Com isso cada rainha é representada por um (x, y) que é sua posição no tabuleiro.

Para representar um individuo a gente usa um conjunto $\mathbf{Q}=(\mathbf{x}_1, y_1), (x_2, y_2), .., (x_n, y_n) onde a tupla (\mathbf{x}_1,$

A nossa metodologia foi utilisar um *ILS* basico. Ele gera um início randômico, passa ele para o *Local Search*, salva o individuo se o fitness dele for o melhor. Em seguida ele pega esse ultimo individuo gerado pelo *Local Search* e o modifica usando nossa função de *Perturbação* para em seguida enviar esse novo individuo para o *Local Search* fazendo um loop. Este exemplo pode ser visto no Pseudocódigo ??.

5.1 Fitness

Nosso fitness é baseado no fitness do paper de algoritimo genérico (??).

Dado o conjunto D

O *ILS* começa gerando um inicio randômico, onde ele bota x rainhas no tabuleiro onde x é uma variável do programa. A única restrição deste inicio aléatorio é que duas rainhas não podem ocupar o mesmo quadrado.

Após isso, o inicio randômico é mandado para o *Local Search* que foi implementado vendo a movimentação de cada rainha para cada direção com 1 de distância (direita, esquerda, cima, baixo e diagonais) ??. Motivo desta implementação é o fato de cada possível movimentação virar uma matriz, e não um vetor, contendo todas as posições a um de distancia de cada rainha no tabuleiro. Embora usar um *Local Search* com distâncias maiores gere resultados melhores, isso teria a consequência negativa de aumentar exponencialmente o tempo computacional da execução do algoritmo.

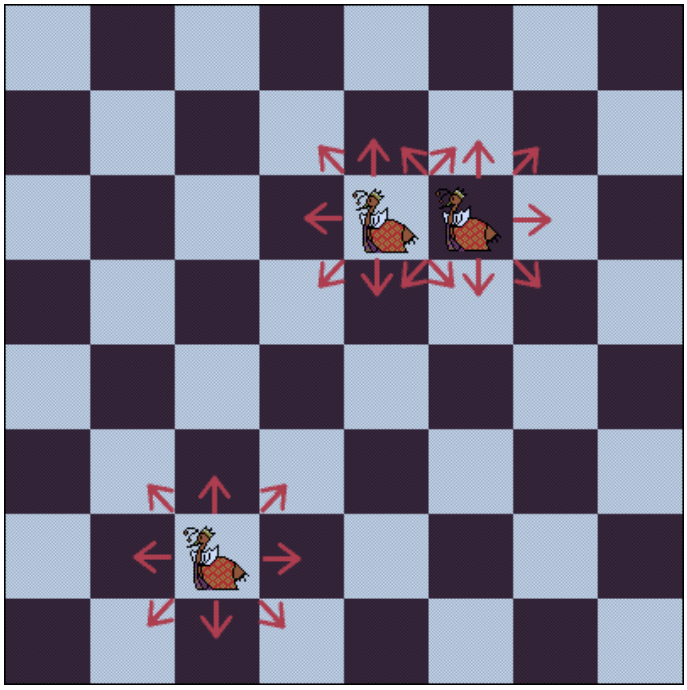

Figura 3 – Exemplo de possiveis movimentações de três rainhas no local search e na perturbação

Pseudocódigo do ILS

Data: rainhas, best, maxIterações
Result: best - Melhor solução
$rainhas \leftarrow randomStart(rainhas)$;
$rainhas \leftarrow LocalSearch(rainhas)$;
$best \leftarrow rainhas$;
repeat
  $rainhas \leftarrow perturbation(rainhas)$;
  $rainhas \leftarrow LocalSearch(rainhas)$;
  if $fitness(rainhas) > fitness(best)$ **then**
    $best \leftarrow rainhas$;
  end
until $maxIterações\ OU\ fitness(best) = 1$;

Algorithm 1: Pseudocódigo do algoritmo de ILS utilizado

6 Resultados

Nossos testes foram rodados em uma máquina Intel Core i5-7200U com 8GB de RAM, usando o sistema operacional Manjaro Linux com o pacote gráfico KDE Plasma. A linguagem de programação utilizada foi Python 3.7.1.

Data: population, best
Result: best - Melhor fitness na população
$population \leftarrow randomStart()$;
$populationFitness \leftarrow fitness(population)$;
repeat
  $population \leftarrow algoritmoGenetico(population)$;
  $populationFitness \leftarrow fitness(population)$;
until $maxIterações\ OU\ \exists populationFitness = 1$;
$best \leftarrow bestIn(population, populationFitness)$;

Algorithm 2: Pseudocódigo do Algoritmo Genético utilizado

7 Conclusão

Text...