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. [Vide Figura 1] O objetivo do

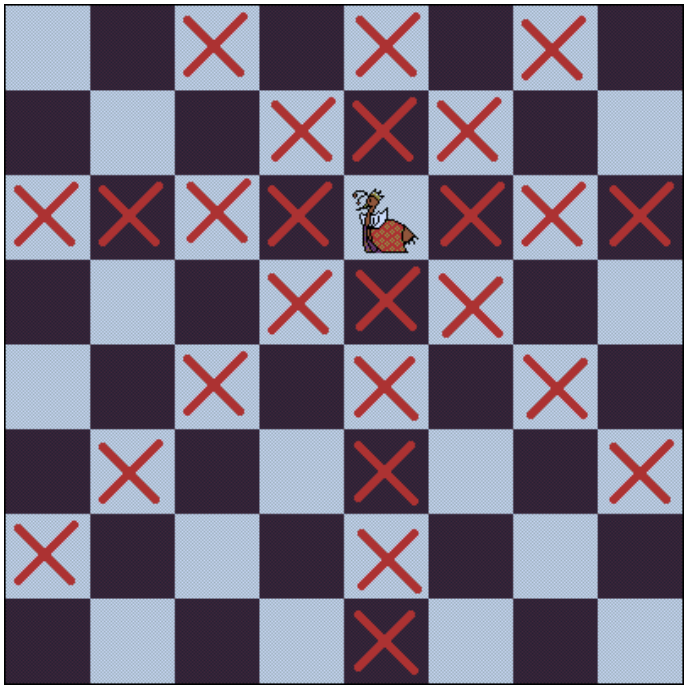

Figura 1 – Exemplo da linha de dominação de uma rainha em um tabuleiro de xadrez 8×8

problema é descobrir a disposição da menor quantidade de rainhas possível de forma à dominar todo o tabuleiro.

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. (DOERR et al., 2011)

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. (HOOS; STÜTZLE, 2004)

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. (LOURENÇO; MARTIN; STÜTZLE, 2010)

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 (ALHARBI; VENKAT, 2017), 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 (ALHARBI; VENKAT, 2017), 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. (BELL; STEVENS, 2009)

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. (ALHARBI; VENKAT, 2017) Burger e Mynhart apontaram que o problema de dominação das rainhas é um dos mais difíceis problemas de xadrez. (BURGER; MYNHARDT, 2002) 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 1] (COCKAYNE, 1990; BURGER; MYNHARDT, 2002; WEAKLEY, 2002; GIBBONS; WEBB, 1997).

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, onde x e y podem ser um número de 0 até n - 1, onde o tamanho total do tabuleiro é representado por $n \times n$. Com isso cada rainha é representada por um (x, y) que é sua posição no tabuleiro.

Para representar um indivíduo usaremos um conjunto Q de rainhas de modo que $Q = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$ onde a tupla (x_1, y_1) representa a rainha de índice 1.

Cada rainha por sua vez possui um conjunto D que representa todos as posições de dominação daquela rainha, onde $D_i = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$ onde D_i representa a rainha de i-ésima posição no indivíduo.

Além disso, D_{total} é representado por:

$$D_{total} = \sum_{i=1}^{r} \bigcup D_i \tag{1}$$

4.1 Cálculo da Dominação

Para calcular a dominação de uma rainha nós separamos nas seguintes equações:

$$\sum_{i=0}^{n} \bigcup (x, i) \tag{2}$$

$$\sum_{i=0}^{n} \bigcup (i, y) \tag{3}$$

$$\sum_{i=1}^{min(x,y)+1} \bigcup (x-i, y-i) \tag{4}$$

$$\sum_{i=1}^{min(x,n-y-1)+1} \bigcup (x-i, y+i) \tag{5}$$

$$\sum_{i=1}^{min(n-x-1,y)+1} \bigcup (x+i, y-i) \tag{6}$$

$$\sum_{i=1}^{min(n-x,n-y)} \bigcup (x+i, y+i) \tag{7}$$

As equações (2) e (3) representam a horizontal e vertical, respectivamente, (4) e (5) representam respectivamente as diagonais superiores esquerda e direita e (6) e (7) as diagonais inferiores esquerda e direita.

E o D de uma rainha é a união de todas essas equações para ela.

5 Metodologia

A nossa metodologia foi utilizar um *ILS* basico. Ele gera um início randômico, passa ele para o *Local Search*, salva o indivíduo se o $fitness$ dele for o melhor. Em seguida ele pega esse ultimo indivíduo gerado pelo *Local Search* e o modifica usando nossa função de *Perturbação* para em seguida enviar esse novo indivíduo para o *Local Search* fazendo um loop. Este exemplo pode ser visto no Pseudocódigo 1.

5.1 $Fitness$

Nosso $fitness$ é baseado no $fitness$ do *paper* de algoritimo genético (ALHARBI; VENKAT, 2017).

Dado o conjunto D_{total} que representa a dominação das rainhas, e n representa a quantidade de casas em uma coordenada do tabuleiro temos que:

$$fitness = \frac{|D_{total}|}{n^2} \tag{8}$$

Temos as seguintes propriedades:

- Quanto maior o $fitness$, melhor o indivíduo.
- Se o $fitness$ for igual à 1, aquela solução é ótima.

5.2 Início Randômico

O *ILS* começa gerando um inicio randômico, onde ele bota r rainhas no tabuleiro onde r é uma argumento do programa. A única restrição deste inicio aléatorio é que duas rainhas não podem ocupar o mesmo quadrado e cada rainha deve estar dentro dos limites do tabuleiro.

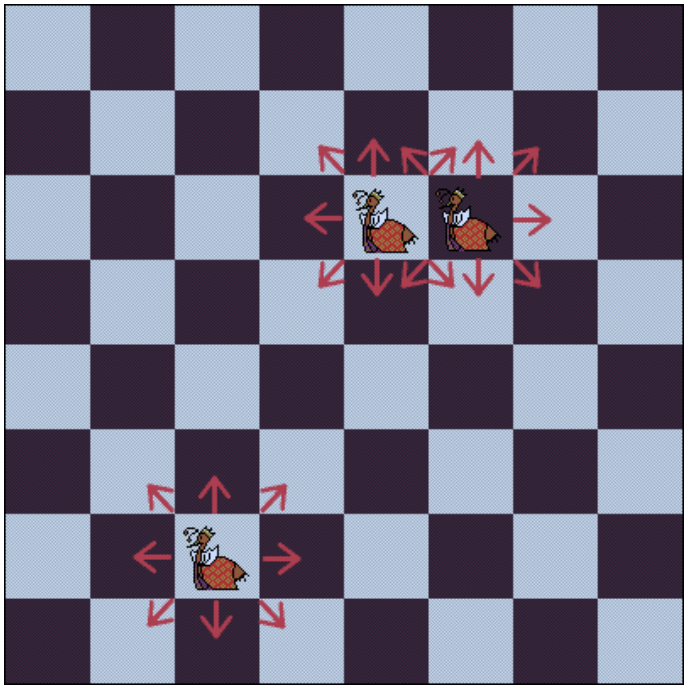

Figura 3 – Exemplo de possiveis movimentações de três rainhas no local search e na perturbação

5.3 *Local Search*

O *Local Search* calcula a movimentação de cada rainha para cada direção com 1 de distância (direita, esquerda, cima, baixo e diagonais) [Vide Figura 3] Essa movimentação deve atendender as mesmas restrições do início randômico. 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.

5.4 Perturbação

Para fazer a perturbação a gente realiza um número p de pertubações, onde p é um argumento inicial do programa. Para cada pertubação, escolhe-se uma rainha e realiza-se uma movimentação randômica de uma casa, caindo nas restrições citadas no início randômico.

6 Resultados

Nossos testes foram rodados em uma máquina Intel Core i5-7400 com 8GB de RAM, usando o sistema operacional Windows 10. A linguagem de programação utilizada foi Python 3.7.0 32-bit.

7 Conclusão

Text...

Algorithm 1: Pseudocódigo do algoritmo de ILS utilizado

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

Algorithm 2: Pseudocódigo do Algoritmo Genético utilizado

```
Data: population, best
Result: best - Melhor fitness na população
population ← randomStart();
populationFitness ← fitness(population);
repeat
    population ← algoritmoGenetico(population);
    populationFitness ← fitness(population);
until maxIterações OU ∃populationFitness = 1;
best ← bestIn(population, populationFitness);
```

Tabela 2 – Resultados conseguidos em um tabuleiro 14×14 com 7 rainhas.

	Melhor *fitness* (%)	Tempo de execução (s)
Algoritmo Genético	91.3265306122448	23.75254201889038
ILS (perturbação 5)	91.8367346938775	20.71260714530945
ILS (pertubação 10)	91.3265306122448	23.75254201889038

Referências

ALHARBI, S.; VENKAT, I. A genetic algorithm based approach for solving the minimum dominating set of queens problem. *Journal of Optimization*, Hindawi, v. 2017, 2017. Citado 4 vezes nas páginas 2, 3, 4 e 5.

BELL, J.; STEVENS, B. A survey of known results and research areas for n-queens. *Discrete Mathematics*, Elsevier, v. 309, n. 1, p. 1–31, 2009. Citado na página 3.

BURGER, A. P.; MYNHARDT, C. M. An upper bound for the minimum number of queens covering the n× n chessboard. *Discrete Applied Mathematics*, Elsevier, v. 121, n. 1-3, p. 51–60, 2002. Citado na página 4.

Tabela 3 – Resultados conseguidos em um tabuleiro 14×14 com 8 rainhas.

	Melhor *fitness* (%)	Tempo de execução (s)
Algoritmo Genético	95.4081632653061	115.54515361785889
ILS (perturbação 5)	95.9183673469387	30.753767013549805
ILS (pertubação 10)	95.4081632653061	29.001476049423218

Tabela 4 – Resultados conseguidos em um tabuleiro 18×18 com 9 rainhas.

	Melhor *fitness* (%)	Tempo de execução (s)
Algoritmo Genético	92.9012345679012	251.10087895393372
ILS (perturbação 5)	90.7407407407407	59.24564051628113
ILS (pertubação 10)	90.4320987654321	67.17142534255981

COCKAYNE, E. J. Chessboard domination problems. *Discrete Mathematics*, Elsevier, v. 86, n. 1-3, p. 13–20, 1990. Citado na página 4.

DOERR, B. et al. Evolutionary algorithms and dynamic programming. *Theoretical computer science*, Elsevier, v. 412, n. 43, p. 6020–6035, 2011. Citado na página 2.

GIBBONS, P. B.; WEBB, J. Some new results for the queens domination problem. *Australasian Journal of Combinatorics*, CENTRE FOR COMBINATORICS, v. 15, p. 145–160, 1997. Citado na página 4.

HOOS, H. H.; STÜTZLE, T. *Stochastic local search: Foundations and applications*. [S.l.]: Elsevier, 2004. Citado na página 2.

LOURENÇO, H. R.; MARTIN, O. C.; STÜTZLE, T. Iterated local search: Framework and applications. In: *Handbook of metaheuristics*. [S.l.]: Springer, 2010. p. 363–397. Citado na página 2.

WEAKLEY, W. D. Upper bounds for domination numbers of the queen's graph. *Discrete mathematics*, North-Holland, v. 242, n. 1-3, p. 229–243, 2002. Citado na página 4.