Análise Comparativa de Abordagens para o Problema das N Rainhas: Backtracking, Algoritmos Genéticos e Otimização por Enxame de Partículas

Guilherme Henrique dos Santos Noronha ;
Universidade La Salle ;
guilherme.202110448@unilasalle.edu.br .

Resumo

Este artigo apresenta uma análise comparativa de três abordagens distintas para a resolução do Problema das N Rainhas, um desafio clássico no campo da Inteligência Artificial. Foram implementadas e avaliadas soluções baseadas em Backtracking, Algoritmos Genéticos (GA) e Otimização por Enxame de Partículas (PSO). O objetivo principal é investigar o desempenho de cada método, considerando aspectos como custo computacional, uso de memória e tempo de execução para encontrar uma ou todas as 92 soluções válidas para $N = 8$. Detalhes sobre a adaptação de GA e PSO para um problema discreto e as dificuldades encontradas durante a implementação e validação são discutidos. Os resultados demonstram a superioridade do Backtracking para problemas de busca exaustiva em espaços de busca limitados, enquanto GA e PSO se revelam ferramentas versáteis para otimização em cenários mais complexos e estocásticos.

Problema das N Rainhas; Backtracking; Algoritmos Genéticos; Otimização por Enxame de Partículas; Inteligência Artificial; Busca; Otimização.

1 Introdução

O Problema das N Rainhas é um desafio combinatorial que consiste em posicionar N rainhas em um tabuleiro de xadrez $N \times N$ de forma que nenhuma rainha ataque outra. Isso significa que nenhuma rainha pode compartilhar a mesma linha, coluna ou diagonal com qualquer outra rainha. Este problema é um pilar no estudo de algoritmos de busca e resolução de problemas em Inteligência Artificial (IA), servindo como um excelente exemplo para ilustrar conceitos de espaço de estados, satisfação de restrições e eficiência algorítmica [1]. Para $N = 8$, o problema das 8 rainhas possui 92 soluções únicas.

A relevância do Problema das N Rainhas na IA transcende sua simplicidade aparente. Ele é frequentemente utilizado como um benchmark para avaliar a eficácia de diferentes paradigmas de algoritmos, desde métodos de busca exaustiva até abordagens heurísticas e meta-heurísticas. Este artigo tem como objetivo principal explorar e comparar o desempenho de três metodologias distintas na resolução deste problema: Backtracking, Algoritmos Genéticos (GA) e Otimização por Enxame de Partículas (PSO).

Serão analisados aspectos críticos como custo computacional teórico, o consumo de memória durante a execução e, de forma empírica, o tempo de execução necessário para encontrar uma solução válida, bem como o tempo para identificar todas as 92 soluções existentes para $N = 8$. Adicionalmente, serão detalhadas as principais dificuldades encontradas durante o processo de implementação e as estratégias adotadas para superá-las, especialmente no que tange à adaptação de GA e PSO para um problema de natureza discreta.

2 Fundamentação Teórica

As abordagens exploradas neste trabalho representam diferentes paradigmas de resolução de problemas em IA.

2.1 Backtracking

O Backtracking é uma técnica algorítmica recursiva para resolver problemas combinatórios. Ele constrói soluções passo a passo, e a cada passo, verifica se a solução parcial é válida. Se uma escolha leva a uma solução inválida ou a um beco sem saída, o algoritmo "volta atrás"(backtracks) para o passo anterior e tenta uma alternativa diferente. Para o Problema das N Rainhas, o Backtracking funciona colocando uma rainha em cada coluna, uma por vez, verificando a cada passo se a nova rainha está em uma posição segura (não atacada por rainhas anteriores). Se uma linha na coluna atual não permite uma colocação segura, o algoritmo retrocede para a coluna anterior para alterar a posição da rainha previamente colocada [2]. Este método garante a descoberta de todas as soluções possíveis.

2.2 Algoritmos Genéticos (GA)

Algoritmos Genéticos são uma classe de algoritmos de otimização inspirados no processo de seleção natural e genética. Eles operam sobre uma população de soluções candidatas (indivíduos), que evoluem ao longo de gerações. Cada indivíduo é uma "cromossomo"que representa uma possível solução para o problema. O processo básico de um GA envolve:

1. **População Inicial:** Geração de um conjunto aleatório de indivíduos.
2. **Função de Fitness:** Avalia a "qualidade"de cada indivíduo na população. Indivíduos mais aptos têm maior chance de sobreviver e se reproduzir.
3. **Seleção:** Escolha de indivíduos (pais) para a reprodução, baseada em seu fitness.
4. **Crossover (Cruzamento):** Combinação de material genético de dois pais para criar novos indivíduos (filhos).
5. **Mutação:** Pequenas alterações aleatórias nos genes dos filhos para introduzir diversidade e evitar convergência precoce a ótimos locais.
6. **Substituição:** A nova geração de indivíduos substitui a antiga.

Para o Problema das N Rainhas, a representação de um indivíduo pode ser um array onde o índice é a coluna e o valor é a linha da rainha, garantindo que não haja duas rainhas na mesma coluna. O fitness é determinado pelo número de pares de rainhas que não se atacam [3].

2.3 Otimização por Enxame de Partículas (PSO)

A Otimização por Enxame de Partículas (PSO) é uma meta-heurística de otimização inspirada no comportamento social de bandos de pássaros ou cardumes de peixes. No PSO, cada solução candidata é uma "partícula"que se move no espaço de busca. O movimento de cada partícula é influenciado por sua melhor posição histórica (melhor pessoal) e pela melhor posição encontrada por qualquer partícula no enxame (melhor global). A atualização da velocidade e posição de uma partícula é dada por:

$$v_i(t+1) = wv_i(t) + c_1 r_1(pbest_i - x_i(t)) + c_2 r_2(gbest - x_i(t))$$

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

Onde v_i é a velocidade da partícula i, x_i é sua posição, w é o fator de inércia, c_1 e c_2 são os coeficientes cognitivo e social, e r_1 e r_2 são números aleatórios entre 0 e 1. $pbest_i$ é a melhor posição pessoal e $gbest$ é a melhor posição global [4].

A aplicação do PSO ao Problema das N Rainhas exige uma adaptação, pois o problema é discreto. A interpretação da posição e velocidade de uma partícula deve considerar que as "dimensões"são as linhas das rainhas, que devem ser inteiras e dentro dos limites do tabuleiro.

3 Metodologia

A metodologia deste trabalho envolveu a implementação de três algoritmos em Python 3.x, a definição de um conjunto de testes e a coleta de métricas de desempenho para cada abordagem.

3.1 Implementação

Todas as soluções foram implementadas em Python. A representação do tabuleiro para o Problema das N Rainhas foi padronizada como uma lista de inteiros, onde 'board[col]' indica a linha da rainha na coluna 'col'. Essa representação garante intrinsecamente que não há duas rainhas na mesma coluna. A função 'calculate$_attacks$'e'$get_fitness$'($utilizadaporGAePSO$)$foramimplementadasparadeterminaravalidade

Para o Backtracking, uma função recursiva foi utilizada para explorar o espaço de busca. Para GA e PSO, a função de fitness foi definida como 1.0/(1 + número de ataques), garantindo que soluções perfeitas (0 ataques) tivessem o fitness máximo de 1.0.

3.2 Configuração dos Experimentos

Os experimentos foram conduzidos para o Problema das 8 Rainhas ($N = 8$).

- **Backtracking:** Não requer parâmetros de ajuste além de N.
- **Algoritmo Genético (GA):**
  - Tamanho da População: 300 indivíduos
  - Número de Gerações: 3000
  - Taxa de Mutação: 0.1 (10%)
  - Elitismo: 10% da população
- **Otimização por Enxame de Partículas (PSO):**
  - Número de Partículas: 100
  - Máximo de Iterações: 1500
  - Coeficientes Cognitivo (c_1) e Social (c_2): 2.0
  - Fator de Inércia (w): Decresce linearmente de 0.9 (w_start) para 0.4 (w_end)

GA e PSO, sendo algoritmos estocásticos, foram executados 10 vezes cada para mitigar a variabilidade aleatória e obter médias de desempenho mais representativas.

3.3 Métricas de Avaliação

As seguintes métricas foram coletadas e analisadas para cada abordagem:

1. **Custo Computacional (Teórico):** Analisado com base na complexidade assintótica dos algoritmos ($O(N!), O(G \cdot P \cdot N^2)$, etc.).
2. **Custo em Memória:** Medido pelo aumento do consumo de RAM (Resident Set Size - RSS) do processo Python durante a execução de cada função, utilizando a biblioteca 'psutil'. O valor reportado é a diferença entre a memória após a execução e antes da execução, em MegaBytes (MB).
3. **Tempo de Execução para encontrar uma solução válida:** Medido em segundos utilizando 'time.perf$_counter$()'$desdeoiníciodaexecuçãodafunçãoatéomomentoemqueumasoluçãoperfeita$($fitness1.0paraGA/PS$ sucedidas.$**Tempo de Execução para achar as 92 soluções existentes:**$Aplicávelprimariamenteao Backtracking$

4 Resultados e Discussão

Os experimentos foram executados em um ambiente de desenvolvimento padrão, e os resultados obtidos para $N = 8$ são apresentados a seguir.

4.1 Resultados Empíricos

Figura 1: Exemplo de Solução para 8 Rainhas encontrada via Backtracking.

Figura 2: Exemplo de Solução para 8 Rainhas encontrada via Algoritmo Genético.

Figura 3: Exemplo de Solução para 8 Rainhas encontrada via Otimização por Enxame de Partículas.

Tabela 1: Comparativo de Desempenho para o Problema das 8 Rainhas

Métrica	**Backtracking**	**Algoritmo Genético (GA)**	**PSO**
Tempo para 1 Solução (s)	**0.0001 - 0.0003** (aprox.)	0.1 - 2.0 (méd.)	0.5 - 5.0 (méd.)
Tempo para 92 Soluções (s)	**0.0005 - 0.0010** (aprox.)	N/A	N/A
Memória RAM (MB)	**0.05 - 0.15** (aprox.)	1.0 - 5.0 (méd.)	1.0 - 5.0 (méd.)
Garantia de Solução Ótima	Sim (exata)	Não (heurística)	Não (heurística)
Garantia de Todas as Soluções	Sim	Não	Não

Os valores na Tabela 1 são aproximados e dependem do hardware e da carga do sistema. Os tempos para GA e PSO são médias das execuções bem-sucedidas.

4.1.1 Análise de Custo Computacional e Tempo de Execução

O **Backtracking** demonstrou ser o método mais eficiente para o Problema das 8 Rainhas. Seu tempo de execução para encontrar a primeira solução é praticamente instantâneo (na ordem de microssegundos ou poucos milissegundos), e ele também é capaz de encontrar todas as 92 soluções em um tempo extremamente curto (milissegundos). Isso ocorre devido à natureza do problema, onde as podas na árvore de busca (verificações de segurança) são muito eficazes, limitando o espaço de busca efetivamente explorado.

Em contraste, o **Algoritmo Genético (GA)** e a **Otimização por Enxame de Partículas (PSO)** apresentaram tempos de execução significativamente maiores para encontrar uma única solução. Como são algoritmos estocásticos, seus tempos variam e exigem múltiplas execuções para uma análise representativa. Embora consigam encontrar uma solução válida, eles não são tão determinísticos ou rápidos quanto o Backtracking para este problema específico. Suas complexidades teóricas mais elevadas ($O(\text{Gerações} \times \text{TamanhoPopulação} \times N^2)$ e $O(\text{IteraçõesMáximas} \times \text{NúmeroDePartículas} \times N^2)$ respectivamente) refletem que, para $N = 8$, a constante multiplicativa é maior, tornando-os menos eficientes que o $N!$ podado do Backtracking.

Nenhum dos métodos heurísticos (GA e PSO) é adequado para encontrar todas as 92 soluções existentes, pois sua finalidade é convergir para uma ou algumas soluções ótimas ou próximas do ótimo, não para enumerar todo o espaço de soluções.

4.1.2 Análise de Custo em Memória

O **Backtracking** exibiu o menor consumo de memória, utilizando apenas alguns kilobytes para a pilha de recursão e para armazenar as 92 soluções. Sua abordagem construtiva e de retorno de chamadas minimiza a necessidade de grandes estruturas de dados.

GA e PSO, por outro lado, exigem o armazenamento de uma **população** ou **enxame** de indivíduos/partículas. Cada indivíduo/partícula representa uma solução completa e consome $O(N)$ de memória. Assim, o custo total de memória para GA e PSO é $O(\text{TamanhoPopulação/NúmeroDePartículas} \times N)$. Para os parâmetros utilizados, isso resultou em um uso de memória notavelmente maior do que o Backtracking, embora ainda gerenciável para $N = 8$.

4.2 Dificuldades Encontradas e Soluções

A implementação das abordagens apresentou desafios específicos, principalmente na adaptação das meta-heurísticas para o domínio discreto do Problema das N Rainhas.

4.2.1 Adaptação de GA e PSO para Problemas Discretos

4. **Dificuldade:** Algoritmos como GA e PSO são inerentemente projetados para otimização em espaços de busca contínuos. O Problema das N Rainhas, no entanto, opera em um espaço discreto (linhas inteiras no tabuleiro). As operações padrão de atualização de posição (PSO) ou combinação genética (GA) poderiam gerar valores não inteiros ou fora dos limites do tabuleiro.

- **Solução:** A representação do tabuleiro como um array de linhas (‘board[col] = row‘) já assegurou que as rainhas estivessem em colunas distintas. Para o GA, as operações de crossover e mutação foram implementadas para garantir que os valores das linhas permanecessem inteiros dentro do intervalo $[0, N-1]$. Para o PSO, as posições calculadas (que poderiam ser flutuantes) foram **arredondadas** para o inteiro mais próximo e, em seguida, **clipadas** (limitadas) para permanecerem dentro dos limites válidos do tabuleiro $[0, N-1]$. Essa discretização foi crucial para a aplicação bem-sucedida do PSO.

4.2.2 Definição da Função de Fitness/Objetivo

- **Dificuldade:** Uma função de fitness eficaz é fundamental para guiar a busca de GA e PSO. Inicialmente, a contagem de ataques diretos parecia óbvia, mas converter essa contagem em um "fitness"otimizável era a chave.

- **Solução:** A função ‘get$_f itness(individual) = 1.0/(1+attacks)‘ foi adotada. Essa formula$ção$ garante que o fitness sejas$

4.2.3 Ajuste de Parâmetros das Heurísticas

- **Dificuldade:** O desempenho de GA e PSO é altamente sensível aos seus parâmetros (tamanho da população, gerações, taxas de mutação, coeficientes de inércia, etc.). Valores inadequados podem levar a uma convergência lenta, estagnação em ótimos locais ou falha na descoberta de soluções.

- **Solução:** Um processo iterativo de experimentação e ajuste fino dos parâmetros foi realizado. Iniciou-se com valores comuns da literatura e foram ajustados observando o tempo de convergência e a frequência de sucesso na obtenção de uma solução ótima para $N = 8$. Para o GA, a inclusão de **elitismo** (preservar os melhores indivíduos) foi fundamental para evitar a perda de boas soluções entre as gerações. Para o PSO, a estratégia de **inércia dinâmica** (decaimento de ‘w‘ ao longo das iterações) foi empregada para equilibrar a exploração e a explotação do espaço de busca.

4.2.4 Medição Precisa de Desempenho

- **Dificuldade:** A medição de tempo e memória em ambientes de execução de alto nível como Python pode ser suscetível a variações devido a processos em segundo plano, gerenciamento de memória do interpretador (coleta de lixo) e a natureza estocástica dos algoritmos heurísticos.

- **Solução:** Para o tempo, ‘time.perf$_c ounter()‘ foi usado para a alta precisão. Para a memória, a biblioteca ‘psutil‘ permitium$

5 Conclusão

Este estudo comparou três abordagens distintas para a resolução do Problema das 8 Rainhas: Backtracking, Algoritmo Genético (GA) e Otimização por Enxame de Partículas (PSO). Os resultados empíricos e a análise teórica demonstram que, para este problema específico, o **Backtracking** emerge como a metodologia mais eficiente e confiável. Ele é capaz de encontrar todas as 92 soluções em um tempo insignificante e com um consumo de memória mínimo, devido à sua natureza exaustiva e às eficazes podas que reduzem o espaço de busca.

Por outro lado, enquanto GA e PSO foram capazes de encontrar soluções válidas para o Problema das 8 Rainhas, seu desempenho foi inferior ao do Backtracking em termos de tempo de execução e eficiência de memória. Isso se deve à sua natureza estocástica e à complexidade inerente de suas operações, que são mais adequadas para problemas de otimização em espaços de busca muito maiores e mais complexos, onde uma busca exaustiva seria inviável. As adaptações para o domínio discreto, embora bem-sucedidas, adicionaram uma camada de complexidade na implementação.

Em suma, para problemas de busca e satisfação de restrições em espaços de busca gerenciáveis e com a necessidade de encontrar todas as soluções, o Backtracking permanece a escolha ideal. Para problemas de otimização com grandes espaços de busca e onde soluções "boas o suficiente"são aceitáveis, as meta-heurísticas como GA e PSO oferecem uma alternativa poderosa e flexível. Trabalhos futuros poderiam explorar o desempenho dessas abordagens para valores de N significativamente maiores, onde a complexidade do Backtracking se tornaria proibitiva, evidenciando ainda mais a relevância das meta-heurísticas.

Referências

[1] Russell, Stuart J.; Norvig, Peter. Inteligência Artificial: Uma Abordagem Moderna. 3ª Edição. Pearson Education, 2010.

[2] Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford. Algoritmos: Teoria e Prática. 3ª Edição. LTC, 2012.

[3] Goldberg, David E. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, 1989.

[4] Kennedy, James; Eberhart, Russell. Particle Swarm Optimization. In: Proceedings of IEEE International Conference on Neural Networks, Vol. IV, pp. 1942-1948. IEEE, 1995.