TP1 - O problema das N-Rainhas e Função Rastrigin

Guilherme Castro Silva

***Abstract*— Neste trabalho é apresentado como Algoritmos Genéticos podem ser utilizados para resolver problemas de permutação como o das N-Rainhas e também aplicados na minimização de funções multi-modais como a Rastrigin. Ademais, é mostrado as estratégias adotadas para representação, função de aptidão, operadores de variação, mecanismos de seleção, modelos de população e condições de parada para ambos problemas afim de encontrar boas soluções.**

I. INTRODUÇÃO

Algoritmos Genéticos (AG) são uma classe particular de algoritmos evolutivos que utilizam conceitos provenientes da biologia evolutiva como: hereditariedade, mutação, seleção natural e recombinação para abordar problemas, em especial de otimização.

De forma geral, são implementados de forma que uma população de representações abstratas de solução é selecionada em busca de soluções melhores. A evolução geralmente se inicia a partir de um conjunto de soluções criado aleatoriamente e é realizada por meio de gerações. A cada geração, a adaptação de cada solução na população é avaliada, alguns indivíduos são selecionados para a próxima geração, e recombinados ou mutados para formar uma nova população.

Como operam sobre uma população de candidatos em paralelo, a busca é realizada em diferentes áreas do espaço de solução ao mesmo tempo, possibilitando uma busca mais diversa sobre o espaço. Esse classe de algoritmos é robusta, genérica e facilmente adaptável, além disso, tem como vantagem a capacidade de encontrar boas soluções para problemas que não podem ser resolvidos em tempo polinomial.

A seguir será descrito as estratégias utilizadas para resolver o problema das N-Rainhas e também para encontrar soluções para a Função *Rastrigin* utilizando Algoritmos Genéticos.

II. N-RAINHAS

Dado um tabuleiro de xadrez regular (NxN) e N rainhas, o problema das N-rainhas consiste em posicioná-las no tabuleiro de forma que nenhuma rainha seja capaz de capturar outra rainha. A Figura 1 ilustra a área de captura de uma rainha posicionada no tabuleiro.

O problema pode ser formulado como um problema de Satisfação de Restrições, no qual o objetivo é encontrar uma configuração das rainhas sobre o tabuleiro que não viole nenhuma restrição.

Para resolver o problema das N-rainhas utilizando Algoritmos Genéticos (AG) deve-se, primeiramente, definir:

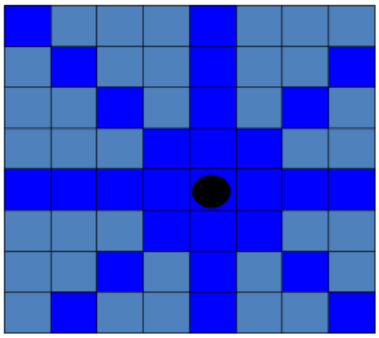

Fig. 1. Ilustração do problema das N-Rainhas com um tabuleiro 8x8.

- uma representação (codificação) para os indivíduos (i.e. solução candidata);
- uma função de aptidão (objetivo) para avaliar a qualidade de uma solução candidata;
- operadores de variação;
- mecanismos de seleção;
- modelo de população;
- uma condição de término, por exemplo, número de gerações ou um valor de fitness.

A. Representação

Fenótipo (f): uma dada configuração do tabuleiro com N rainhas;

Genótipo (g) ou cromossomo: permutação de inteiros 1, 2, 3, ..., N. Isto é, g = i_1, i_2, ..., i_N denotando uma configuração onde a k-ésima coluna (posição do vetor) contém uma rainha posicionada na i_k-ésima linha, conforme ilustrado a baixo:

Exemplo de codificação: [4, 8, 6, 3, 2, 5, 7, 1].

Vale ressaltar que ao usarmos esta representação genotípica para uma solução candidata, o espaço de busca fica restrito às configurações do tabuleiro onde violações horizontais (duas rainhas na mesma linha) e verticais (duas rainhas na mesma coluna) não ocorrem.

B. Função Objetivo

A qualidade de uma solução candidata q(f) = número de xeques entre pares de rainhas.

Dado que a representação utilizada garante que não há xeques na horizontal e vertical, a função objetivo é calculada somente entre o número de colisões que ocorre entre as rainhas nas diagonais do tabuleiro.

Para o tipo de representação adotado (permutação de N inteiros), a quantidade máxima de xeques que podem ocorrer é q(f) = N(N - 1)/2 e a quantidade míinima é q(f) = 0.

C. Operadores de Variação

O cruzamento utilizado para gerar novas soluções, com permutações válidas, foi o Cut-and-crossfill. No pseudo-código a baixo é descrito como foi gerado dois filhos com permutações geradas a partir de dois pais.

Algorithm 1: "Cut-and-crossfill"Crossover

Result: [Filho1, Filho2]

1. Selecione uma posição aleatória, o ponto de corte, i $\in 1, 2, 3, .., 7$;
2. Corte os pais em dois segmentos após o ponto de corte;
3. Copie o primeiro segmento do pai 1 para o filho 1 e o primeiro segmento do pai 2 para o filho 2;
4. Busque no pai 2, da esquerda para a direita, e preencha o filho 1 com os valores do pai 2 pulando os valores já contidos no filho 1;
5. Repita o mesmo processo para o pai 1 e filho 2.

Adotou-se o método de Inversão (*Swap*) como operador de mutação. Consiste em trocar dois números aleatórios do vetor (conservando assim os números presentes, porém alterando sua ordem). Portando, esse operador não possibilita a geração de uma solução inválida. A imagem a seguir ilustra a aplicação deste operador sobre um *array*.

Fig. 2. Exemplo do operador de Swap aplicado a um array.

D. Mecanismo de Seleção

O método de seleção proposto foi o Torneio, onde um número n de indivíduos da população é escolhido aleatoriamente para formar uma sub-população temporária. Deste grupo, os melhores indivíduos são selecionados. No implementação escolhida, definiu-se n = 5, ou seja cinco indivíduos foram escolhidos aleatoriamente na população e selecionado os dois melhores para efetuar *crossover*.

E. Modelo de População

O modelo de população adotado para este problema foi o Steady-State: apenas uma parte dos pais é substituída pelos descendentes.

Os indivíduos são ordenados (população original mais 2 descendentes) e os dois piores são eliminados a cada geração.

F. Condição de Parada

A execução é finalizada quando a solução ótima é encontrada, i.e. q(f) = 0, ou quando o número de gerações atinge o valor 5.000.

G. Testes

O algoritmo foi testado utilizando os seguintes parâmetros para a população:

Número de Indivíduos por geração: 100.

Probabilidade de Cruzamento (P_c): 100%.

Probabilidade de Mutação (P_m): 10%, 20%, ..., 60%.

Segue a baixo uma tabela comparativa entre o número médio de iterações utilizado pelo algoritmo para convergir para uma solução ótima ao variar a probabilidade de mutação. Os valores de iteração iguais a zero, isto é soluções ótimas geradas de forma aleatória, foram descartados para poder avaliar o comportamento do algoritmo ao longo das gerações.

TABLE I
NÚMERO DE EXECUÇÕES MÉDIA POR PROBABILIDADE DE MUTAÇÃO

Execuções	**Probabilidade de Mutação**					
	10%	**20%**	**30%**	**40%**	**50%**	**60%**
1	100	29	68	20	13	24
2	107	18	52	60	82	31
3	109	11	3	277	109	20
4	26	167	8	39	83	84
5	20	80	112	75	118	56
6	16	164	21	14	27	27
7	76	42	83	31	37	44
8	182	10	212	6	66	1
9	99	83	81	40	10	3
10	86	137	66	36	82	75
Média	82,1	74,1	70,6	59,8	62,7	36,5

De acordo com a Tabela 1, percebe-se que a medida que a probabilidade de mutação aumentava o algoritmo convergia de forma mais rápida. Isso pode ser explicado pelo fato de a alta probabilidade de mutação possibilitar uma melhor exploração do espaço de soluções para o dado problema. Na Figura 3 é mostrado um exemplo onde o algoritmo obteve sucesso para encontrar a solução para o problema na sexagésima geração. As curvas mostram exatamente o comportamento esperado pelo algoritmo, que a população foi evoluindo ao longo das gerações até encontrar a solução para o problema.

Por outro lado, as versões do algoritmo que possuíam valores de P_m entre 10% e 30% algumas vezes não conseguiam encontrar a solução para o problema, pois sua população convergia para uma população totalmente homogênea (todos indivíduos iguais) dada a baixa variabilidade, conforme mostrado na Figura 4.

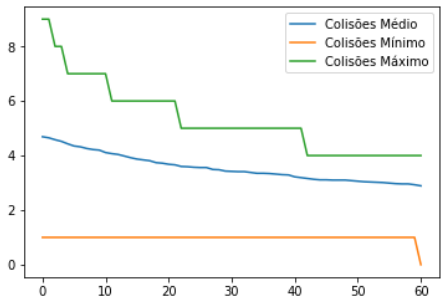

Fig. 3. Exemplo de evolução do algoritmo ao longo das gerações.

Na Figura 3 a condição de parada utilizada pelo algoritmo foi que um indivíduo encontrou a solução para o problema (Colisões igual a zero). já na Figura 4 a população não

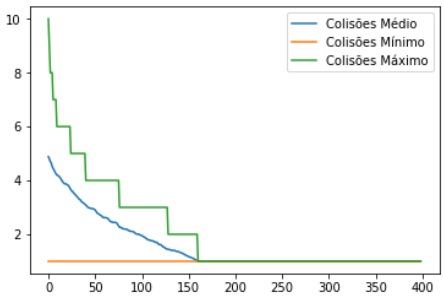

Fig. 4. Exemplo de convergência do algoritmo para população homogênea.

encontrou a solução ao longo de 5000 gerações, portanto a condição de parada acionada foi o número máximo de gerações.

III. Rastrigin

A função Rastrigin é uma função não convexa utilizada para avaliar o desempenho de algoritmos de otimização. Ela foi proposta inicialmente por Rastrigin [1] como uma função bidimensional e posteriormente generalizada por Muhlenbein et al. [2]. Esta função tem sua importância devido ao grande número de mínimos que apresenta.

$$minf(x) = 10n + \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i))) \quad (1)$$

com $x_i \in$ [-5.12, 5.12], onde n é a dimensão do espaço de busca onde a função é definida (correspondente ao número de variáveis do indivíduo). O mínimo global ocorre em x^* = 0, onde f(x^*) = 0.

Com a utilização de um método determinístico, seria praticamente impossível otimizar a função Rastrigin. Isto acontece devido à natureza multimodal da função. Para tentar encontrar o mínimo da função, para n=10 variáveis, utilizando Algoritmos Genéticos (AG), iremos novamente definir os seguintes tópicos:

- uma representação (codificação) para os indivíduos (i.e. solução candidata);
- uma função de aptidão (objetivo) para avaliar a qualidade de uma solução candidata;
- operadores de variação;
- mecanismos de seleção;
- modelo de população;
- uma condição de término.

A. Representação

Fenótipo (f): dez variáveis reais para entrada na função Rastrigin;

Genótipo (g) ou cromossomo: array binário de 100 posições (N = 100). Isto é, g = i_1, i_2, ..., i_N onde a cada dez números binários consecutivos denotam o valor de uma variável real, conforme ilustrado a baixo:

Exemplo de codificação: [1, 1, 0, 1, ... , 0, 1], array de tamanho 100. De acordo com essa representação, o valor da precisão corresponde a:

$$Precisao = (LimSup - LimMin)/(2^{NBits} - 1) \quad (2)$$

$$Precisao = (5.12 - (-5.12))/1023 \approx 0.01 \quad (3)$$

B. Função Objetivo

A qualidade de uma solução candidata q(f) = avaliação das 10 variáveis reais na função Rastrigin. Porém, como um dos operadores de seleção adotados será a Roleta, transformaremos a função objetivo para maximização.

$$maxf(x) = 1/(Rastrigin + e) \quad (4)$$

onde 'e' é um valor muito baixo (0.01) utilizado para não causar descontinuidade na função.

C. Operadores de Variação

O cruzamento utilizado para gerar novas soluções, com variação em todas as variáveis, foi o crossover com 1 ponto de corte por variável. No pseudo-código a baixo é descrito como foi gerado dois filhos a partir de dois pais.

Algorithm 2: Crossover com 1 ponto de corte por variável

Result: [Filho1, Filho2]

1. Selecione N posições aleatórias, os pontos de corte por variável, i $\in 1, 2, 3, .., 10$;
2. Corte os pais em dois segmentos após o ponto de corte;
3. Por variável faça:

3.1 Estabeleça cortes nos cromossomos pais no ponto sorteado para a variável em questão;

3.2 Gere novas variáveis filhas a partir da troca das caudas dos pais;

Adotou-se o método *Bit-flip* como operador de mutação. Consiste em alterar cada gene ($0 \rightarrow 1$; $1 \rightarrow 0$) com uma probabilidade pM. Para um cromossomo de tamanho N, na média (N x pM) bits são alterados. Portando, esse operador mantém a essência do array binário o que não possibilita a geração de uma solução inválida. A imagem a seguir ilustra a aplicação deste operador sobre um *array*.

Fig. 5. Exemplo do operador de Bit-flip aplicado a um array.

D. Mecanismo de Seleção

Para a seleção dos indivíduos foram selecionados duas técnicas com probabilidade de ocorrência de 50% para ambas, sendo elas a Roleta e o Torneio.

O primeiro método de seleção, o da Roleta, funciona da seguinte forma: indivíduos são mapeados para segmentos de reta contíguos no intervalo entre 0 e 1, baseado em sua Probabilidade de Seleção (PS) obtida pela seguinte equação:

$$PS_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \quad (5)$$

Então é sorteado um valor i tal que pertence ao intervalo [0; 1] e seleciona-se o indivíduo x tal que a ele corresponda a faixa do somatório onde i se localiza. Segue a Figura 2 onde é exemplificado a utilização desta técnica.

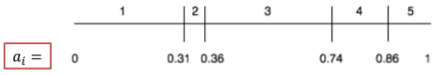

Fig. 6. Exemplo de uma roleta para uma população de 5 indivíduos, onde a linha superior indica o índice dos indivíduos e a inferior a probabilidade de seleção dada pelo intervalo.

A roleta foi girada 2 vezes para selecionar 2 indivíduos para efetuar cruzamento.

O segundo método de seleção, o Torneio, funciona da seguinte forma: um número n de indivíduos da população é escolhido aleatoriamente para formar uma sub-população temporária. Deste grupo, os melhores indivíduos são selecionados. No implementação escolhida, definiu-se n = 5, ou seja cinco indivíduos foram escolhidos aleatoriamente na população e selecionado os dois melhores para efetuar *crossover*.

E. Modelo de População

O modelo de população adotado para este problema foi o Steady-State: apenas uma parte dos pais é substituída pelos descendentes.

- a cada geração são gerados até 20 filhos;
- os filhos são introduzidos na população;
- ordena-se a população de acordo com a aptidão;
- as melhores 50 soluções são salvas para próxima geração.

F. Condição de Parada

A execução é finalizada quando a solução ótima é encontrada, i.e. q(f) = 0, ou quando o orçamento computacional é de 10.000 avaliações de função.

G. Testes

O algoritmo foi testado utilizando os seguintes parâmetros para a população:

Número de Indivíduos por geração: 50.

Probabilidade de Cruzamento (P_c): 60%.

Probabilidade de Mutação (P_m): 2%, 4%, ..., 10%.

Segue a baixo a Tabela II que mostra o comparativo entre o valor médio de função objetivo encontrado pelo algoritmo para cada configuração ao variar a probabilidade de mutação.

Com o objetivo de resolver o problema de *Hamming Cliffs*, realizou-se os mesmos testes para a codificação em código Gray. Isso foi feito, pois na codificação binária uma pequena diferença entre valores no espaço de fenótipos pode significar uma grande diferença entre estes valores no espaço dos genótipos. A grande vantagem do código Gray é que na passagem de um valor para outro sucessivo ou antecedente apenas um bit ou dígito muda, fazendo com que a estrutura de vizinhança adotada seja mais linear. Os resultados encontrados estão contidos na Tabela III.

TABLE II

NÚMERO DE EXECUÇÕES MÉDIA POR PROBABILIDADE DE MUTAÇÃO - BINÁRIO

Execuções	**Probabilidade de Mutação**				
	2%	**4%**	**6%**	**8%**	**10%**
1	5,36	6,11	19,39	17,50	25,97
2	2,17	10,61	13,24	12,81	17,74
3	8,17	5,25	9,79	15,56	26,70
4	9,14	9,49	10,49	15,10	19,34
5	8,02	6,75	12,30	16,31	31,66
6	3,37	6,13	12,14	18,11	23,67
7	8,61	4,56	11,15	12,64	32,28
8	7,21	7,53	14,89	12,19	31,33
9	3,92	4,99	3,77	18,43	31,58
10	2,33	4,33	11,73	15,31	36,58
Média	5,83	6,575	11,889	15,396	27,685

TABLE III

NÚMERO DE EXECUÇÕES MÉDIA POR PROBABILIDADE DE MUTAÇÃO - GRAY

Execuções	**Probabilidade de Mutação**				
	2%	**4%**	**6%**	**8%**	**10%**
1	3,09	5,63	10,70	10,86	15,05
2	2,08	6,13	4,91	11,01	19,30
3	3,02	1,19	8,53	10,67	15,60
4	6,00	4,62	5,92	8,53	14,38
5	8,01	2,06	12,38	10,64	19,25
6	4,02	2,67	8,97	10,51	16,91
7	7,98	12,11	13,00	8,79	13,72
8	3,25	8,07	13,06	16,20	19,54
9	7,21	6,34	6,82	15,15	22,12
10	2,85	6,41	6,67	7,56	19,78
Média	4,751	5,523	9,096	10,992	17,565

De acordo com as Tabelas 2 e 3 é possível perceber que quanto maior a probabilidade de mutação, pior é a convergência do algoritmo. Isso se deve ao fato de que a mutação estava realizando grandes alterações no genótipo e configurando piores soluções candidatas. Além disso, os resultados obtidos pela representação em Gray apresentaram uma média menor em relação a representação em Binário, mostrando na prática o problema de *Hamming Cliffs* e o código Gray se propõe uma boa alterativa para a representação da população.

Na Figura 7 é mostrado um exemplo de comportamento da população ao longo das gerações. As curvas mostram exatamente o comportamento esperado pelo algoritmo, onde a população foi evoluindo ao longo das gerações até encontrar a condição de parada.

Em ambas as configurações propostas não foram encontrados o mínimo global antes que a condição de parada fosse acionada, o que mostra a complexidade da função Rastrigin para 10 variáveis. No entanto, ao remover o critério de parada de "10.000 avaliações de função"o algoritmo desenvolvido encontrava soluções bem próximas do ótimo global.

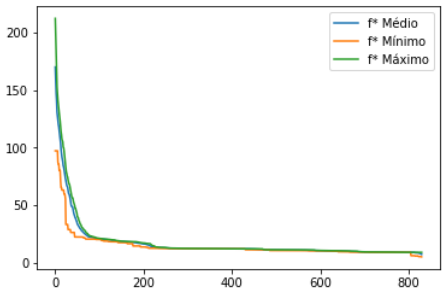

Fig. 7. Exemplo de evolução da população ao longo das gerações.

IV. CONCLUSÃO

Os Algoritmos Genéticos se propuseram como uma eficiente solução para resolver problemas de computação e também para otimização de funções multi-modais. Isso pode ser constatado devido aos valores coletados durantes as simulações que apresentaram resultados bem próximos do esperado. Contudo, essa classe de algoritmos requer uma grande atenção na forma de representação da solução, na escolha dos hiper-parâmetros e na definição de estruturas de vizinhança como: cruzamento, mutação pois estes são cruciais para o bom desempenho do algoritmo.

REFERENCES

[1] L. A. Rastrigin. Systems of extremal control. Nauka, 1974.
[2] H. Mühlenbein, D. Schomisch and J. Born. ”The Parallel Genetic Algorithm as Function Optimizer ”. Parallel Computing, 17, 1991.
[3] Agoston E. Eiben, and James E. Smith. Introduction to evolutionary computing. Vol. 53. Berlin: springer, 2003.