Proceeding Series of the Brazilian Society of Computational and Applied Mathematics, v. 10, n. 1, 2023.

Trabalho apresentado no XLII CNMAC, Universidade Federal de Mato Grosso do Sul - Bonito - MS, 2023

Proceeding Series of the Brazilian Society of Computational and Applied Mathematics

Evolução Diferencial, Algoritmo Genético e Simulated Annealing aplicados ao problema da k-dispersividade no círculo

Josimar da Silva Rocha[1]
DAMAT/UTFPR, Cornélio Procópio, PR

Resumo. Neste trabalho utilizaremos os algorimos Evolução Diferencial, Algoritmo Genético e Simulated Annealing para determinar, para cada inteiro k, de 2 até 10, uma aproximação para a constante $d(k)$ que indica a máxima dispersividade de k pontos no círculo e compararemos os métodos utilizados em termos de precisão e eficácia.

Palavras-chave. Evolução Diferencial, Simulated Annealing, Algoritmo Genético, Dispersão, Círculo.

1 Introdução

A solução de problemas de dispersão são bastante conhecidos devido a importância de dar um destino adequado a uma série de materiais, rejeitos industriais e nucleares ou mesmo distribuição inteligente de instalações nucleares como podemos ver em [3, 4]. Acontecimentos recentes como pandemias vem aumentando a necessidade também de promover ambientes que mantenham o distanciamento de pessoas ou de instalações contaminantes o que aumentam a necessidade de utilização de parâmetros que sirvam de base para promover a dispersão ou máximo de distânciamento. É neste sentido que utilizamos a métrica max-min para o problema da k-dispersividade de pontos no interior de um círculo. Esta métrica é bastante utilizada em também em grafos, quando desejamos encontrar um subgrafo com k vértices de tal forma a maximizar a menor distância entre dois vértices, também conhecido como o problema da k-dispersão discreta, conforme abordado em [1, 2, 5].

Neste trabalho, utilizaremos os Algoritmos Evolução Diferencial, Algoritmo Genético e Simulated Annealing para determinar, para cada inteiro k tal que $2 \leq k \leq 10$, uma aproximação para a constante $d(k)$ que indica a máxima dispersividade de k pontos em um círculo.

1.1 O problema da k-dispersão no círculo

Seja C_r um círculo de raio r e centrado na origem.

O problema da k-dispersão **max-min** no círculo C_r consiste em encontrarmos um subconjunto $S = \{P_1, P_2, \cdots, P_k\}$ de k pontos círculo C_r de tal forma a maximizar a função

$$f(S) = \min_{1 \leq i < j \leq k} d(P_i, P_j).$$

Outro tipo de problema da k-dispersão no círculo é o problema da k-dispersão **max-soma** no círculo C_r que consiste em encontrarmos um subconjunto $S = \{P_1, \cdots, P_k\}$ de k pontos no círculo

[1] jrocha@utfpr.edu.br

C_r de tal forma a maximizar a função

$$f(S) = \sum_{1 \leq i < j \leq k} d(P_i, P_j).$$

Em ambos os tipos de problemas da k-dispersão, as funções a serem maximizadas, que são as nossas funções objetivo, podem ser também chamadas de funções de aptidão ou funções *fitness*.

1.2 Problema da k-dispersividade no círculo

O problema da k-dispersividade (max-min) no círculo C_r consiste em encontrarmos para cada inteiro positivo k uma constante $d = \max_{P \in C_r^k} f(P)/r$, chamada de **constante de k-dispersividade**, que depende de k mas não depende do raio r do círculo, onde

$$f(P_1, P_2, \cdots, P_k) = \min_{1 \leq i < j \leq k} d(P_i, P_j),$$

para um subconjunto $S = \{P_1, \cdots, P_k\}$ de pontos no círculo C_r.

2 Evolução Diferencial (DE)

Os seguintes passos são realizados no algoritmo Evolução Diferencial, onde n é o tamanho da população e k é a quantidade de pontos:

1) Iniciar com $m = 0$ e criar uma população inicial $X^{(0)} = [X_1^{(0)}, \cdots, X_n^{(0)}]$, onde, para cada $i = 1, \cdots, n, X_i^{(0)} = [x_{i1}^{(0)}, \cdots, x_{ik}^{(0)}]$ é uma lista de k pontos do círculo C_r;

2) Criar uma população de mutantes $Y^{(m+1)} = [Y_1^{(m+1)}, Y_2^{(m+1)}, \cdots, Y_n^{(m+1)}]$ com $Y_i^{(m+1)} = [y_{i1}^{(m+1)}, \cdots, y_{ik}^{(m+1)}]$, efetuando o operador de mutação

$$Y_i^{(m+1)} = X_a^{(m)} + F(X_b^{(m)} - X_c^{(m)}),$$

onde F é uma constante real do intervalo $[0, 2]$ e a, b e c são números inteiros distintos escolhidos de forma aleatória em $\{1, 2, \cdots, n\}$ desde que satisfaçam a condição de que os pontos em $Y_i^{(m+1)}$ pertençam ao círculo C_r;

3) Criar uma população de filhos $W^{(m+1)} = [W_1^{(m+1)}, \cdots, W_n^{(m+1)}]$ onde cada filho $W_i^{(m+1)} = [w_{i1}^{(m+1)}, \cdots, w_{ik}^{(m+1)}]$ é criado a partir de uma taxa de cruzamento p_c fixada do intervalo $(0, 1)$ realizando a operação

$$w_{ij}^{(m+1)} = \begin{cases} y_{ij}^{(m+1)}, & \text{se } r_{ij} \leq p_c \\ x_{ij}^{(m)}, & \text{se } r_{ij} > p_c \end{cases},$$

sendo que, para cada inteiro $i \in \{1, \cdots, n\}$ e $j \in \{1, \cdots, k\}, r_{ij}$ é um número aleatório escolhido no intervalo $[0, 1)$;

4) Aplicar o operador de seleção de tal forma a criar indivíduos para a próxima geração, realizando a seguinte operação:

$$X_j^{(m+1)} = \begin{cases} X_j^{(m)}, & \text{se } f\left(X_j^{(m)}\right) \geq f\left(W_j^{(m+1)}\right) \\ W_j^{(m+1)}, & \text{caso contrário} \end{cases}$$

para $j = 1, \cdots, n$;

5) Faça $m = m + 1$ e verificar se o critério de parada está satisfeito. Caso o critério de parada não seja satisfeito repetir os passos a partir do passo 2;

6) Retornar $X_{Best}^{(m+1)}$ onde $Best$ é o índice do indivíduo melhor avaliado.

Uma forma simplificada do Algoritmo Evolução Diferencial é indicada no Algorimo 1.

Algoritmo 1: Evolução Diferencial (DE)

Entrada: $k, tamanho, MAX, r, F, p_C$
Saída: Melhor solução M;
início
  $Pop = CriaPopulacao(k, tamanho, r)$;
  $i = 0$;
  caso $(i < MAX)$ **faça**
    $Pop_M = Mutacao(Pop, F)$;
    $Pop_C = Cruzamento(Pop, Pop_M, p_C)$;
    $Pop = Selecao(Pop, Pop_C)$;
    $i = i + 1$;
  fim
fim

Os seguintes parâmetros foram utilizados no DE, obtidos de forma empírica por experimentos computacionais de tal forma a obter valores maiores para a constante de k-dispersividade com o menor custo computacional:

- Parâmetro k do problema da k-dispersão max-min no círculo C_r para $r = 1000$;
- Tamanho da população: $tamanho = 30$;
- Quantidade de iterações: $MAX = 1000$;
- Parâmetro de mutação: $F = 0.5$;
- Taxa de troca de genes entre dois indivíduos da população: $p_C = 0.8$;

O algoritmo foi rodado 100 vezes com os parâmetros acima.

3 Algoritmo Genético (GA)

Algoritmos genéticos são algoritmos para otimização inspirados no processo de seleção natural. Tal qual no processo de seleção natural o algoritmo cria uma população inicial que é um conjunto de indivíduos que são primeiras aproximações para a solução do problema de otimização considerado e esta população sofre transformações a cada iteração ou geração. Assim como no processo de seleção natural, onde indivíduos mais aptos geralmente sobrevivem o algoritmo genético conta com uma função *fitness* que é responsável por mensurar a aptidão dos indivíduos da população e comparar a aptidão entre indivíduos de uma mesma população, estabelecendo também quais indivíduos são mais aptos e quais indivíduos são menos aptos.

Para tanto o algoritmo conta com alguns operadores também inspirados no processo de seleção natural para que os indivíduos da população sofram alterações a cada geração que são os seguintes: **Seleção** - Replicação dos indivíduos mais aptos e eliminação de indivíduos menos aptos com quantidade de replicações e eliminações como parâmetro de entrada no algorimo.

DOI: 10.5540/03.2023.010.01.0051

Cruzamento - consiste em criar dois indivíduos a partir de dois indivíduos da população, de tal forma que estes indivíduos filhos herdem características dos indivíduos originais. Nesta versão proposta os indivíduos originais são trocados por estes novos indivíduos na população, de forma a manter a quantidade de indivíduos da população constante. A quantidade máxima de pares de indivíduos que sofreram ação deste operador a cada iteração é um parâmetro de entrada do algoritmo genético proposto.
Mutação - O operador de mutação é responsável por proporcionar pequenas perturbações em indivíduos arbitrários da população com taxa de perturbação máxima e quantidade máxima de indivíduos sob a ação deste operador previamente estipulados por parâmetros iniciais do algoritmo.

Para garantir que a sequência de populações obtidas a cada iteração seja uma sequência com indivíduos cada vez mais aptos utilizaremos o chamado **elitismo** que consiste em levar o melhor indivíduo de uma geração para a próxima geração. O elitismo faz com que o indivíduo mais apto da geração seguinte não seja menos apto do que o indivíduo mais apto da geração anterior. Uma versão simplificada do Algoritmo Genético pode ser vista no Algoritmo 2.

Algoritmo 2: Algoritmo Genético (GA)

Entrada: $k, tamanho, MAX, r, s, pmut, pcruz$
Saída: Melhor solução M;
início
  $Pop = CriaPopulacao(k, tamanho, r)$;
  $M = MelhorSolucao(Pop)$;
  $i = 0$;
  caso $(i < MAX)$ **faça**
    $Selecao(Pop, s)$;
    $Cruzamento(Pop, pcruz)$;
    $Mutacao(Pop, pmut)$;
    $Elitismo(Pop, M)$;
    $M = MelhorSolucao(Pop)$;
    $i = i + 1$;
  fim
fim

Os seguintes parâmetros foram utilizados no GA, após observações empíricas, de tal forma a equilibrar a eficiência computacional com a precisão na busca por soluções de alta aptidão:

- Parâmetro k do problema da k-dispersão max-min no círculo C_r para $r = 1000$;
- Tamanho da população: $tamanho = 30$;
- Quantidade de gerações: $MAX = 1000$;
- Parâmetro de seleção por rateio: $s = 3$;
- Porcentagem máxima de cruzamento sobre a população: $pcruz = 20\%$;
- Porcentagem máxima de troca de genes entre dois indivíduos da população no cruzamento: $pgene = 30\%$;
- Porcentagem de mutação sobre a população: $pmut = 30\%$.

O algoritmo foi rodado 100 vezes com os parâmetros acima.

DOI: 10.5540/03.2023.010.01.0051

4 Simulated Annealing (SA)

O Simulated Annealing é uma meta-heurística inspirada no fenômeno da termodinâmica do recozimento. Para isto, o Simulated Annealing propõe explorar o espaço de busca através de uma estratégia de múltiplos reinícios aleatórios cuja transição é controlada através de uma simplificação da distribuição de probabilidade proposta por Maxwell-Boltzmann e com um esquema de resfriamento da temperatura ao longo das iterações. Critério de parada para este algoritmo é o resfriamento chegar na zona limítrofe. O Algoritmo 3 contém a versão proposta do Simulated Annealing, onde a solução C próxima de B é criada perturbando os pontos de B em um raio de, no máximo, 20% de r/k, para efetuar buscas locais levando em consideração o número de pontos k.

Algoritmo 3: Algoritmo Simulated Annealing (SA)

Entrada: Solução inicial B^* e parâmetros α e MAX
Saída: Solução B

```
início
    B := B*;
    Começar com a temperatura (inicial) T := 1;
    caso (T > 0.00001) faça
        i = 0
        para i < MAX faça
            i := i + 1;
            Gerar uma solução C próxima de B;
            D = f(B) − f(C);
            Selecionar aleatoriamente x ∈ [0, 1];
            se (D < 0) ou exp(−D/T) > x então
                B := C;
        fim
        T = αT;
    fim
fim
```

Os seguintes parâmetros foram utilizados no SA, após observações empíridas relacionadas a eficiência computacional e valores obtidos para aproximações para a constante de k-dispersibidade:

- Parâmetro k do problema da k-dispersão max-min no círculo C_r para $r = 1000$;
- Temperatura inicial: $T = 1$;
- Quantidade de iterações para cada temperatura: $MAX = 1000$;
- Parâmetro de resfriamento: $\alpha = 0.1$;
- Parâmetro de perturbação de 20% de r/k.

O algoritmo foi rodado 100 vezes com os parâmetros acima.

5 Resultados e Análises

Todos os resultados experimentais apresentados são obtidos utilizando um laptop com processador Intel Core i7 4510U com CPU de 2.00 GHz e 8GB de memória e sistema operacional Linux da Debian utilizando a linguagem de programação Python.

DOI: 10.5540/03.2023.010.01.0051

Para cada número de pontos k e para cada algoritmo, seja d a melhor aproximação para a constante de k-dispersividade obtida em um experimento.

Na tabela 1 foram obtidos os seguintes parâmetros para cada número de pontos k após repetirmos o experimento 100 vezes para cada um dos algoritmos:

- d_{min} e d_{max}, o menor (pior) e maior (melhor) valor encontrado, respectivamente, entre os 100 valores para d encontrados como aproximações para a constante de k-dispersividade obtidos nos 100 experimentos;
- $\overline{d}$ é a média aritmética entre os 100 valores para d obtidos como aproximações para a constante de k-dispersividade;
- σ_d representa o desvio padrão entre todos os 100 valores de d obtidos nos 100 experimentos realizados.

Tabela 1: Dados obtidos pelos Algoritmos DE, GA e SA

	k	2	3	4	5	6	7	8	9	10
DE	$\overline{d}$	2.00000	1.73205	1.41421	1.16840	0.99423	0.95254	0.80888	0.70570	0.65467
	σ_d	0.00000	0.00000	0.00001	0.00344	0.01411	0.05837	0.06775	0.04064	0.04774
	d_{max}	2.00000	1.73205	1.41421	1.17557	1.00000	1.00000	0.86777	0.76396	0.70888
	d_{min}	2.00000	1.73205	1.41416	1.00000	0.92747	0.74899	0.59837	0.56536	0.47050
GA	$\overline{d}$	1.99976	1.73022	1.37776	1.06956	0.97666	0.85573	0.71869	0.63047	0.55388
	σ_d	0.00013	0.00086	0.11175	0.07909	0.03250	0.08503	0.04495	0.04643	0.04710
	d_{max}	1.99999	1.73144	1.41318	1.17240	0.99942	0.99493	0.83310	0.72001	0.63773
	d_{min}	1.99923	1.72739	0.99837	0.99217	0.88187	0.67574	0.59862	0.48314	0.42531
SA	$\overline{d}$	1.99996	1.73183	1.35582	1.10120	0.99920	0.98166	0.81708	0.71150	0.64955
	σ_d	0.00003	0.00011	0.14370	0.08659	0.00053	0.04772	0.04230	0.02469	0.02949
	d_{max}	2.00000	1.73201	1.41412	1.17535	0.99992	0.99936	0.86646	0.75274	0.69186
	d_{min}	1.99986	1.73148	0.99922	0.99890	0.99740	0.82122	0.71937	0.62547	0.54851

Os resultados experimentais obtidos nos mostram que o Algoritmo Evolução Diferencial obteve melhores soluções do que o Algoritmo Simulated Annealing que por sua vez obteve melhores soluções do que o Algoritmo Genético, uma vez que os valores de d_{max} obtidos no Algoritmo Evolução Diferencial são maiores do que os valores obtidos de d_{max} pelo Algoritmo Simulated Annealing e que, por sua vez, são maiores do que os valores de d_{max} obtidos pelo Algoritmo Genético. Usando a Distribuição t de Student para testar a hipótese de comparação entre os valores médios das aproximações para a constante de k-dispersividade encontradas, podemos chegar a conclusão que o Algoritmo Evolução Diferencial possui valores médios significativamente maiores do que o Algoritmo Genético com probabilidade de erro em 0.01, possui valores médios significativamente maiores do que o Algoritmo Simulated Annealing para k de 2 até 7 com probabilidade de erro em 0.01 e possuem valores médios estatisticamente equivalentes aos do Simulated Annealing para valores de k maiores do que 7. Na comparação dos valores médios encontrados nas aproximações para a constante de k-dispersividade obtidos entre o Algoritmo Genético e o Algoritmo Simulated Annealing, obtemos que os valores médios obtidos pelo Algoritmo Simulated Annealing são significativamente maiores do que os valores médios obtidos pelo Algoritmo Genético com probabilidade de erro em 0.01 com exceção de k=4, onde não houve diferença entre os valores médios com significância estatística. Os valores obtidos para constantes de k-dispersividade nos sugerem que soluções para o problema da 2-dispersão no círculo são pontos diametralmente opostos, sugerem que soluções para o problema da k-dispersão no círculo para $3 \leq k \leq 5$ são vértices de polígonos regulares de k lados inscritos no círculo, enquanto que no caso $k = 6$ sugere como solução ou os vértices de um polígono regular de 6 lados inscrito no círculo ou os vértices um polígono de 5 lados

cujos lados são maiores ou iguais ao raio do círculo e um ponto no centro do círculo. A constante de 7-dispersividade sugere como solução para o problema da 7-dispersão no círculo os vértices de um polígono regular de 6 lados inscrito no círculo com o ponto no centro deste círculo. Os demais casos parecem não seguir este padrão.

6 Considerações Finais

Podemos notar que os Algoritmos considerados encontram boas aproximações para a constante de k-dispersividade no círculo, embora o algorimo Evolução Diferencial obteve melhores resultados quando comparados com o Algoritmo Genético e o Simulated Annealing. Resulados experimentais vislumbram a possibilidade de encontrarmos uma solução para o problema da k-dispersividade para cada valor de k, embora para os casos em que $k \geq 8$ os resultados sugerem que novas idéias sejam necessárias para a determinação de alguma solução. Embora neste trabalho consideramos o problema da k-dispersividade no círculo considerando a métrica max-min, poderíamos utilizar a métrica max-soma para obtermos resultados a constante de k-dispersividade para este caso ou mesmo considerar o problema de resolvermos o problema bi-objetivo da k-dispersividade max-min e o da k-dispersividade max-soma.

Agradecimentos

Agradecemos ao revisores pelos comentários e sugestões que contribuiram para o aperfeiçoamento deste trabalho.

Referências

[1] Toshihiro Akagi et al. “Exact Algorithms for the Max-Min Dispersion Problem”. Em: **Frontiers in Algorithmics**. Ed. por Jianer Chen e Pinyan Lu. Cham: Springer International Publishing, 2018, pp. 263–272. ISBN: 978-3-319-78455-7.

[2] T. Araki e Si. Nakano. “Max–min dispersion on a line”. Em: **Journal of Combinatorial Optimization** 44 (2022), pp. 1824–1830.

[3] R. L. Church e R. S Garfinkel. “Locating an Obnoxious Facility on a Network”. Em: **Trans. Sci.** 12 (1978), pp. 107–118.

[4] E. Erkut e S. Neuman. “Analitical Models for Locating Undesirable Facilities”. Em: **EJOR** 40 (1989), pp. 275–291.

[5] Takashi Horiyama et al. “Max-Min 3-Dispersion Problems”. Em: **Computing and Combinatorics**. Ed. por Ding-Zhu Du, Zhenhua Duan e Cong Tian. Cham: Springer International Publishing, 2019, pp. 291–300. ISBN: 978-3-030-26176-4.

