Algoritmos genéticos aplicados a robótica

Pedro Henrique Centenaro

1 Visão geral de problemas de otimização

Os problemas de otimização, também conhecidos como problemas de programação matemática, envolvem encontrar um ou mais valores ótimos para um conjunto de expressões matemáticas. Um ótimo pode ser um valor que maximiza uma expressão (máximo) ou que minimiza uma expressão (mínimo). Quando o problema em questão envolve a determinação de máximos, chama-se problema de maximização – ou minimização, no caso de mínimos.

Em cursos de cálculo, os problemas de otimização propostos costumam ser relativamente simples. Geralmente, apresenta-se uma *função objetivo* de uma variável e pede-se que seja maximizada ou minimizada. Neste caso, a resolução costuma ser simples, de acordo com o seguinte passo a passo [15]:

1. Dada a função a otimizar, $f(x)$, determinar as derivadas $f'(x)$ e $f''(x)$.
2. Encontrar os zeros de $f'(x)$, conhecidos como pontos críticos de $f(x)$.
3. Sejam $x_1, \ldots, x_n$ os pontos críticos de f. Calcular $y_k = f''(x_k) \forall k \in \{1, \ldots, n\}$.
4. Se o problema for de maximização, os pontos de máximo são os x_k para os quais $y_k < 0$. Se o problema for de minimização, os pontos de mínimo são os x_k para os quais $y_k > 0$.

Nota-se que podem existir vários pontos máximos e mínimos. É possível, portanto, que x_a e x_b sejam ambos pontos de máximo, com $x_a > x_b$. Isso significa que x_a e x_b resultam nos maiores valores de $f(x)$ em intervalos específicos da função. A Figura 1 exemplifica graficamente esta questão. A função ilustrada contém dois pontos de máximo e um ponto de mínimo. Em relação aos pontos de máximo, o ponto à esquerda está claramente acima de todos os outros pontos da função. Por esta razão, diz-se que este é um máximo *global* da função. Já o ponto à direita, por ser máximo de um intervalo reduzido do domínio da função, é chamado de máximo *local*. Por fim, apesar de a função ter um único ponto de mínimo, este não é um mínimo global, dado que $x \to -\infty \Rightarrow f(x) \to -\infty$ e $x \to \infty \Rightarrow f(x) \to -\infty$. Ou seja, é possível que um problema não tenha ótimos globais.

Problemas reais de otimização costumam ser mais complexos do que isso. Um exemplo clássico considera uma empresa que precisa decidir qual quantidade de cada produto produzir para maximizar seu lucro [16]. Neste caso, não faz sentido qualquer solução onde a quantidade de determinado produto seja < 0 e, portanto, é necessário impor *restrições* ao modelo do problema. Também é importante observar que n produtos

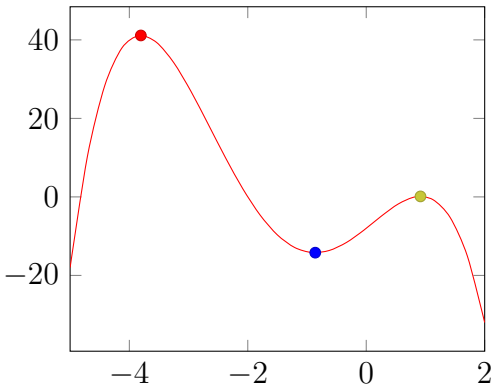

Figura 1: Pontos de máximo e mínimo de uma função

diferentes representam n quantidades diferentes que devem ser definidas para resolver o problema de otimização. Ou seja, a função objetivo é uma função de n variáveis.

Todos os fatores apresentados até aqui conspiram contra a resolução do problema por métodos manuais. De fato, a maioria dos problemas de otimização tem sua resolução feita por métodos computacionais, como o método simplex [2, 3], comumente utilizado para resolver problemas de programação linear – ou seja, problemas cuja função objetivo e cujas restrições são lineares. Entretanto, mesmo o simplex tem suas limitações. Em trabalho de iniciação científica realizado sobre este método [5], constatou-se que os principais fatores que dificultam a resolução de problemas são:

- Grande número de restrições: A resolução de problemas de otimização por meio do método simplex é baseada numa interpretação geométrica das restrições impostas sobre o modelo. Assim, quanto maior o número de restrições, mais complexa é a geometria do problema, o que pode resultar em mais passos para resolvê-lo.

- Restrições inteiras: Pode-se querer que certas variáveis do problema sejam números naturais, inteiros ou binários. Neste caso, o método simplex deixa de funcionar, e costumam-se aplicar métodos de otimização inteira como o *branch-and-bound*, que cria uma árvore de sub-problemas que podem ser resolvidos usando o simplex. Tal árvore pode crescer rapidamente, executando o simplex várias vezes, sem a garantia de convergência para um valor ótimo em cada galho.

No caso de problemas não-lineares, a otimização é ainda mais difícil, pois existem vários métodos diferentes, que têm vantagens e desvantagens dependentes de características matemáticas específicas de cada problema [12].

1.1 Heurísticas para otimização

Métodos como o simplex são ditos *exatos*, pois são feitos para retornar valores ótimos de fato. Contudo, como arguido anteriormente, problemas muito complexos podem ter uma resolução exata muito lenta. Por este motivo, muitos problemas práticos de otimização são resolvidos por *heurísticas*. As heurísticas são métodos que visam simplificar modelos de otimização de modo a resolvê-los mais rapidamente. Frequentemente, isso resulta em soluções não-ótimas, de onde resulta que uma "boa" heurística deve ser rápida e, ao mesmo tempo, conduzir a valores próximos do ótimo. Convém observar que heurísticas costumam ser separadas em duas categorias, que são apresentadas a seguir.

Heurísticas convencionais

Heurísticas convencionais são projetadas para resolver problemas específicos. Um exemplo é a heurística de George e Robinson [8] para carregamento de contêineres com itens cuboides. Para concluir este objetivo, a heurística fatia o contêiner em vários volumes cuboides menores e aplica uma série de procedimentos para preenchê-los. Uma heurística popular para resolver problemas de roteamento de veículos é o método *nearest neighbor* [10], que consiste em construir rotas ponto a ponto, sempre escolhendo o ponto não visitado menos distante como próximo ponto do trajeto. Como estas heurísticas criam soluções do zero, são chamadas de heurísticas *construtivas*. Geralmente, heurísticas construtivas são aperfeiçoadas por heurísticas *de melhoria*, que costumam introduzir alterações aleatórias nas soluções geradas em busca de soluções melhores.

Meta-heurísticas

Em geral, as meta-heurísticas operam em uma de duas maneiras. No primeiro caso, o método extrai informações específicas sobre o problema e em seguida escolhe uma heurística que parece mais apropriada para resolvê-lo. No segundo caso, a meta-heurística é baseada em processos naturais que podem ser abstraídos para resolver uma vasta gama de problemas. É possível citar métodos como recozimento simulado (*simulated annealing*), algoritmos genéticos, colônia de formigas, busca tabu, entre outros. Como exemplo, o primeiro procedimento citado é descrito a seguir.

O recozimento simulado é baseado numa técnica metalúrgica de remoção de tensões internas provenientes de deformidades na estrutura cristalina de um metal. De acordo com Delahaye, Chaimatanan e Mongeau [6], o processo consiste em aquecer o metal até seu ponto de fusão. Neste estado, os átomos que constituem o metal estão dispersos

aleatoriamente no espaço. Em seguida, inicia-se um processo lento de resfriamento, após o qual os átomos tendem a se distribuir de forma homogênea, de modo a minimizar a energia retida na estrutura cristalina.

Para imitar os procedimentos descritos, o recozimento simulado precisa de uma função (f) capaz de avaliar soluções para o problema especificado, o que é análogo a medir a energia da estrutura cristalina do metal. No lugar de tempo, usa-se o número de iterações do método (k) para "cronometrar" o processo. Além disso, introduz-se um parâmetro de temperatura (T) que é reduzido a cada iteração. O método costuma se resumir aos seguintes passos:

1. Gerar uma solução inicial e inicializar os parâmetros.
2. Gerar uma nova solução, fazendo modificações na solução atual.
3. Comparar a solução nova à atual usando a função f.
4. a) Se a solução nova for melhor do que a atual, ela passa a ser a solução atual.
   b) Se a solução nova for pior do que a atual, utiliza-se um método aleatório para determinar se ela deve substituir a atual, de modo que as chances sejam maiores quando a temperatura T for maior.
5. Reduzir a temperatura T.
6. Se a temperatura T tiver chegado ao seu valor mínimo, terminar o algoritmo e retornar a solução atual. Do contrário, voltar ao passo 2.

2 Algoritmos genéticos

Algoritmos genéticos (AGs) são uma meta-heurística baseada em princípios de seleção natural para obtenção de boas soluções para problemas. As características dos AGs os tornam excelentes ferramentas para vários propósitos diferentes. Nesta seção, são descritos os passos que todos os AGs devem seguir, e são apresentadas aplicações destes algoritmos.

2.1 Estrutura de algoritmos genéticos

Os AGs simulam mecanismos evolutivos para obter soluções para problemas. Para tal, os AGs dependem de uma operação de conversão de soluções para cromossomos e vice-versa. Um cromossomo é uma estrutura de dados que contém as informações necessárias para reconstruir e avaliar uma solução. Geralmente, utilizam-se vetores de números para representar um cromossomo. Um exemplo simples de cromossomo é descrito por Hermawanto [9], que propõe um AG para resolução de um problema de minimização de quatro variáveis. Para resolver este problema, o autor representa os cromossomos como vetores de quatro números inteiros, cada um correspondendo ao valor de uma variável. Em problemas mais complexos, a conversão cromossomial pode ser mais complicada. No caso do exemplo citado, como os genes informam diretamente os valores das variáveis do problema, diz-se que a informação necessária para decodificar o cromossomo está no seu *genótipo*. No entanto, se os valores nos genes precisarem passar por algum processo de "tradução" para corresponderem a informações concretas, diz-se que as informações das variáveis estão contidas no *fenótipo* do cromossomo [7].

Após a determinação do processo de conversão cromossomial supracitado, é necessário determinar uma função de aptidão (*fitness*), ou seja, uma função que retorne valores através dos quais seja possível comparar cromossomos. Na realidade, tal função não precisa ser matematicamente objetiva – o processo de classificação de uma solução pode ser subjetivo [14], o que significa que AGs podem ser utilizados para resolver problemas de otimização de caixa preta (*blackbox*), ou seja, problemas cuja função objetivo ou restrições não são conhecidas ou bem-definidas [1]. Feito isso, resta implementar os passos seguintes, cuja descrição é baseada, principalmente, na obra de Sastry, Goldberg e Kendall [14].

Inicialização

Na etapa de inicialização, as primeiras soluções para o problema são geradas. Tais soluções podem ser completamente aleatórias ou advir de heurísticas de construção espe-

cializadas. Uma diferença importante entre os AGs e as técnicas de recozimento simulado (RS) está no fato que o RS mantém uma única solução ao longo de todo o processo de resolução do problema, ao passo que nos AGs existem $n \geq 2$ soluções. O conjunto de soluções é chamado de *população*, e traz consigo a vantagem de explorar, concorrentemente, várias alternativas diferentes de resolução de problema (a este respeito, o trabalho de Murawski e Bossaerts [11] é bastante elucidativo, mostrando como vários agentes, utilizando diferentes heurísticas, descobrem uma parte muito maior do conjunto solução de um problema do que agentes individuais).

Apesar do poder de exploração da população, é necessário escolher o seu tamanho com cautela. Populações muito pequenas podem apresentar baixa variedade cromossomial e, consequentemente, convergir para ótimos locais ao longo do processo evolutivo. Por outro lado, populações muito grandes podem ser computacionalmente ineficientes, se populações menores forem capazes de atingir os mesmos resultados [13]. Para problemas com custo computacional muito elevado, Delahaye, Chaimatanan e Mongeau [6] recomendam a utilização de métodos sem população, como o RS.

Avaliação

Na avaliação, os cromossomos da população atual são avaliados pela função de aptidão. Como abordado anteriormente, esta função pode ser objetiva ou subjetiva – o importante é que, ao fim do processo, os cromossomos possam ser comparados. Quando o valor da função de aptidão é objetivo, é necessário especificar o que faz de um valor melhor ou pior do que o outro. Por exemplo, no AG de Hermawanto [9] para resolução de problemas de minimização, a função de aptidão é a própria função a minimizar e, portanto, soluções que obtiverem valores menores serão melhores. Se o problema a resolver fosse de maximização, valores maiores seriam melhores.

Seleção

O processo de seleção leva em conta os resultados da etapa de avaliação para definir casais (pares) de cromossomos. Os métodos de seleção costumam implementar processos estocásticos em que os cromossomos com avaliações melhores são favorecidos. Na sequência, são abordados alguns métodos.

Método da roleta [4]: Sejam $n \geq 2$ o tamanho da população e q_i $(i = 1, \ldots, n)$ os valores de aptidão de cada solução. A cada solução i é associada uma probabilidade $p_i = q_i / \sum_{j=1}^{n} q_j$ de que i seja selecionada. À solução 1 fica atribuído o intervalo $I_1 = [0, p_1]$. A partir disso, é possível determinar os intervalos das soluções $k = 2, \ldots, n$, em sequência,

com $I_k = \left(\sum_{j=1}^{k-1} p_j, \sum_{j=1}^{k} p_j\right]$. Evidentemente, $\cup_{j=1}^{n} I_j = [0, 1]$, então, gera-se um valor aleatório $r \in [0, 1]$, de modo que $r \in I_k$ significa que a solução k deve ser selecionada.

A título de exemplo, supõe-se uma população de cinco soluções, cujos dados são apresentados na Tabela 1. A roleta obtida é apresentada na Figura 2. Desta maneira, fica claro que as melhores soluções têm mais chances de serem selecionadas para cruzamento.

Tabela 1

Solução (k)	Aptidão (q_k)	Probabilidade (p_k)	Intervalo (I_k)
1	4.52	0.156	[0, 0.156]
2	9.04	0.313	(0.156, 0.469]
3	7.33	0.254	(0.469, 0.723]
4	3.21	0.111	(0.723, 0.834]
5	4.79	0.166	(0.834, 1]

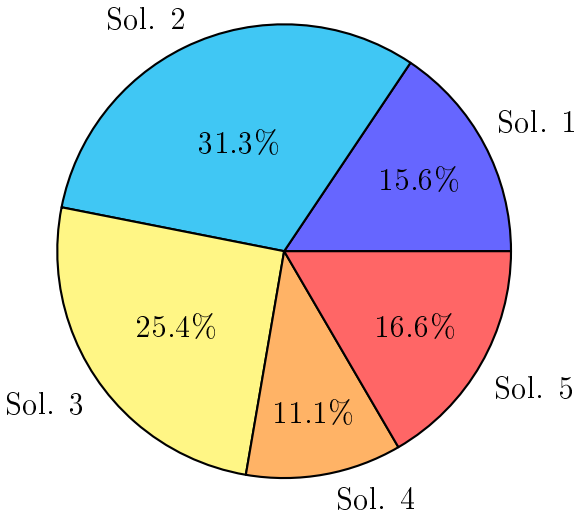

Figura 2: Probabilidades de seleção de soluções pelo método da roleta.

Referências

[1] Stéphane Alarie, Charles Audet, Aïmen E. Gheribi, Michael Kokkolaras e Sébastien Le Digabel. "Two Decades of Blackbox Optimization Applications". Em: *EURO Journal on Computational Optimization* 9 (2021), p. 100011. DOI: 10.1016/j.ejco.2021.100011. URL: https://www.sciencedirect.com/science/article/pii/S2192440621001386 (acesso em 05/06/2024).

[2] Mokhtar S. Bazaraa, John J. Jarvis e Hanif D. Sherali. *Linear Programming and Network Flows*. 4ª ed. Wiley, 2010.

[3] Dimitris Bertsimas e John N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, Dynamic Ideas, 1997.

[4] André Ponce de Leon F. de Carvalho. *Algoritmos Genéticos*. URL: https://sites.icmc.usp.br/andre/research/genetic/ (acesso em 05/06/2024).

[5] Pedro Henrique Centenaro. *Fundamentos Matemáticos e Computacionais de Problemas de Roteamento de Veículos*. 2023. URL: https://repositorio.ufsc.br/handle/123456789/250623.

[6] Daniel Delahaye, Supatcha Chaimatanan e Marcel Mongeau. "Simulated Annealing: From Basics to Applications". Em: *Handbook of Metaheuristics*. Ed. por Michel Gendreau e Jean-Yves Potvin. Vol. 272. Cham: Springer International Publishing, 2019, pp. 1–35. DOI: 10.1007/978-3-319-91086-4_1. URL: http://link.springer.com/10.1007/978-3-319-91086-4_1 (acesso em 05/06/2024).

[7] Michel Gendreau e Jean-Yves Potvin, ed. *Handbook of Metaheuristics*. Vol. 146. International Series in Operations Research & Management Science. Boston, MA: Springer US, 2010. DOI: 10.1007/978-1-4419-1665-5. URL: https://link.springer.com/10.1007/978-1-4419-1665-5 (acesso em 03/06/2024).

[8] J.A. George e D.F. Robinson. "A Heuristic for Packing Boxes into a Container". Em: *Computers & Operations Research* 7.3 (1980), pp. 147–156. DOI: 10.1016/0305-0548(80)90001-5. URL: https://linkinghub.elsevier.com/retrieve/pii/0305054880900015 (acesso em 06/04/2024).

[9] Denny Hermawanto. "Genetic Algorithm for Solving Simple Mathematical Equality Problem". Em: (2013).

[10] Fei Liu, Chengyu Lu, Lin Gui, Qingfu Zhang, Xialiang Tong e Mingxuan Yuan. *Heuristics for Vehicle Routing Problem: A Survey and Recent Advances*. 2023. arXiv: 2303.04147 [cs, math]. URL: http://arxiv.org/abs/2303.04147 (acesso em 05/06/2024).

[11] Carsten Murawski e Peter Bossaerts. "How Humans Solve Complex Problems: The Case of the Knapsack Problem". Em: *Scientific Reports* 6.1 (2016), p. 34851. DOI: 10.1038/srep34851. URL: https://www.nature.com/articles/srep34851 (acesso em 05/06/2024).

[12] Ademir Alves Ribeiro e Elizabeth Wegner Karas. *Otimização Contínua: Aspectos Teóricos e Computacionais*. 1ª ed. Curitiba: Editora Cengage, 2013.

[13] Olympia Roeva, Stefka Fidanova e Marcin Paprzycki. "Influence of the Population Size on the Genetic Algorithm Performance in Case of Cultivation Process Modelling". Em: (2013).

[14] "Genetic Algorithms". Em: *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. Ed. por Kumara Sastry, David Goldberg e Graham Kendall. New York: Springer, 2005.

[15] James Stewart. *Cálculo*. 8ª ed. Vol. 1. Cengage Learning, 2016.

[16] Xiaolan Zhang. "Linear Programming". Em: (2020). URL: https://storm.cis.fordham.edu/~zhang/cs5835/slides/LinearProgramming_handout.