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 [8]:

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 [9]. 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

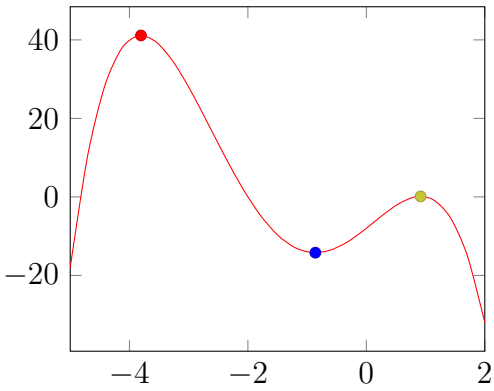

Figura 1: Pontos de máximo e mínimo de uma função

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 [1, 2], 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 [3], 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 [7].

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 [5] 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* [6], 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 [4], 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 problemas que os AGs têm ajudado a resolver, e, em seguida, são definidos os passos que todos os AGs devem seguir.

Referências

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

[2] Dimitris Bertsimas e John N. Tsitsiklis. *Introduction to Linear Optimization*. 4ª ed. Belmont, Massachusetts: Athena Scientific e Dynamic Ideas, 1997.

[3] Pedro Henrique Centenaro e Luiz-Rafael Santos. *Fundamentos matemáticos e computacionais de problemas de roteamento de veículos*. 2023. URL: https://repositorio.ufsc.br/handle/123456789/250623.

[4] Daniel Delahaye, Supatcha Chaimatanan e Marcel Mongeau. "Simulated annealing: From basics to applications". Em: *Handbook of metaheuristics* (2019), pp. 1–35.

[5] J.A. George e D.F. Robinson. "A heuristic for packing boxes into a container". en. 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).

[6] Fei Liu, Chengyu Lu, Lin Gui, Qingfu Zhang, Xialiang Tong e Mingxuan Yuan. "Heuristics for vehicle routing problem: A survey and recent advances". Em: *arXiv preprint arXiv:2303.04147* (2023).

[7] Ademir Alves Ribeiro e Elizabeth Wegner Karas. *Otimização Contínua: Aspectos Teóricos e Computacionais*. Curitiba: Cengage Learning, 2013.

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

[9] Xiaolan Zhang. *Linear Programming CISC5835, Algorithms for Big Data CIS, Fordham Univ.* 2020. URL: https://storm.cis.fordham.edu/~zhang/cs5835/slides/LinearProgramming_handout.