

JuMP - Julia para Programação Matemática

Joaquim Dias Garcia

PSR

Rio de Janeiro, RJ, Brasil

joaquim@psr-inc.com

RESUMO

JuMP é uma linguagem *open-source* de modelagem e um ecossistema de pacotes para otimização na linguagem de programação Julia. JuMP permite formular e resolver problemas de otimização de variadas classes de forma fácil, independente de otimizadores e computacionalmente eficiente. Com a biblioteca é possível formular problemas de programação linear, programação inteira, programação não linear, programação cônica e programação por restrições e otimizar tais problemas com mais de 50 otimizadores. JuMP vem sendo usado por diversos agentes do governo, indústria e academia em diversas áreas de tradicional aplicação de pesquisa operacional como sistemas elétricos, controle de estoque, otimização em redes, mas também outras áreas como física teórica e biologia molecular. Adicionalmente, é possível criar extensões da infraestrutura básica para modelar problemas específicos como otimização multiestágio estocástica, otimização binível, otimização polinomial e otimização em dimensão infinita.

PALAVRAS CHAVE. JuMP, Julia, Programação Matemática, Otimização, Linguagem de Modelagem Algébrica.

PM – Programação Matemática, OC – Otimização Combinatória

ABSTRACT

JuMP is an open-source modeling language and an ecosystem of packages for optimization in the Julia programming language. JuMP allows you to formulate and solve optimization problems of various classes in an easy, optimizer-independent, and computationally efficient way. With the library it is possible to formulate linear programming, integer programming, nonlinear programming, conic programming and constraint programming problems and optimize these problems with more than 50 optimizers. JuMP has been used by several agents of government, industry and academia in several areas of traditional application of operations research such as electrical systems, inventory control, network optimization, but also other areas such as theoretical physics and molecular biology. Additionally, it is possible to create extensions of the underlying infrastructure to model specific problems such as stochastic multistage optimization, bilevel optimization, polynomial optimization, and infinite-dimensional optimization.

KEYWORDS. JuMP, Julia, Mathematical Programming, Optimization, Algebraic Modelling Language.

PM – Mathematical Programming, OC – Combinatorial Optimization

1. Introdução

JuMP é uma linguagem de modelagem matemática disponível na linguagem de programação Julia. JuMP permite ao usuário modelar problemas de diversas classes como Programação Linear, Programação Inteira, Programação Cônica, Programação Semidefinida, Programação Não Linear e Programação por Restrições. Além da definição de modelos, JuMP facilita a interação com os modelos e a comunicação com mais de 50 *solvers* de modo transparente para o usuário.

O fato de JuMP ser uma biblioteca comum da linguagem Julia tem muitas consequências positivas. Primeiro, ambos Julia e JuMP são programas de código aberto com licenças muito permissivas (MIT e MPL 2.0, respectivamente) o que as permite ser usadas em diversos ambientes como industriais, acadêmicos e governamentais. Segundo, JuMP pode ser usada em conjunto com outros códigos como bibliotecas para lidar com bases de dado, gráficos, estruturas de dado avançadas totalmente independentes do JuMP. Terceiro, a alta performance comum a diversos códigos de qualidade na linguagem Julia se estende ao JuMP, o que permite o desenvolvimento de algoritmos sofisticados sem grandes gargalos. Quarto, JuMP pode ser facilmente estendido para diversos outros paradigmas de otimização como já foi feito em: i) `BilevelJuMP.jl`, para otimização em dois níveis [Garcia et al., 2024], ii) `SDDP.jl`, para otimização estocástica multi-estágio [Dowson e Kapelevich, 2021], iii) `PolyJuMP.jl`, para otimização polinomial [Legat, 2020], iv) `InfiniteOpt.jl` otimização em dimensão infinita [Pulsipher et al., 2022] entre muitos outros.

JuMP vem sendo utilizado em diversos meios como: governos, indústria e academia. Alguns exemplos típicos são enumerados a seguir. 1) Modelagem de infra estrutura de gás, água e energia [Bent et al., 2024], 2) Sistemas de potência [Jenkins e Sepulveda, 2017; Coffrin et al., 2018], 3) Computação quântica [Xavier et al., 2023], 4) Controle de estoque [Bouvier et al., 2024], 5) Otimização em redes ferroviárias [Gupta et al., 2016], 6) Biologia molecular [Langlieb et al., 2023], 7) Teoria de campos na física [White et al., 2021].

Uma ferramenta flexível e poderosa como o JuMP pode ser extremamente útil para a comunidade de pesquisa operacional. Vislumbramos grande potencial de acelerar a produção de novas descobertas e também de novos produtos industriais.

2. Objetivos

O objetivo é introduzir os alunos a linguagem de modelagem algébrica especializada em otimização: JuMP (*Julia for Mathematical Programming*). Em particular vamos:

- Revisar conceitos básicos da linguagem Julia e discutir ferramentas fundamentais para um desenvolvimento efetivo e confortável de modelos de otimização com JuMP.
- Detalhar os conceitos fundamentais da modelagem matemática em JuMP de modo que os alunos possam desenvolver seus próprios modelos com facilidade.
- Detalhar exemplos de usos das funcionalidades da biblioteca JuMP para descrever e otimizar problemas de diversas classes como: otimização linear, otimização inteira, otimização cônica, otimização não linear e programação por restrições.
- Apresentar conceitos intermediários como ferramentas para *debug* e técnicas para tornar códigos mais eficientes.
- Apresentar exemplos de extensões da biblioteca JuMP que permitem ao usuários modelar classes ainda mais complexas de problemas e comentar passos e referencias para criação de novas extensões.

3. Pré-requisitos

Os pré-requisitos serão mínimos:

- Familiaridade com linguagens de programação: ambientes de desenvolvimento, estruturas de dados simples, vetores, controle de fluxo, *loops*, funções, e *package managers*.
- Conhecimentos básicos de modelagem em otimização. Conceitos necessários: variáveis de decisão, restrições, função objetivo, solução primal, solução dual. Conhecimento básico de programação linear.

Com os dois grupos de pré-requisitos acima será possível acompanhar todo o curso. Outros conceitos importantes serão apresentados durante o mini-curso.

Conhecimentos adicionais de otimização inteira [Wolsey, 2020], cônica [Boyd e Vandenberghe, 2004], não-linear [Bazaraa et al., 2006], multi-objetivo [Ehrgott, 2005], bínivel [Beck e Schmidt, 2021], estocástica [Birge e Louveaux, 2011] permitirão um aproveitamento ainda mais efetivo.

4. Tópicos a serem abordados

1. Breve introdução / revisão da linguagem Julia. Começamos com os passos básicos para instalar Julia e o ambiente de programação VSCode. Apresentamos o *REPL* e referências de ajuda geral para a linguagem. Seguimos com números, vetores, *arrays* e dicionários, estruturas, funções, *loops*, fluxo de controle. Finalizamos com o *package manager* referências de pacotes úteis.
2. Conceitos básicos de JuMP. Instalação da biblioteca, estrutura geral do manual, discussão de outras referências úteis. Detalhamento de um exemplo simples: definições de variáveis, restrições e funções objetivo. Otimização e interação com resultados. Otimizadores e suas opções. Conjuntos e indexação de variáveis e restrições.
3. Modelagem de diversas classes de otimização. Programação Inteira. Programação Cônica. Programação Não Linear. Programação por restrições. Cada uma das classes será apresentada por meio de exemplos práticos.
4. Funcionalidades adicionais. Debugando modelos de otimização. Sugestões de performance para modelos de otimização. Chamadas sequenciais e modificações iterativas. *Callbacks*.
5. Bibliotecas complementares e extensões de programação matemática. Otimização Estocástica e Multi-estágio com *SDDP.jl*. Otimização Binível com *BilevelJuMP.jl*. Otimização Multi-Objetivo. Otimização Paramétrica. Comentários sobre criação de extensões de JuMP.

5. Bibliografia Básica

[JuMP Developers, 2024] Manual JuMP. Contém detalhamento de todas as funções da API. Lista de otimizadores disponíveis e tutorias de Julia e JuMP.

[Siqueira, 2020] Vídeo curso online de JuMP (em português). Neste curso de 5 vídeos no Youtube, Abel Siqueira explica diversas funcionalidades do pacote JuMP.

[Lubin et al., 2023] Artigo sobre a versão 1.0 do JuMP.

[Dowson, 2024] Tutorial de Julia e JuMP lecionado no *SESO 2023* e no *Julia and Optimization days 2023* - Paris.

[Bezanson et al., 2014] Manual de Julia.

[Bezanson et al., 2017] Artigo técnico sobre a linguagem Julia.

[Pulsipher, 2022] Tutorial sobre o pacote `InfiniteOpt.jl` que contem como primeira etapa tutorias de Julia e JuMP.

6. Bibliografia Complementar

[Lusby e Stidsen, 2023] Livro gratuito e disponível online. Além do PDF do livro, se encontram diversos *scripts* com exemplos de problemas modelados e resolvido com JuMP.

[Kwon, 2019] Livro, com versão online gratuita, sobre pesquisa operacional aplicada com códigos escritos em Julia. Há diversos exemplo de uso de JuMP no livro.

[Legat et al., 2022] Artigo descrevendo a biblioteca `MathOptInterface.jl` que contem os fundamentos e *backend* do JuMP 1.0.

[Lubin e Dunning, 2015] Artigo seminal com a primeira apresentação do JuMP.

[Dunning et al., 2017] Artigo com a primeira versão robusta do JuMP.

[Boyd e Vandenberghe, 2004] Livro sobre otimização convexa para a parte técnica sobre otimização cônica.

[Bertsimas e Tsitsiklis, 1997] Referência para programação linear e inteira.

[Wolsey, 2020] Referência para programação inteira.

[Maculan e Fampa, 2006] Referência, em português, para programação linear e inteira

Referências

Bazaraa, M. S., Sherali, H. D., e Shetty, C. M. (2006). *Nonlinear programming: theory and algorithms*. John wiley & sons.

Beck, Y. e Schmidt, M. (2021). A gentle and incomplete introduction to bilevel optimization.

Bent, R., Tasseff, B., e Coffrin, C. (2024). Infrastructuremodels: Composable multi-infrastructure optimization in julia. *INFORMS Journal on Computing*, 36(2):600–615.

Bertsimas, D. e Tsitsiklis, J. N. (1997). *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA.

Bezanson, J., Edelman, A., Karpinski, S., e Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98.

Bezanson, J., Karpinski, S., Shah, V., Edelman, A., et al. (2014). Julia language documentation. *The Julia Manual*, p. 1–261.

Birge, J. R. e Louveaux, F. (2011). *Introduction to stochastic programming*. Springer Science & Business Media.

- Bouvier, L., Dalle, G., Parmentier, A., e Vidal, T. (2024). Solving a continent-scale inventory routing problem at renault. *Transportation Science*, 58(1):131–151.
- Boyd, S. P. e Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.
- Coffrin, C., Bent, R., Sundar, K., Ng, Y., e Lubin, M. (2018). Powermodels. jl: An open-source framework for exploring power flow formulations. In *2018 Power Systems Computation Conference (PSCC)*, p. 1–8. IEEE.
- Dowson, O. (2024). Julia and optimization days 2023 and seso2023 - tutorials on julia and jump. <https://github.com/odow/SESO2023>. Acessado: 2024-06-06.
- Dowson, O. e Kapelevich, L. (2021). Sddp. jl: a julia package for stochastic dual dynamic programming. *INFORMS Journal on Computing*, 33(1):27–33.
- Dunning, I., Huchette, J., e Lubin, M. (2017). Jump: A modeling language for mathematical optimization. *SIAM review*, 59(2):295–320.
- Ehrgott, M. (2005). *Multicriteria optimization*, volume 491. Springer Science & Business Media.
- Garcia, J. D., Bodin, G., e Street, A. (2024). Bileveljump. jl: Modeling and solving bilevel optimization problems in julia. *INFORMS Journal on Computing*, 36(2):327–335.
- Gupta, S. D., Tobin, J. K., e Pavel, L. (2016). A two-step linear programming model for energy-efficient timetables in metro railway networks. *Transportation Research Part B: Methodological*, 93:57–74.
- Jenkins, J. D. e Sepulveda, N. A. (2017). Enhanced decision support for a changing electricity landscape: the genx configurable electricity resource capacity expansion model.
- JuMP Developers (2024). The jump manual. <https://jump.dev/JuMP.jl/stable/>. Acessado: 2024-06-06.
- Kwon, C. (2019). *Julia Programming for Operations Research*. Changhyun Kwon.
- Langlieb, J., Sachdev, N. S., Balderrama, K. S., Nadaf, N. M., Raj, M., Murray, E., Webber, J. T., Vanderburg, C., Gazestani, V., Tward, D., et al. (2023). The molecular cytoarchitecture of the adult mouse brain. *Nature*, 624(7991):333–342.
- Legat, B. (2020). *Thesis submitted in partial fulfillment of the requirements for the Degree of Doctor in Applied Sciences*. PhD thesis, Université catholique de Louvain.
- Legat, B., Dowson, O., Garcia, J. D., e Lubin, M. (2022). Mathoptinterface: a data structure for mathematical optimization problems. *INFORMS Journal on Computing*, 34(2):672–689.
- Lubin, M., Dowson, O., Garcia, J. D., Huchette, J., Legat, B., e Vielma, J. P. (2023). Jump 1.0: Recent improvements to a modeling language for mathematical optimization. *Mathematical Programming Computation*, 15(3):581–589.
- Lubin, M. e Dunning, I. (2015). Computing in operations research using julia. *INFORMS Journal on Computing*, 27(2):238–248.
- Lusby, R. e Stidsen, T. (2023). Mathematical programming with julia. URL <https://www.man.dtu.dk/mathprogrammingwithjulia>.

- Maculan, N. e Fampa, M. H. C. (2006). Otimização linear. *Editora Universidade de Brasília: Brasília*.
- Pulsipher, J. (2022). Infiniteopttutorials - short course. https://github.com/infiniteopt/InfiniteOptTutorials/tree/main/short_course. Acessado: 2024-06-06.
- Pulsipher, J. L., Zhang, W., Hongisto, T. J., e Zavala, V. M. (2022). A unifying modeling abstraction for infinite-dimensional optimization. *Computers & Chemical Engineering*, 156:107567.
- Siqueira, A. (2020). Tutoriais de julia em português - jump.jl parte 1 a 5. https://www.youtube.com/watch?v=vlhJsmMqpvw&list=PLOOY0eChAluyjKRQDeR4-LfsObNqslIUh&index=8&ab_channel=AbelSiqueira. Acessado: 2024-06-06.
- White, C. D., Cao, C., e Swingle, B. (2021). Conformal field theories are magical. *Physical Review B*, 103(7):075145.
- Wolsey, L. A. (2020). *Integer programming*. John Wiley & Sons.
- Xavier, P. M., Ripper, P., Andrade, T., Garcia, J. D., Maculan, N., e Neira, D. E. B. (2023). Qubo. jl: A julia ecosystem for quadratic unconstrained binary optimization. *arXiv preprint arXiv:2307.02577*.