What’s new in JuMP

Miles Lubin (Hudson River Trading*)

ISMP 2024

*Speaking in a personal capacity. Views and opinions expressed do not necessarily reflect those of HRT.

What is JuMP?

Part of the zoo of algebraic modeling languages

CMPL, CPLEX Concert, FICO Xpress Mosel + APIs, GNU MathProg, Gurobi APIs, linopy, MATLAB “problem-based optimization workflow”, Mosek Fusion, MOSEL, ompr, OPTMODEL, PuLP, PyOptInterface, Python-MIP, YALMIP, ZIMPL, …

What is JuMP?

An algebraic modeling language in Julia

```
using JuMP, Ipopt
function constrained_linear_regression(A::Matrix, b::Vector)
    model = Model(Ipopt.Optimizer)
    @variable(model, 0 <= x[1:size(A, 2)] <= 1)
    @variable(model, residuals[1:size(A, 1)])
    @constraint(model, residuals == A * x - b)
    @constraint(model, sum(x) <= 1)
    @objective(model, Min, sum(r^2 for r in residuals))
    optimize!(model)
    return value.(x)
end
A, y = rand(30, 20), rand(30)
x = constrained_linear_regression(A, b)
```

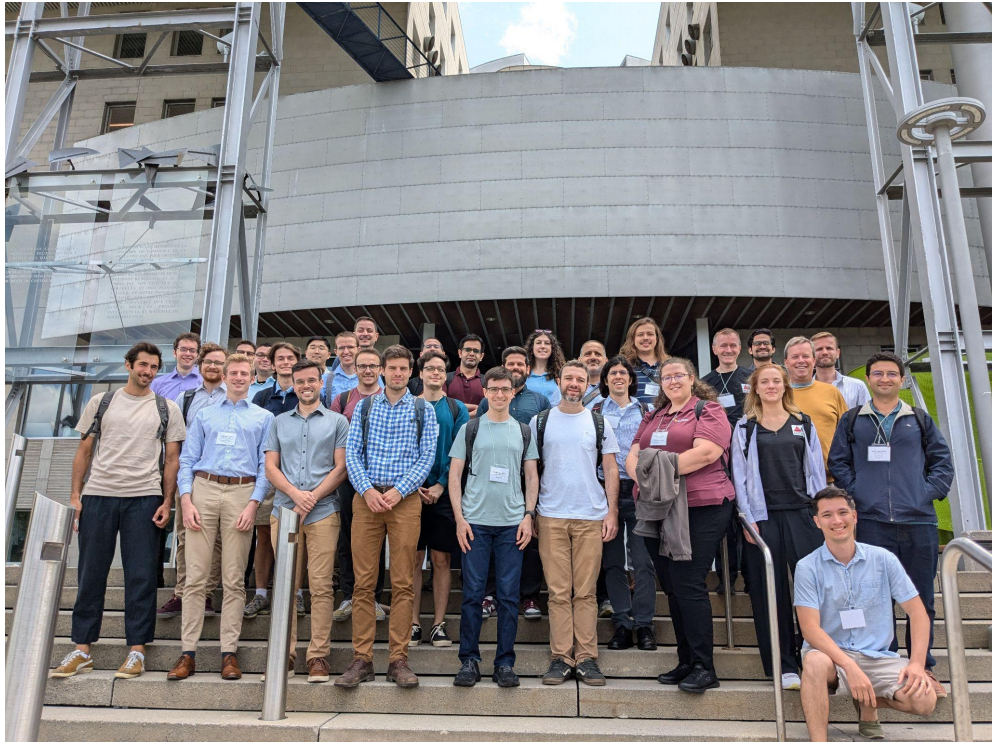
Why is JuMP interesting?

- Nice syntax
- Comprehensive documentation
- Vibrant community
- Open source
- Solver independent (50+ supported solvers)
- Embedded in Julia
- Supports interacting with solvers while they’re running
- Low overhead for model generation
- Extensible to new solvers
- Extensible to new problem classes

Who is (some of) JuMP?

https://github.com/jump-dev/JuMP.jl/graphs/contributors

JuMP-dev 2024



In the last 12 months of github.com/jump-dev…

>695,000
downloads of jump-dev packages

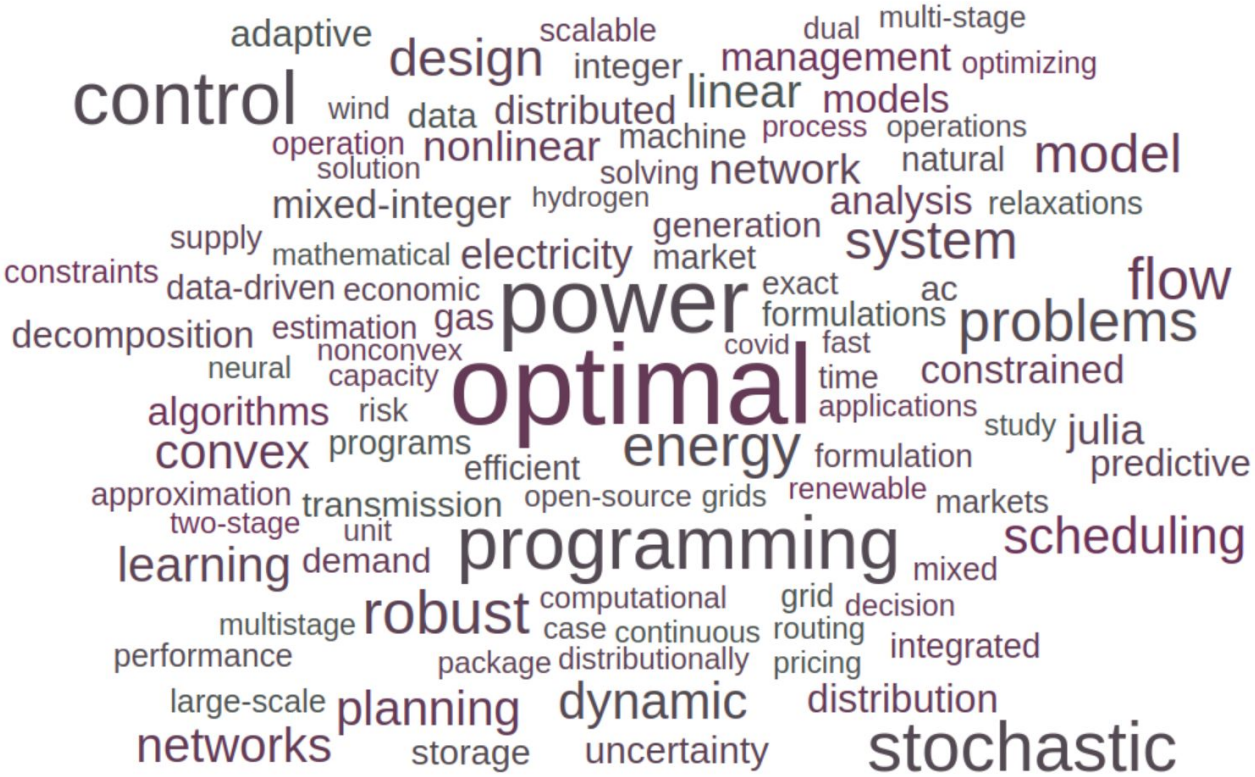
1,174
pull requests opened

52
unique contributors

373
issues opened

Who's citing JuMP?

~1800 citations to the 3 JuMP papers (Semantic Scholar)



So, what's new in JuMP?

JuMP 1.0 released March 2022. Since then:

- Improved nonlinear support
- Constraint programming
- Multiobjective
- Generic numbers
- Time to first solve
- Improved documentation
- Parameters

Improving nonlinear programming support in JuMP

https://jump.dev/JuMP.jl/stable/manual/nonlinear/

```
using JuMP
model = Model()
@variable(model, x[1:2])
@objective(model, Min, x[2]^3 * sin(x[1])^x[2])
my_func(y) = sum(2^y[1] .+ exp.(y))
@expression(model, expr, 2 * my_func(x))
@constraint(model, expr <= 100)
@constraint(model, sqrt(x' * x) <= 1)
```

Nonlinear complementarity

https://jump.dev/JuMP.jl/stable/tutorials/nonlinear/complementarity/

```
using JuMP, PATHSolver
model = Model(PATHSolver.Optimizer)
@variable(model, 0 <= x[1:2] <= 1, start = 0.5)
# To add a constraint of the form `F(x) ⊥ x` do
@constraint(model, x[2]^3 - x[1] ⊥ x[1])
@constraint(model, 1 - exp(x[1]) ⊥ x[2])
optimize!(model)
value.(x)
```

Complex number support

https://jump.dev/JuMP.jl/stable/manual/complex/

```
using JuMP
model = Model()
@variable(model, x in ComplexPlane())
#     real(x) + imag(x) im

@variable(model, y[1:2, 1:2] in HermitianPSDCone())
#     2×2 LinearAlgebra.Hermitian{...}:
#      real(y[1,1])                        real(y[1,2])+imag(y[1,2])*im
#      real(y[1,2])-imag(y[1,2])*im        real(y[2,2])
```

Constraint programming

https://jump.dev/JuMP.jl/stable/tutorials/linear/constraint_programming/

```
using JuMP, MiniZinc
model = Model(() -> MiniZinc.Optimizer{Float64}("highs"))
@variable(model, 1 <= x[1:3] <= 3, Int)
@variable(model, 0 <= z <= 1, Bin)
@constraint(model, x in MOI.AllDifferent(3))
@constraint(model, z <--> {x[1] == 1.0})
@objective(model, Max, sum(i * x[i] for i in 1:3))
optimize!(model)
value.(x)
```

Boolean SAT

https://jump.dev/JuMP.jl/stable/manual/constraints/#Boolean-constraints

```
using JuMP, MiniZinc
model = GenericModel{Bool}(() ->
    MiniZinc.Optimizer{Bool}("chuffed"))
@variable(model, x[1:2])
@constraint(model, x[1] || x[2] := true)
@constraint(model, x[1] && x[2] := false)
optimize!(model)
value.(x)
```

Multi-objective support

https://jump.dev/JuMP.jl/stable/tutorials/linear/multi_objective_examples/

```
using JuMP, HiGHS
import MultiObjectiveAlgorithms as MOA
model = Model(() -> MOA.Optimizer(HiGHS.Optimizer))
set_attribute(model, MOA.Algorithm(), MOA.Dichotomy())
@variable(model, 0 <= x[1:2] <= 3)
@objective(model, Min, [3x[1] + x[2], -x[1] - 2x[2]])
@constraint(model, 3x[1] - x[2] <= 6)
optimize!(model)
pareto_frontier = [
    value.(x; result = i) for i in 1:result_count(model)
]
```

Generic number support

https://jump.dev/JuMP.jl/stable/tutorials/conic/arbitrary_precision/

```
using JuMP, CDDLib
model = GenericModel{Rational{BigInt}}(
    CDDLib.Optimizer{Rational{BigInt}},
)
@variable(model, 1 // 7 <= x[1:2] <= 2 // 3)
@constraint(model, c1, (2 // 1) * x[1] + x[2] <= 1)
@constraint(model, c2, x[1] + 3x[2] <= 9 // 4)
@objective(model, Max, sum(x))
optimize!(model)
value.(x)  # Returns [1 // 6, 2 // 3]
```

Generic number support

https://jump.dev/JuMP.jl/stable/tutorials/conic/arbitrary_precision/

```
using JuMP, Clarabel
model = GenericModel{BigFloat}(
    Clarabel.Optimizer{BigFloat},
)
@variable(model, x[1:2, 1:2] in PSDCone())
@variable(model, t)
y = rand(2, 2)
@constraint(model, [t; vec(x .- y)] in SecondOrderCone())
@objective(model, Min, t)
optimize!(model)
value.(x)  # Returns Vector{BigFloat}
```

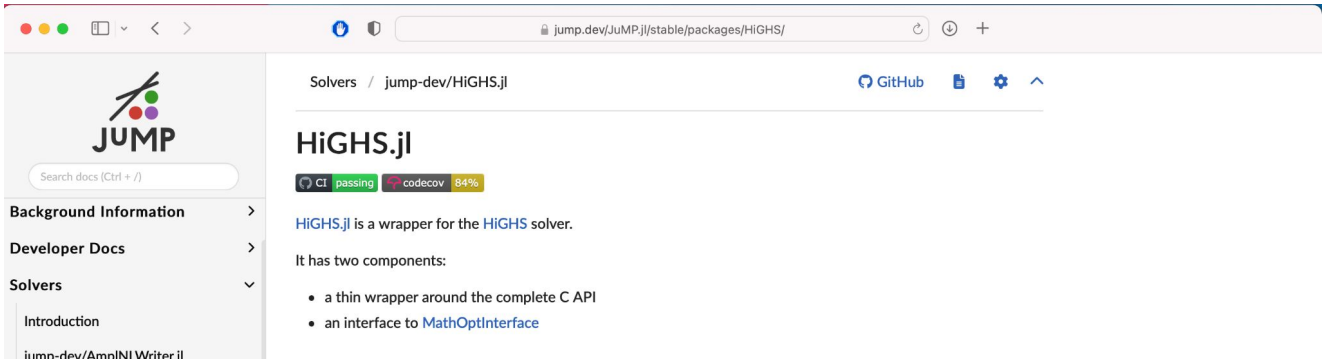
Time-To-First-Solve

```
% julia +1.6 bench.jl
VERSION = v"1.6.7"
  5.689157 seconds
  9.472270 seconds
```

```
% julia +1.9 bench.jl
VERSION = v"1.9.3"
  4.332634 seconds
  0.190987 seconds
```

```
@show VERSION
@time using JuMP, HiGHS
@time begin
    model = Model(HiGHS.Optimizer)
    set_silent(model)
    @variable(model, x >= 0)
    @variable(model, 0 <= y <= 3)
    @objective(model, Min, 12x + 20y)
    @constraint(model, c1, 6x + 8y >= 100)
    @constraint(model, c2, 7x + 12y >=
120)
    optimize!(model)
end
```

Improved documentation



- New set of tutorials
- Integrated solver READMEs
- Many misc. improvements
- PDF is 1200+ pages

Parameters

```
using JuMP
model = Model()
@variable(model, x in Parameter(2))
@variable(model, y[1:2])
@constraint(model, sum(y) >= x)
```

Use with extensions like ParametricOptInterface.jl

Also

- Convex.jl backend rewrite
- Gurobi automated install
- New solvers/interfaces:
  - MINOTAUR
  - PolyJuMP.QCQP
  - Octeract
  - MiniZinc
  - Percival
  - Manopt
  - SDPLR
  - Optim
  - MAiNGO
  - DSDP

Upcoming roadmap items

- Nonlinear expressions with vector inputs and outputs
- First-class support for units
  - `@variable(model, x, u"m/s")`

JuMP-inspired packages

I’d love to use JuMP but I’m coding in ___

- C++/Python: MathOpt (Google OR Tools)
- Python: PyOptInterface
- R: ompr

Great!

Thank you!

Go to jump.dev for more information