Introducción a la optimización en el módulo JuMP de Julia

Leonardo Rivera C.

Escuela de Ingeniería Industrial - Universidad del Valle

Motivación

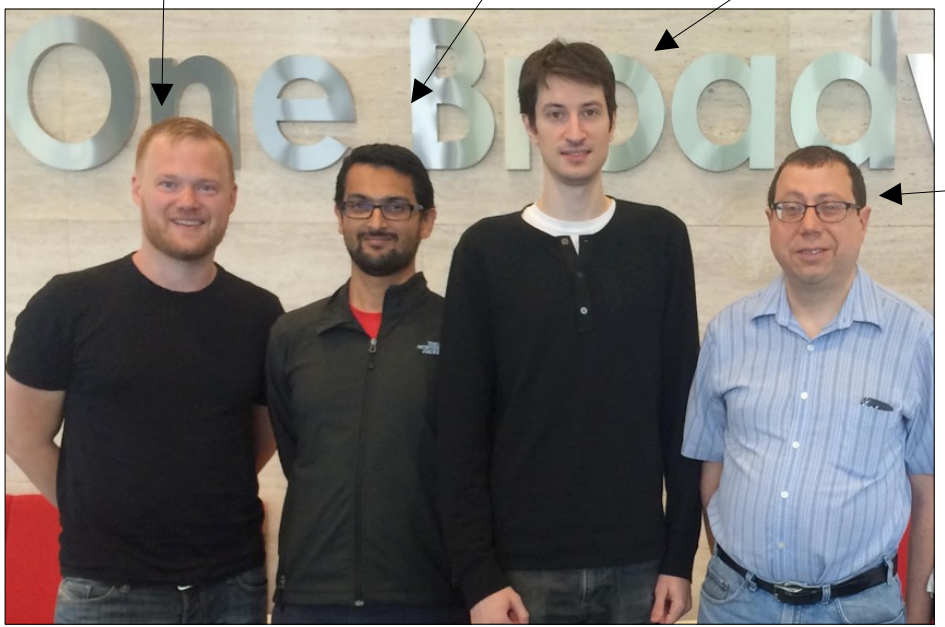
Es más fácil programar en un lenguaje de alto nivel (Python, R, Matlab)

Me gustaría que mi programa fuera rápido (C, C++, Fortran)

julia

Historia

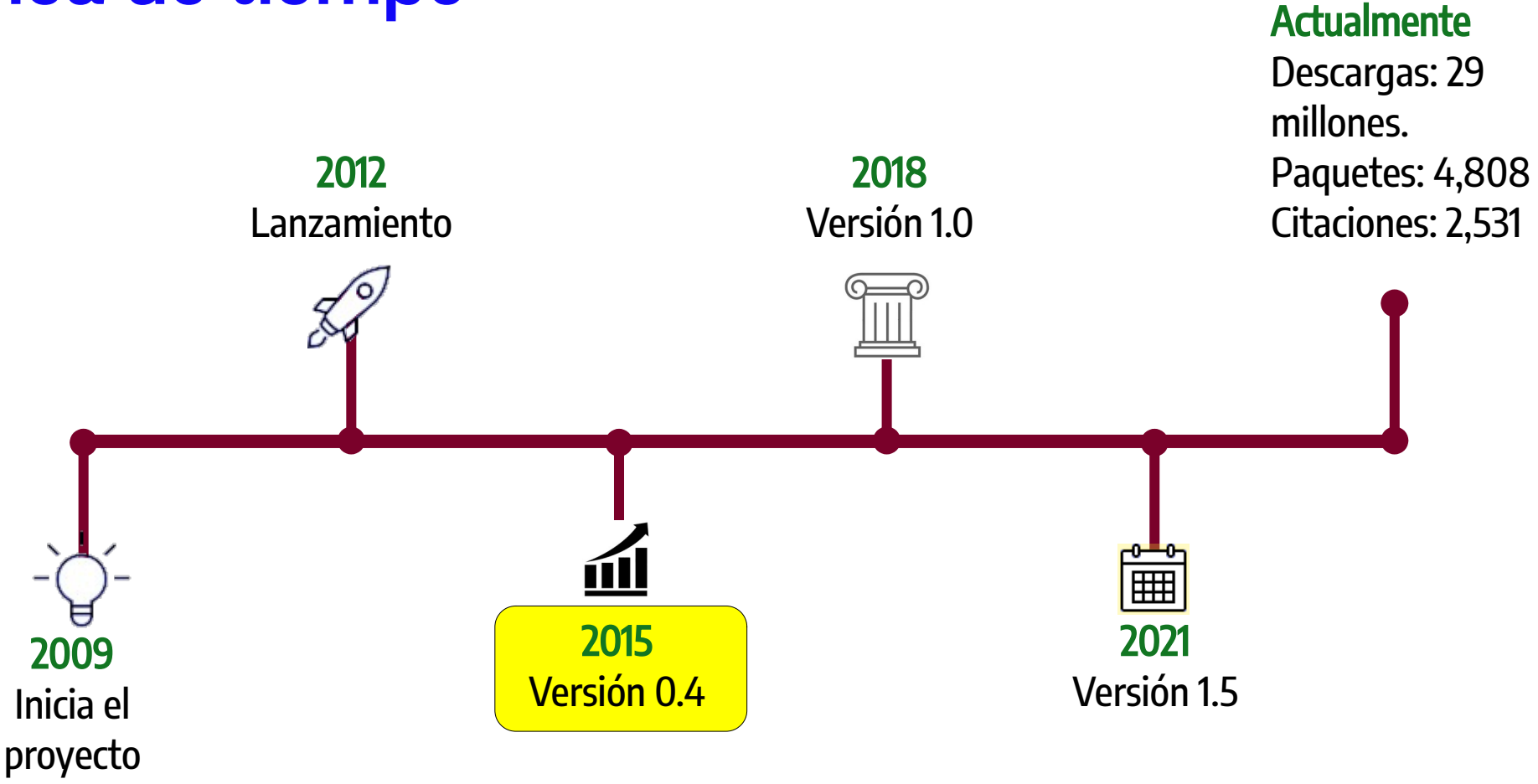
Karpinsky Shah Bezanson



Edelman

2009

Línea de tiempo



Casos de éxito

https://juliacomputing.com/case-studies/

UNITED THERAPEUTICS Pharmaceutical

Pharmaceutical Modeling

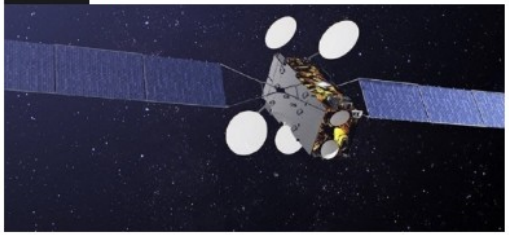
United Therapeutics uses JuliaHub to build a computational model of the lung to develop treatments for rare diseases, including diseases affecting the lungs

AVIVA Insurance

Solvency II Compliance

One of Europe's largest insurers is using Julia for Solvency II compliance

INPE Space



Planning Space Missions

The Brazilian National Institute for Space Research (INPE) is the Brazilian government's research institute for planning space missions

ASTRAZENECA Pharmaceutical

Predicting Toxicity

AstraZeneca and Prioris.ai researchers use Julia, Flux.jl and Turing.jl to predict toxicity with a Bayesian neural network

ATTESON RESEARCH Quantitative Finance

Financial Modeling

Atteson Research uses Julia for faster, better financial modeling

PFIZER Pharmaceutical

Pharmaceutical Development

Pfizer uses Julia to accelerate simulations of new therapies for metabolic diseases up to 175x

Casos de éxito

https://juliacomputing.com/case-studies/

BOSTON PUBLIC SCHOOLS Transportation

Optimizing Bus Routes and Times

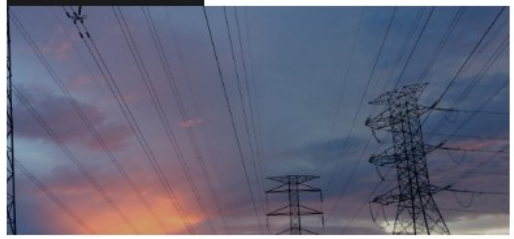
Optimization in Julia helps Boston schools eliminate up to 200 school buses, save up to $18 million and lets students get more sleep

NERSC Astronomy

Parallel Supercomputing for Astronomy

Researchers use Julia on a NERSC supercomputer (650,000 cores) to speed astronomical image

FUGRO ROAMES Machine Learning



Protecting the Electrical Grid

Fugro Roames engineers use machine learning in Julia to identify network failures and potential failures 100x faster

NY FEDERAL RESERVE Central Banking

Macroeconomic Modeling

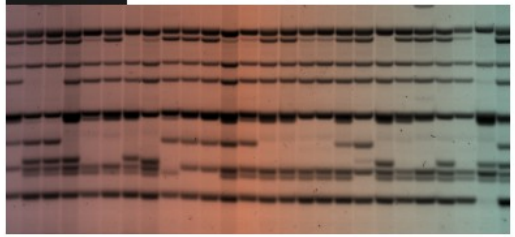
The Federal Reserve Bank of New York publishes its trademark Dynamic Stochastic General Equilibrium models in Julia

PSR Energy

Energy Analytics and Optimization

PSR uses Julia for energy market simulation, analytics and planning

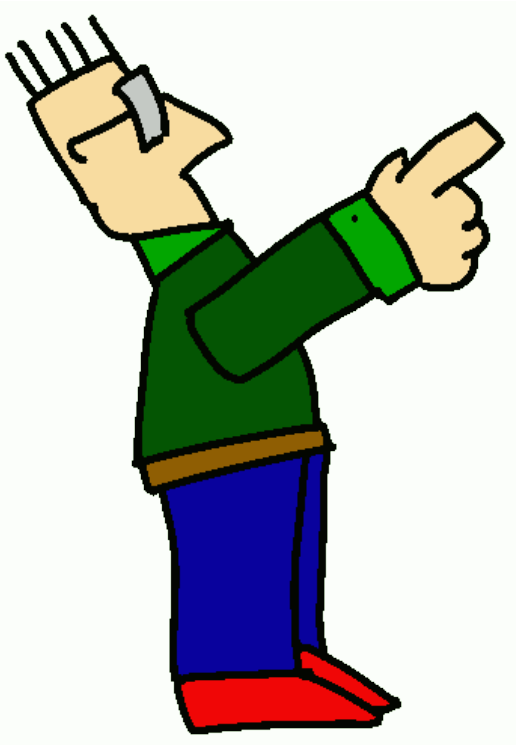
SCIENCE Genetic Diversity



Mapping Global Genetic Diversity

Researchers map global genetic diversity using Julia

Optimización en Julia



JUMP

- Gratuito
- Fácil sintaxis
- Sin límites de tamaño
- ¿Rápido?

Primer ejemplo (1)

Cargamos los módulos

Creamos el modelo

Agregamos las variables

Función Objetivo

Restricciones

```
using JuMP
using GLPK
using MathOptInterface

inicio = time()

m = Model(with_optimizer(GLPK.Optimizer))

@variable(m, X1 >= 0, base_name = "Prod_1")
@variable(m, X2 >= 0, base_name = "Prod_2")

@objective(m, Max, 3X1 + 5X2)

@constraint(m, depto1, X1 <= 4)
@constraint(m, depto2, 2X2 <= 12)
@constraint(m, depto3, 3X1 + 2X2 <= 18)
```

Primer ejemplo (2)

Imprimimos el modelo

Resolvemos

Imprimimos resultados

```
print(m)

status = optimize!(m)

println("Utilidad total  : ", JuMP.objective_value(m))
println(X1, "           : ", JuMP.value(X1))
println(X2, "           : ", JuMP.value(X2))

final = time()

println("El tiempo de ejecución fue de ", (final - inicio), "
segundos")
```

Modelo Algebraico (1)

Cargamos los módulos

```
using JuMP
using GLPK
using MathOptInterface

inicio = time()

DEPTOS = ["DP1", "DP2", "DP3"]
PRODS = ["Prod1", "Prod2"]

dp = size(DEPTOS)[1]
pd = size(PRODS)[1]

rec_depto = [4, 12, 18]
ut_prod = [3, 5]
t_depto = [1 0; 0 2; 3 2]

m = Model(with_optimizer(GLPK.Optimizer))
```

Datos indexados

Creamos el modelo

Modelo Algebraico (2)

x_i , $i \in PROD$

$$Max\ z = \sum_{i \in PROD} (x_i * ut\_prod_i)$$

```
@variable(m, X[1:pd] >= 0)

@objective(m, Max, sum(X[i]*ut_prod[i] for i in 1:pd))

@constraint(m, capdep[j = 1:dp], sum(X[i]*t_depto[j,i] for i in 1:pd) <= rec_depto[j])
```

$\forall\ j \in DEP$

$$\sum_{i \in PROD} (x_i * t\_depto_{i,j}) \leq rec\_depto_j$$

Modelo Algebraico (3)

Imprimimos

```
print(m)
```

Resolvemos

```
status = optimize!(m)
```

Imprimimos resultados

```
println("Utilidad total  : ", JuMP.objective_value(m))

for i in 1:pd
    println(X[i],  "           : ", JuMP.value(X[i]))
end

final = time()

println("El tiempo de ejecución fue de ", (final - inicio), "
segundos")
```

Nuevo modelo

La empresa **Jacuzzis Rivera (JR)** fabrica tres productos: Tina Básica (X1), Tina Chévere (X2) y Tina De Lujo (X3). Para el siguiente mes tiene disponibles cantidades limitadas de los recursos que usa (bombas de agua, horas de mano de obra y pies de tubería). Deseamos fabricar la cantidad de unidades de cada producto que maximice la utilidad total de la empresa.

Utilidades

Disponibles

```
MAX Z = 350 X1 + 300 X2 + 320 X3
Subject to:
          1 X1 +   1 X2 +   1 X3 ≤   200   Bombas
          9 X1 +   6 X2 +   8 X3 ≤ 1,566   Horas MdO
         12 X1 +  16 X2 +  13 X3 ≤ 2,880   Tubería
```

Consumos

Lectura de datos externos

Consumos.txt

```
1 1 1
9 6 8
12 16 13
```

```
julia> using DelimitedFiles

julia> h = readdlm("Consumos.txt")
3×3 Array{Float64,2}:
  1.0   1.0   1.0
  9.0   6.0   8.0
 12.0  16.0  13.0
```

Lectura de datos

```
julia> h[1,3]
1.0

julia> h[3,2]
16.0
```

Uso del parámetro:

`h[fila,columna]`

Índices con base 1

Análisis de Sensibilidad (1)

Costos Reducidos

```
# IMPRESIÓN DE COSTOS REDUCIDOS
println("\nCOSTOS REDUCIDOS")
for i in 1:prod
    println("X[",i,"]:  ", round(reduced_cost(X[i]),
digits=2))
end
```

Comando Variable

Precios Sombra

```
# IMPRESIÓN DE PRECIOS SOMBRA
println("\nPRECIOS SOMBRA")
for j in 1:rec
    println(recursos[j], "   ",
round(shadow_price(caprec[j]), digits=2))
end
```

Comando Restricción

Generación del reporte

Aumento y disminución permisible en los coef que las variables tienen en la F.O.

Impresión de rangos

```
# GENERACIÓN DEL REPORTE DE SENSIBILIDAD
sens_rep = lp_sensitivity_report(m)

# RANGOS DE VALIDEZ DE LOS COEF-OBJETIVO
rango_vars = zeros(prod,2)

for i in 1:prod
    rango_vars[i,1], rango_vars[i,2] = sens_rep[X[i]]
end

println("\nRangos de los coef en la función objetivo")
println("en los que la solución óptima no cambia\n")
println("              Min       Max")
println("           -------   -------")

for i in 1:prod
    minp = ut_prod[i] + round(rango_vars[i,1], digits=2)
    maxp = ut_prod[i] + round(rango_vars[i,2], digits=2)
    println("Producto ",i,"   ", minp, "      ", maxp)
end
```

Diccionario

An. de Sens. (3)

Reporte

Restricción

Aumento y disminución del lado derecho permisibles

```
# RANGOS DE VALIDEZ DE L0S PRECIOS SOMBRA
rango_rec = zeros(rec,2)

for j in 1:rec
    rango_rec[j,1], rango_rec[j,2] = sens_rep[caprec[j]]
end

println("\nRangos de los valores del lado derecho")
println("en los que son válidos los precios sombra\n")
println("           Min         Max")
println("         -------    -------")
```

Impresión de rangos de validez de los precios sombra

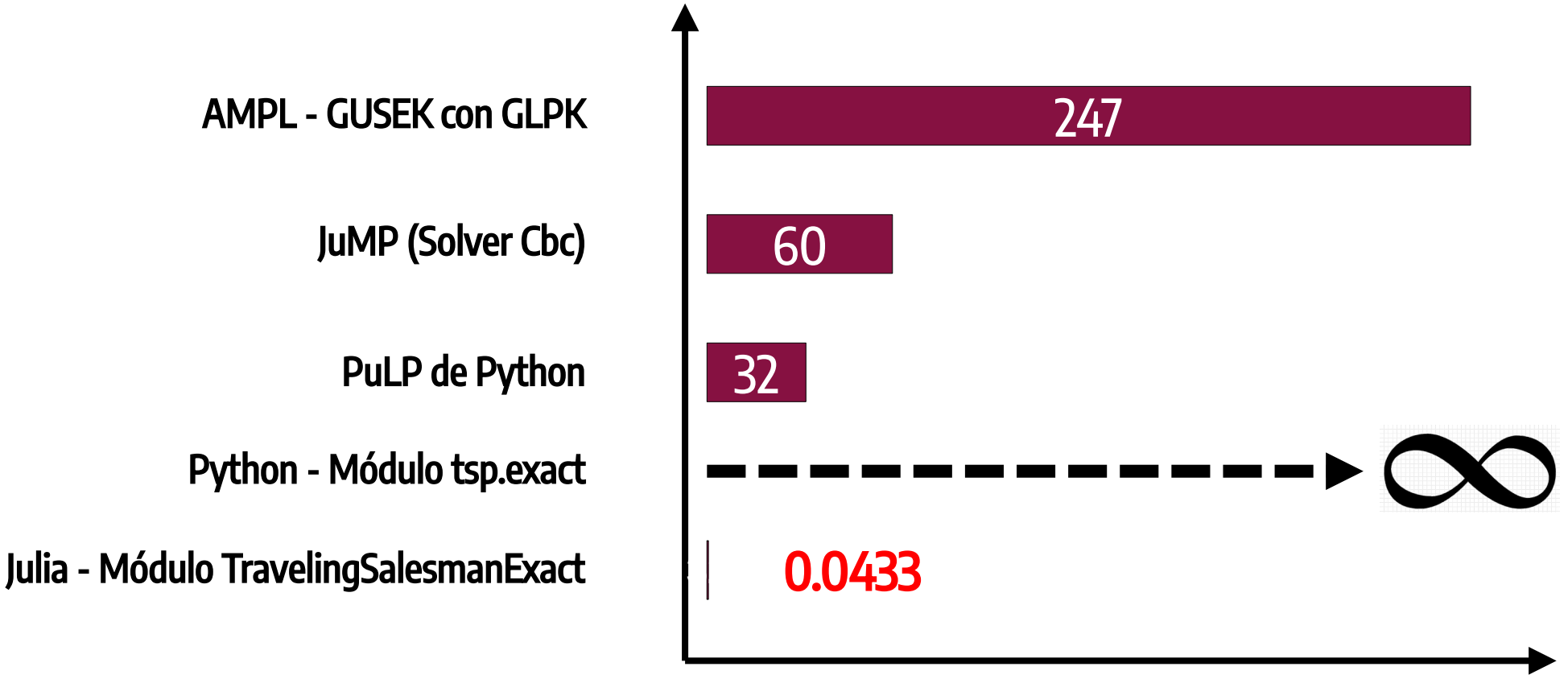
```
for j in 1:rec
    minr = disp_rec[j] + round(rango_rec[j,1], digits=2)
    maxr = disp_rec[j] + round(rango_rec[j,2], digits=2)
    println(recursos[j], "   ", minr, "      ", maxr)
end
```

An. de Sens. (4)

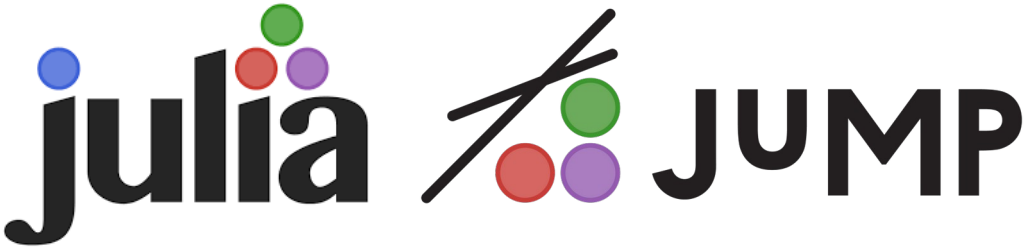
Consulta del lado izquierdo de la restricción


```
# HOLGURAS (Recursos sobrantes en las restricciones)

slacks = disp_rec .- value.(caprec)
println("\nLas holguras de los recursos son:")

for i in 1:length(slacks)
    println(recursos[i], ": ", slacks[i])
end
```

Comparación de desempeño – TSP 38 (Djibouti)



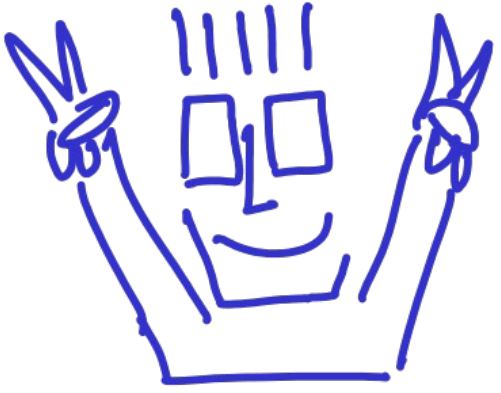
A manera de cierre:



Construimos varios modelos sencillos en JuMP

Hicimos un análisis de sensibilidad

Comparamos el desempeño de varios módulos en un ejemplo de TSP



leonardo.rivera.c@correounivalle.edu.co