Optimization and mathematical programming in Julia with applications to spatial data

Przemysław Szufel
https://szufel.pl/

Basics…

Linear optimization

```
using JuMP, HiGHS
m = Model(optimizer_with_attributes(HiGHS.Optimizer))
@variable(m,      x₁ >= 0)
@variable(m,      x₂ >= 0)
@objective(m,     Min, 50x₁ + 70x₂)
@constraint(m,    200x₁ + 2000x₂ >= 9000     )
@constraint(m,    100x₁ +   30x₂ >=  300     )
@constraint(m,      9x₁   +   11x₂ >=   60   )
optimize!(m)
JuMP.value.([x₁,x₂])
```

Note – how to type indexes in Julia

- julia> x
- julia> x_
- julia> x_1
- julia> x_1***<TAB>***
- julia> x_1

… and Integer programming

```julia
using JuMP, HiGHS
m = Model(optimizer_with_attributes(HiGHS.Optimizer))
@variable(m, x₁ >= 0, Int)
@variable(m, x₂ >= 0)
@objective(m, Min, 50x₁ + 70x₂)
@constraint(m, 200x₁ + 2000x₂ >= 9000)
@constraint(m, 100x₁ +   30x₂ >=  300)
@constraint(m, 9x₁   +   11x₂ >=   60)
optimize!(m)
```

How it works - metaprogramming

```
julia> code = Meta.parse("x=5")
:(x = 5)

julia> dump(code)
Expr
  head: Symbol =
  args: Array{Any}((2,))
    1: Symbol x
    2: Int64 5

julia> eval(code)
5

julia> x
5
```

Macros – hello world...

```
macro sayhello(name)
    return :( println("Hello, ", $name) )
end

julia> macroexpand(Main,:(@sayhello("aa")))
:((Main.println)("Hello, ", "aa"))

julia> @sayhello "world!"
Hello, world!
```

Macro @variable

```
julia> @macroexpand @variable(m, x₁ >= 0)
quote
    (JuMP.validmodel)(m, :m)
    begin
        #1###361 = begin
                let
                    #1###361 = (JuMP.constructvariable!)(m, getfield(JuMP, Symbol("#_error#107")){Tuple{Symbol,Expr}}((:m, :(x₁ >= 0))), 0, Inf, :Default, (JuMP.string)(:x₁), NaN)
                    #1###361
                end
            end
        (JuMP.registervar)(m, :x₁, #1###361)
        x₁ = #1###361
    end
end
```

Some of JuMP Solvers (over 40 as of today)

Solver	Julia Package	License	LP	SOCP	MILP	NLP	MINLP	SDP
Artelys Knitro	KNITRO.jl	Comm.				X	X	
BARON	BARON.jl	Comm.				X	X	
Bonmin	AmplNLWriter.jl CoinOptServices.jl	EPL	X		X	X	X	
Cbc	**Cbc.jl**	**EPL**			**X**			
Clp	Clp.jl	EPL	X					
Couenne	AmplNLWriter.jl CoinOptServices.jl	EPL	X		X	X	X	
CPLEX	**CPLEX.jl**	Comm.	X	X	X			
ECOS	ECOS.jl	GPL	X	X				
FICO Xpress	Xpress.jl	Comm.	X	X	X			
HiGHS	HiGHSMathProgInterface	GPL	X		X			
Gurobi	**Gurobi.jl**	Comm.	X	X	X			
Ipopt	**Ipopt.jl**	**EPL**	X			X		
MOSEK	Mosek.jl	Comm.	X	X	X	X		X
NLopt	NLopt.jl	LGPL				X		

JuMP
Transportation of good among branches

Use case scenario

The Subway restaurant chain in Las Vegas has a total of 118 restaurants in different parts of the city.

18 restaurants have adjacent huge product warehouses that keep ingredients cool and fresh, moreover fresh vegetables are delivered only to those warehouses (rather than to every restaurant) daily at 3am.

Subway has signed a contract with a transportation agency and is billed by the multiple of the weight of transported goods and the distance.

Knowing the amount of available stock at each warehouse and the expected demand at each restaurant (measured in kg), the company needs to decide how the goods should be distributed among warehouses.

Transportation problem statement

- Variables
  - x_{ij} – number of units transported for i-th supplier to j-th requester
  - c_{ij} – unit transportation cost between i-th supplier to j-th requester
- Cost function C

$$C = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

- Constraints:

  suppliers have maximum capacity S_i

$$\sum_{j=1}^{n} x_{ij} \leq S_i$$

  demand D_j must be met

$$\sum_{i=1}^{m} x_{ij} \geq D_j$$

Implementation in JuMP

```
m = Model(optimizer_with_attributes(HiGHS.Optimizer));
@variable(m, x[i=1:S, j=1:D])
@objective(m, Min, sum( x[i, j]*distance_mx[i, j] for i=1:S, j=1:D))
@constraint(m, x .>= 0)
for j=1:D
   @constraint(m, sum( x[i, j] for i=1:S) >= demand[j]   )
end
for i=1:S
   @constraint(m, sum( x[i, j] for j=1:D) <= supply[i]   )
end
optimize!(m)
termination_status(m)
```

Use case scenario

A tourist in San Fracisco and plans to visit all McDonald’s restaurant in one day

Let’s help him!

Spatial data
OpenStreetMap - https://www.openstreetmap.org

- Open project – “Wikipedia for maps”
- Lots, lots of data
  - Roads, Buildings, trees, …
  - Transportation systems
  - Point-of-interests (POIs) – businesses, restaurants, schools, universities…
- Formats: XML, PBF
- Multilayered structure
  - Nodes (points) 🡐 <tag/>
  - Ways (lines, shapes) 🡐 <tag/>
  - Relations (wider concepts) 🡐 <tag/>

```
<osm>
  <bounds minlat="42.3609500" minlon="-71.0914900" maxlat="42.3621500" maxlon="-71.0898000"/>
  <node id="61317286" lat="42.3611637" lon="-71.0927647"/>
  <node id="61317287" lat="42.3607193" lon="-71.0937014"/>
  <node id="6898815038" lat="42.3608365" lon="-71.0894651">
    <tag k="entrance" v="yes"/>
  </node>
  ....
  <way id="17660188">
    <nd ref="182893079"/>
    <nd ref="182893081"/>
    <tag k="addr:city" v="Cambridge"/>
    <tag k="addr:housenumber" v="43"/>
    ....
   </way>
...
 <relation id="1590059">
    <member type="node" ref="9124611329" role=""/>
    <member type="way" ref="426493700" role=""/>
    <member type="way" ref="8605061" role=""/>

    <tag k="network:wikipedia" v="en:Massachusetts Bay Transp
    <tag k="operator" v="Massachusetts Bay Transportation Aut
    <tag k="public_transport:version" v="2"/>
    <tag k="ref" v="CT2"/>
    <tag k="route" v="bus"/>
    <tag k="to" v="Ruggles Station"/>
    <tag k="type" v="route"/>
  </relation>
```


Libraries for OSM data

OpenStreetMapX.jl – mainly oriented on road system
- Road system extracted as a directed graph (Graphs.jl) along with separate metadata
- Supports routing, road classes, vehicle speeds etc.
- Spatial algebra (ENU, LLA, ECEF), overlap with Geodesy.jl

OpenStreetMapXPlots.jl
- Plotting maps with Plots.jl and PyPlot.jl .

OSMtoolset.jl – https://github.com/pszufe/OSMToolset.jl
- Spatial indexes on maps
- Mass extraction of points-of-interests (POIs) from maps
- Tools for slicing/tiling large OSM *.xml files

(developed under grant National Science Centre, Poland 2021/41/B/HS4/03349)

```julia
using OpenStreetMapX, OpenStreetMapXPlot
m = get_map_data("boston.osm"); # or *.pbf
p = plotmap(m; width=900, height=600)
r = shortest_route(m, 436587028, 5235225267)[1]
addroute!(p,m,r; route_color="red");
```

OSMToolset.jl – point of interest extraction

```
julia> df[end-2:end,:]
3×5 DataFrame
 Row │ class      key        influence  range  values
     │ String15   String31   Int64      Int64  String
─────┼──────────────────────────────────────────────────────────────────────
   1 │ leisure    sport              5    800  fitness
   2 │ leisure    landuse            5   1500  recreation_ground,winter_sports
   3 │ leisure    tourism            5   1500  *
```

Throughput ~ 2GB/min

```
julia> poidf = find_poi("boston.osm"; attract_config=AttractivenessConfig(df))
2576×10 DataFrame
  Row │ elemtype  elemid    nodeid       lat      lon       key               value          class      influence  range
      │ Symbol    Int64     Int64        Float64  Float64   String            String         String     Int64      Int64
──────┼─────────────────────────────────────────────────────────────────────────────────────────────────────────────────────
    1 │ node      69480814     69480814  42.357   -71.0588  public_transport  stop_position  transport          5    300
    2 │ node      69482188     69482188  42.3599  -71.06    public_transport  stop_position  transport          5    300
    3 │ node      69482993     69482993  42.3525  -71.0549  public_transport  stop_position  transport          5    300
    4 │ node      69487423     69487423  42.3736  -71.0697  railway           station        transport         10    700
  ⋮   │    ⋮         ⋮           ⋮          ⋮        ⋮             ⋮                ⋮             ⋮           ⋮        ⋮
 2574 │ relation  14205406   9784109000  42.38    -71.0934  amenity           parking        parking            5    250
 2575 │ relation  14205408    327175969  42.38    -71.0956  amenity           parking        parking            5    250
 2576 │ relation  15704864  10800012568  42.3551  -71.1022  leisure           garden         leisure            5    500
                                                                                                       2569 rows omitted
```

McDonald’s in SF

```
In [48]: using DataFrames, OSMToolset
         config = DataFrame(key="brand", values="McDonald's")
         dfpoi = find_poi("SF.osm"; scrape_config=ScrapePOIConfig{NoneMetaPOI}(config))
```

Out[48]: 18×7 DataFrame

Row	elemtype	elemid	nodeid	lat	lon	key	value
	Symbol	Int64	Int64	Float64	Float64	String	String
1	node	358116917	358116917	37.7264	-122.476	brand	McDonald's
2	node	597382133	597382133	37.7066	-122.415	brand	McDonald's
3	node	1229920544	1229920544	37.7131	-122.445	brand	McDonald's
4	node	2365742146	2365742146	37.789	-122.401	brand	McDonald's
5	node	3455025884	3455025884	37.6522	-122.491	brand	McDonald's
6	node	4626983989	4626983989	37.7892	-122.408	brand	McDonald's
7	node	6959355927	6959355927	37.644	-122.404	brand	McDonald's
8	node	9980865058	9980865058	37.6438	-122.454	brand	McDonald's
9	node	11338930625	11338930625	37.6747	-122.47	brand	McDonald's
10	way	143811393	1573722786	37.669	-122.47	brand	McDonald's
11	way	159024983	1711296799	37.7236	-122.455	brand	McDonald's
12	way	231047897	2394660225	37.752	-122.418	brand	McDonald's
13	way	256462436	2621272506	37.7653	-122.408	brand	McDonald's

JuMP
Travelling salesman problem

Use case scenario

A tourist in San Fracisco and plans to visit all McDonald’s restaurant in one day

Let’s help him!

Traveling salesman problem (TSP)

- Variables:
  - c_{ft} – cost of travel from "f" to "t"
  - x_{ft} – binary variable indicating 1 when agent travels from "f" to "t"

$$\text{Min} \sum_{f=1}^{N} \sum_{t=1}^{N} c_{ft} x_{ft}$$

The mathematical problem framing based on: Julia Programming for Operations Research A Primer on Computing, Changhyun Kwon, 2018, see: http://opensourc.es/blog/mip-tsp

TSP

$$\text{Min} \sum_{f=1}^{N}\sum_{t=1}^{N} c_{ft} x_{ft}$$

Each city visited once

$$\sum_{t=1}^{N} x_{ft} = 1 \quad \forall f \in \{1, \ldots, N\}$$

$$\sum_{f=1}^{N} x_{ft} = 1 \quad \forall t \in \{1, \ldots, N\}$$

City cannot visit itself

$$x_{ff} = 0 \quad \forall f \in \{1, \ldots, N\}$$

Avoid two-city cycles

$$x_{ft} + x_{tf} <= 1 \quad \forall f, t \in \{1, \ldots, N\}$$

Other cycles: /dynamically add a constraint whenever a cycle occurs/

Variables:

- c_{ft} – cost of travel from "f" to "t"
- x_{ft} – binary variable indicating 1 when agent travels from "f" to "t"

For more details see: http://opensourc.es/blog/mip-tsp

JuMP implementation

```
m = Model(optimizer_with_attributes(HiGHS.Optimizer));
@variable(m, x[f=1:N, t=1:N], Bin)
@objective(m, Min, sum( x[i, j]*distance_mx[i,j] for i=1:N,j=1:N))
@constraint(m, notself[i=1:N], x[i, i] == 0)
@constraint(m, oneout[i=1:N], sum(x[i, 1:N]) == 1)
@constraint(m, onein[j=1:N], sum(x[1:N, j]) == 1)
for f=1:N, t=1:N
    @constraint(m, x[f, t]+x[t, f] <= 1)
end
```

Getting a cycle

```
function getcycle(m, N)
    x_val = getvalue(x)
    cycle_idx = Vector{Int}()
    push!(cycle_idx, 1)
    while true
        v, idx = findmax(x_val[cycle_idx[end], 1:N])
        if idx == cycle_idx[1]
            break
        else
            push!(cycle_idx, idx)
        end
    end
    cycle_idx
end
```

Adding a constraint…

```
function solved(m, cycle_idx, N)
    println("cycle_idx: ", cycle_idx)
    println("Length: ", length(cycle_idx))
    if length(cycle_idx) < N
        cc = @constraint(m, sum(x[cycle_idx,cycle_idx])
           <= length(cycle_idx)-1)
        println("added a constraint")
        return false
    end
    return true
end
```

Iterating over the model

```
while true
    status = solve(m)
    println(status)
    cycle_idx = getcycle(m, N)
    if solved(m, cycle_idx,N)
        break;
    end
end
```

Gurobi.jl

- Commercial software
- Free for academic use
- Integrates with JuMP via Gurobi.jl

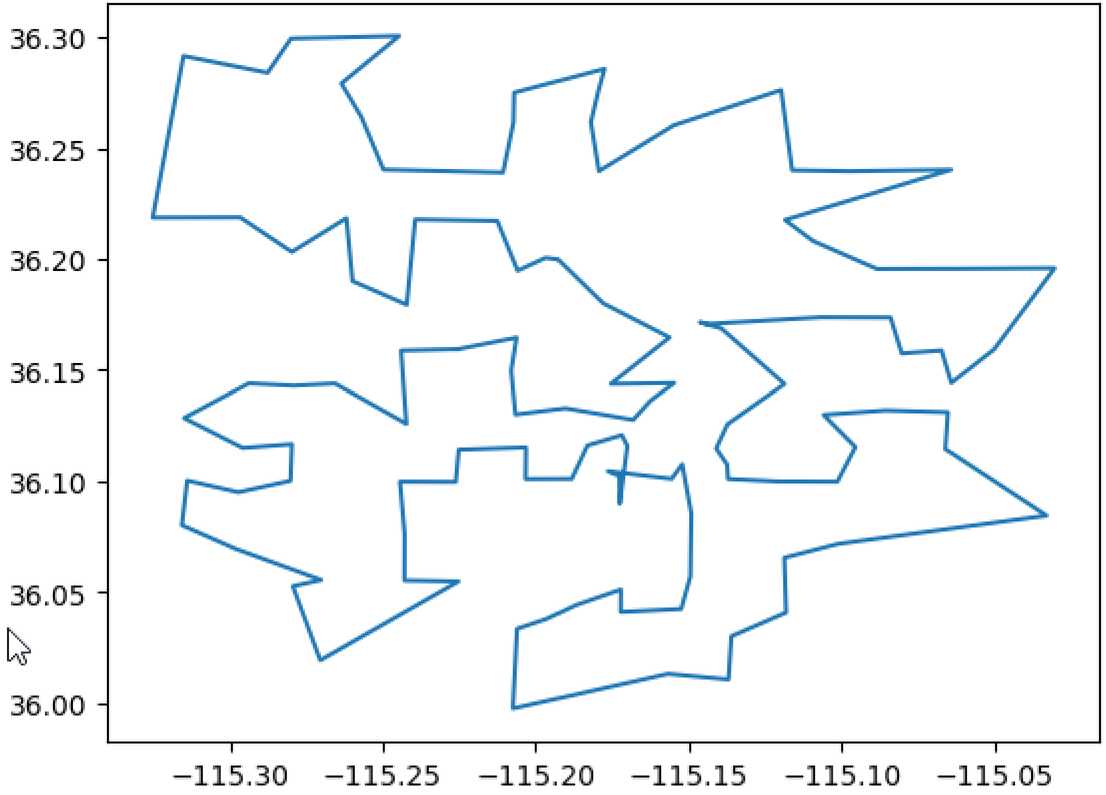
- Supports JuMP Lazy constraints (http://www.juliaopt.org/JuMP.jl/0.18/callbacks.html)

Gurobi callbacks

```
function getcycle(cb, N)
    x_val = callback_value.(Ref(cb), x)
    getcycle(x_val)
end
function callbackhandle(cb)
    cycle_idx = getcycle(cb, N)
    println("Callback! N= $N cycle_idx: ", cycle_idx)
    println("Length: ", length(cycle_idx))
    if length(cycle_idx) < N
        con = @build_constraint(sum(x[cycle_idx,cycle_idx]) <= length(cycle_idx)-1)
        MOI.submit(m, MOI.LazyConstraint(cb), con)
        println("added a lazy constraint")
    end
end
MOI.set(m, MOI.LazyConstraintCallback(), callbackhandle)
```

TravelingSalesmanHeuristics.jl

```
using TravelingSalesmanHeuristics
sol = TravelingSalesmanHeuristics.solve_tsp(
distance_mx,quality_factor =100)
```



More info:
http://evanfields.github.io/TravelingSalesmanHeuristics.jl/latest/heuristics.html

JuMP
Non-Linear Programming

Simple scenario

Estimate parameters of a quadratic form

$$\boldsymbol{y}(\boldsymbol{x}_i) = \boldsymbol{x}_i^T \begin{bmatrix} a & b/2 \\ b/2 & c \end{bmatrix} \boldsymbol{x}_i, \text{ where } \boldsymbol{x}_i = \begin{bmatrix} x_i^1 \\ x_i^2 \end{bmatrix}$$

for a vector of observed values ***y*** to minimize the observed error function

$$\sum_{i=1}^{N} (y(\boldsymbol{x}_i) - y_i)^2$$

Nonlinear optimization Julia

```
m = Model(optimizer_with_attributes(Ipopt.Optimizer));

@variable(m, aa[1:2,1:2])

function errs(aa)
   sum((y .- (x * aa ) .* x * [1;1]) .^ 2)
end

@objective(m, Min, errs(aa))

optimize!(m)
```

Use case scenario
(source: Hart et al, Pyomo-optimization modeling in python, 2017)

Simulate dynamics of disease outbreak in a small community of 300 individuals (e.g. children at school)

Three possible states of a patient:

- susceptible (S)
- infected (I)
- recovered (R)

<u>Infection spread model :</u>

- N – population size
- α, β – model parameters

$$I_i = \frac{\beta I_{i-1}^{\alpha} S_{i-1}}{N}$$

$$S_i = S_{i-1} - I_i$$

Optimization problem for finding parameters α and β

S - susceptible

I - infected

N – population size

α, β – model parameters

SI - time indices {1,2,3,...}

C_i - known input (the actual number of infected patients)

$$\min \sum_{i \in SI} \left(\varepsilon_i^I\right)^2$$

$$I_i = \frac{\beta I_{i-1}^{\alpha} S_{i-1}}{N} \quad \forall\ i \in SI \setminus \{1\}$$

$$S_i = S_{i-1} - I_i \ \forall\, i\ \in SI \setminus \{1\}$$

$$C_i = I_i + \varepsilon_i^I$$

$$0 \leq I_i,\ Si \leq N$$

$$0.5 \leq \beta \leq 70$$

$$0.5 \leq \alpha \leq 1.5$$

Model implementation in JuMP

- Input data (disease dynamics)

```
obs_cases = vcat(1,2,4,8,15,27,44,58,55,32,12,3,1,zeros(13))
```

Full model specification in JuMP

```
m = Model(optimizer_with_attributes(Ipopt.Optimizer));
@variable(m, 0.5 <= α <= 1.5)
@variable(m, 0.05 <= β <= 70)
@variable(m, 0 <= I_[1:SI_max] <= N)
@variable(m, 0 <= S[1:SI_max]  <= N)
@variable(m, ε[1:SI_max])
@constraint(m, ε .== I_ .- obs_cases  )
@constraint(m, I_[1] == 1)
for i=2:SI_max
   @NLconstraint(m, I_[i] == β*(I_[i-1]^α)*S[i-1]/N)
end
@constraint(m, S[1] == N)
for i=2:SI_max
   @constraint(m, S[i] == S[i-1]-I_[i])
end
@NLobjective(m, Min, sum(ε[i]^2 for i in 1:SI_max))
```

JuMP
Non-Linear Programming
for estimation of model parameters

Simple scenario

Estimate parameters of a quadratic form

$$\boldsymbol{y}(\boldsymbol{x}_i) = \boldsymbol{x}_i^T \begin{bmatrix} a & b/2 \\ b/2 & c \end{bmatrix} \boldsymbol{x}_i, \text{ where } \boldsymbol{x}_i = \begin{bmatrix} x_i^1 \\ x_i^2 \end{bmatrix}$$

for a vector of observed values ***y*** to minimize the observed error function

$$\sum_{i=1}^{N} (y(\boldsymbol{x}_i) - y_i)^2$$

Nonlinear optimization Julia

```
m = Model(optimizer_with_attributes(Ipopt.Optimizer));

@variable(m, aa[1:2,1:2])

function errs(aa)
   sum((y .- (x * aa ) .* x * [1;1]) .^ 2)
end

@objective(m, Min, errs(aa))

optimize!(m)
```

Use case scenario
(source: Hart et al, Pyomo-optimization modeling in python, 2017)

Simulate dynamics of disease outbreak in a small community of 300 individuals (e.g. children at school)

Three possible states of a patient:

- susceptible (S)
- infected (I)
- recovered (R)

Infection spread model :

- N – population size
- α, β – model parameters

$$I_i = \frac{\beta I_{i-1}^{\alpha} S_{i-1}}{N}$$

$$S_i = S_{i-1} - I_i$$

Optimization problem for finding parameters α and β

S - susceptible

I - infected

N – population size

α, β – model parameters

SI - time indices {1,2,3,...}

C_i - known input (the actual number of infected patients)

$$\min \sum_{i \in SI} \left(\varepsilon_i^I\right)^2$$

$$I_i = \frac{\beta I_{i-1}^{\alpha} S_{i-1}}{N} \quad \forall\ i \in SI \setminus \{1\}$$

$$S_i = S_{i-1} - I_i \quad \forall\ i\ \in SI \setminus \{1\}$$

$$C_i = I_i + \varepsilon_i^I$$

$$0 \leq I_i,\ Si \leq N$$

$$0.5 \leq \beta \leq 70$$

$$0.5 \leq \alpha \leq 1.5$$

Model implementation in JuMP

- Input data (disease dynamics)

```
obs_cases = vcat(1,2,4,8,15,27,44,58,55,32,12,3,1,zeros(13))
```

Full model specification in JuMP

```
m = Model(optimizer_with_attributes(Ipopt.Optimizer));
@variable(m, 0.5 <= α <= 1.5)
@variable(m, 0.05 <= β <= 70)
@variable(m, 0 <= I_[1:SI_max] <= N)
@variable(m, 0 <= S[1:SI_max]  <= N)
@variable(m, ε[1:SI_max])
@constraint(m, ε .== I_ .- obs_cases  )
@constraint(m, I_[1] == 1)
for i=2:SI_max
   @NLconstraint(m, I_[i] == β*(I_[i-1]^α)*S[i-1]/N)
end
@constraint(m, S[1] == N)
for i=2:SI_max
   @constraint(m, S[i] == S[i-1]-I_[i])
end
@NLobjective(m, Min, sum(ε[i]^2 for i in 1:SI_max))
```