第 91 回 TechTrend Talk Series vol.5

Straightforward *modeling tools* for prescriptive *decision-making*

Jean-François Baffier

Internet Initiative Japan Research Lab

September 26th, 2023

1 Outline

1 Decision-Making and Optimality (history-ish)

> *[···] the principles of mathematics were the principles of all things.*
>
> ARISTOTLE, ***Metaphysics***, *350 BCE*

> *Light travels between points through the path of shortest length.*
>
> HERON OF ALEXANDRIA, *(10-75) CE*

Honeycomb ~~Conjecture~~ Theorem

A regular hexagonal grid or honeycomb has the least total perimeter of any subdivision of the plane into regions of equal area.
It was proven by T.C. HALES in 1999.

Nature optimizes! But slowly, over numerous iterations, through a trial and error process.

Figure: Worker Bees On Honeycomb – Jean BEAUFORT

1 Overview of a Decision-Making process

1 Alisu and Bob

Prescriptive Approach (*e.g.* Optimization)

- Focus: Determining the best course of action.
- Primary Tool: Mathematical models.
- Objective: Optimize a specific criterion.
- Example: Efficient resource allocation, route optimization.

Predictive Approach (*e.g.* Machine Learning)

- Focus: Making informed predictions based on data.
- Primary Tool: Machine learning models.
- Objective: Predict future outcomes or classify data.
- Example: Predicting customer behavior, image recognition.

1 Complementary Nature

Coexistence

- Prescriptive and Predictive approaches are not mutually exclusive.
- Often complement each other for holistic decision-making.

Synergy

- Optimization can use insights from predictive models.
- Machine Learning (predictive models) can use optimization to improve performance. (This is a very hot topic in AI)

- Some problems benefit from both approaches (hybrid models).
- Example: Using predictive analytics for demand forecasting and optimization for inventory management.

1 When to Use Prescriptive and Predictive Approaches

1. Consider Problem Characteristics
   - Complexity:
     - Use Prescriptive (Optimization) for complex decision-making problems where finding the best solution is critical.
   - Data-Driven:
     - Employ Predictive (Machine Learning) when historical data is abundant and patterns need to be extracted.
2. Data Availability and Quality
   - Data Availability:
     - If extensive data is accessible, consider Predictive approaches to leverage it.
   - Data Quality:
     - Ensure data quality; inaccurate data can lead to flawed predictions.
3. Decision Urgency
   - Immediate Decisions:
     - Use Prescriptive approaches for real-time or critical decisions with little room for error.
   - Future Planning:
     - Predictive approaches are valuable for long-term planning and trend analysis.
4. Objective Types
   - Objective Clarity:
     - When objectives are well-defined and measurable, Prescriptive methods are suitable.
   - Objective Uncertainty:
     - For objectives that are uncertain or evolving, Predictive approaches may be more adaptable.

2 Outline

2 Simple and fun: Sudoku

A simple variant of prescriptive problems are *Satisfaction Problems*. They are a class of problems where the goal is to find a solution that satisfies a set of constraints. Let us consider the following example: Sudoku.

Sudoku rules

- Each row, column and 3x3 subgrid must contain the numbers 1 to 9.
- Each number can only appear once in each row, column and subgrid.

Some numbers are already given as clues.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Figure: a Sudoku layout generated by the GNU program Su Doku Solver – Lawrence Leonard Gilbert

2 Simple and fun: Sudoku

A simple variant of prescriptive problems are *Satisfaction Problems*. They are a class of problems where the goal is to find a solution that satisfies a set of constraints. Let us consider the following example: Sudoku.

```julia
function sudoku(n)
    N = n^2
    m = JuMP.Model(CBLS.Optimizer)

    @variable(m, 1 ≤ X[1:N, 1:N] ≤ N, Int)

    for i in 1:N
        @constraint(m, X[i,:] in AllDifferent()) # rows
        @constraint(m, X[:,i] in AllDifferent()) # columns
    end
    for i in 0:(n-1), j in 0:(n-1)
        # blocks
        @constraint(m, vec(X[(i*n+1):(n*(i+1)), (j*n+1):(n*(j+1))]) in AllDifferent())
    end
    return m, X
end
```

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Figure: a Sudoku layout generated by the GNU program Su Doku Solver – Lawrence Leonard Gilbert

2 A bit of magic: Magic Square

Now, let us consider a classical mathematical problem called *Magic Square* (Japan has a strong history of solving Magic Square throughout history).

Magic Square rules

- Each number can only appear once.
- The sum of each row, column and diagonal must be equal.

Size 3×3 is the minimal non-trivial instance.

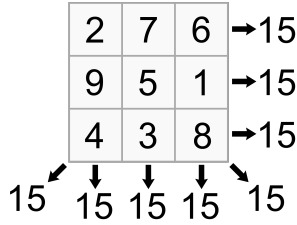


Figure: a 3×3 Magic Square instance with magic number 15 – Sam Ley (Phidauex)

2 A bit of magic: Magic Square

Now, let us consider a classical mathematical problem called *Magic Square* (Japan has a strong history of solving Magic Square throughout history).

```julia
function magic_square(n, ::Val{:JuMP})
    N = n^2
    model = JuMP.Model(CBLS.Optimizer)
    magic_constant = n * (N + 1) / 2

    @variable(model, 1 ≤ X[1:n, 1:n] ≤ N, Int)
    @constraint(model, vec(X) in AllDifferent())

    for i in 1:n
        @constraint(model, X[i,:] in SumEqualParam(magic_constant))
        @constraint(model, X[:,i] in SumEqualParam(magic_constant))
    end
    @constraint(model, [X[i,i] for i in 1:n] in SumEqualParam(magic_constant))
    @constraint(model, [X[i,n + 1 - i] for i in 1:n] in SumEqualParam(magic_constant))

    return model, X
end

"""
    magic_square(n; modeler = :JuMP)

Create a model for the magic square problem of order `n`. The `modeler` argument accepts
:JuMP (default), which refer to the solver the JuMP model.
"""
magic_square(n; modeler = :JuMP) = magic_square(n, Val(modeler))
```

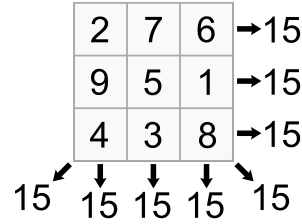


Figure: a 3×3 Magic Square instance with magic number 15 – Sam Ley (Phidauex)

3 Outline

3 A problem to rule them all: Golomb Ruler

A more advanced variant of prescriptive problems are *Optimization Problems*. They are a class of problems where the goal is to find a solution that optimizes a specific criterion. Let us consider the following example: Golomb Ruler.

Golomg Ruler description

- Given n marks, each pair of marks has a different distance.
- The objective is to minimize the size of the ruler, *i.e.* the distance between the first and last marks.

A perfect Golomb Ruler is a Golomb Ruler with the smallest possible size.

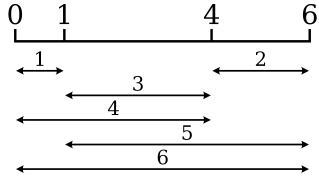


Figure: a perfect Golomb ruler with 4 marks – Khrishna Vedala

3 A problem to rule them all: Golomb Ruler

A more advanced variant of prescriptive problems are *Optimization Problems*. They are a class of problems where the goal is to find a solution that optimizes a specific criterion. Let us consider the following example: Golomb Ruler.

```
dist_different(X) = abs(X[1] - X[2]) ≠ abs(X[3] - X[4])

function golomb(n, L=n²)
    m = JuMP.Model(CBLS.Optimizer)

    @variable(m, 0 ≤ X[1:n] ≤ L, Int)

    @constraint(m, X in AllDifferent()) # different marks
    @constraint(m, X in Ordered()) # for output convenience, keep them ordered

    # No two pairs have the same length
    for i in 1:(n - 1), j in (i + 1):n, k in i:(n - 1), l in (k + 1):n
        (i, j) < (k, l) || continue
        @constraint(m, [X[i], X[j], X[k], X[l]] in Predicate(dist_different))
    end
    # Add objective
    @objective(m, Min, ScalarFunction(maximum))
    return m, X
end
```

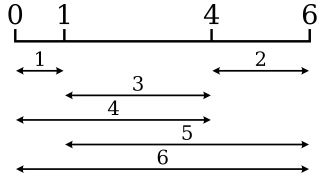
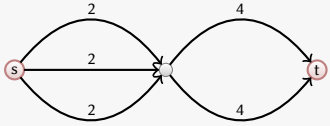


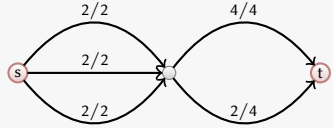
Figure: a perfect Golomb ruler with 4 marks – Khrishna Vedala

3 A personal favorite: Network flow problems

3 A personal favorite: Network flow problems

Maximum Flow problem

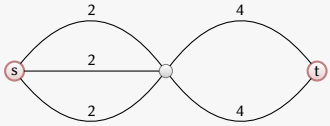


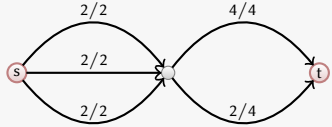


- Amount of flow (data/matter/anything) from a source s to a sink t
- Introduces in 1952 to study the soviet railway system
- Applications to various fields of research, among others:
  - Scheduling problems: course scheduling, task scheduling, transports ...
  - Matching problems: colors, pair of individuals, ...
  - Image Segmentation
- Can be solved efficiently (tractable $\sim$ polynomial algorithm)

3 Network Interdiction problems

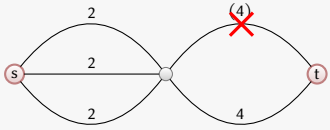
Classical max-flow

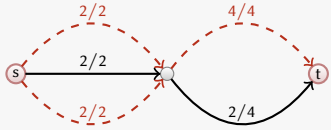




3 Network Interdiction problems

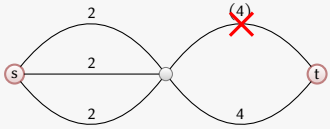
Classical max-flow

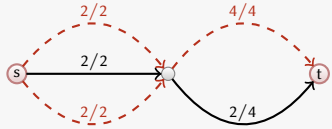




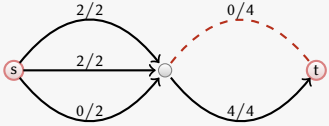
3 Network Interdiction problems

Classical max-flow

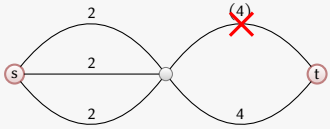


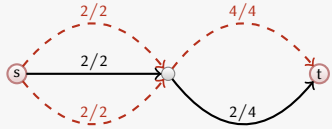


Network Interdiction

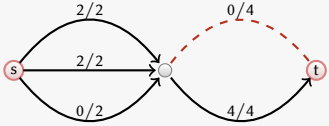


Classical max-flow

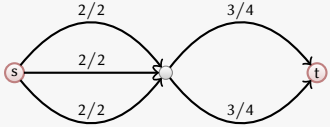




Network Interdiction



Adaptive Network Flow



3 Specialist approach for Adaptive Flow

$$
\left\{
\begin{array}{ll}
\max\limits_{f,\mu} & \sum\limits_{e\in E} f(e)(1-\mu(e)) \\
& f \in \mathcal{F} \\
& \mu \in \operatorname*{argmin}\limits_{\mu} \max\limits_{f'} \quad \sum\limits_{(s,i)\in E} f'(s,i) \\
& \qquad f'(e) \leqslant f(e)(1-\mu(e)), \quad \text{for all } e \in E \\
& \qquad \sum\limits_{i:(i,j)\in E} f'((i,j)) - \sum\limits_{k:(j,k)\in E} f'((j,k)) = 0, \\
& \qquad\qquad \text{for all } j \in V \setminus \{s,t\} \\
& \qquad f(e) \geqslant 0, \quad \text{for all } e \in E.
\end{array}
\right\}
$$

- Upper level is classical max flow problem with an objective function variation coming from the attacked links
- Lower Level is a Network Interdiction problem on a given flow

3 Specialist approach for Adaptive Flow

$$\max_{f} \sum_{e \in E} (f(e)\delta_{ij} - f(e)\nu_{ij})$$

$$f \in \mathcal{F}$$

$$\delta, \nu, \mu \in \min_{\mu,\delta,\sigma} \quad \sum_{e \in E} (f(e)\delta_{ij} - f(e)\nu_{ij})$$

$$\begin{aligned}
&\delta_{ij} + \sigma_j - \sigma_i \geqslant 0, && \text{for all } (i,j) \in E, i \neq s, j \neq t && (1)\\
&\delta_{sj} + \sigma_j \geqslant 1, && \text{for all } j \text{ such that } (s,j) \in E && (2)\\
&\delta_{it} - \sigma_i \geqslant 0, && \text{for all } i \text{ such that } (t,i) \in E && (3)\\
&\delta_{ij} \geqslant 0, && \text{for all } (i,j) \in E && (4)\\
&\nu_{ij} \leqslant \mu((i,j)), && \text{for all } (i,j) \in E && (5)\\
&\nu_{ij} \leqslant \delta_{ij}, && \text{for all } (i,j) \in E && (6)\\
&\nu_{ij} \geqslant 0, && \text{for all } (i,j) \in E && (7)\\
&\sum_{e \in E} \mu(e) = k && && (8)\\
&\mu(e) \in \{0,1\}, && \text{for all } (i,j) \in E && (9)
\end{aligned}$$

3 A simpler model

The maximum network flow problem is a dual case of the minimum cut problem. It is also true for the interdiction version.

```julia
function mincut(graph, source, sink, interdiction = 0)
    m = JuMP.Model(CBLS.Optimizer)
    n = size(graph, 1)
    separator = n + 1

    @variable(m, 0 ≤ X[1:separator] ≤ n, Int)

    @constraint(m, [X[source], X[separator], X[sink]] in Ordered())
    @constraint(m, X in AllDifferent())


    obj(x...) = o_mincut(graph, x...; interdiction)
    @objective(m, Min, ScalarFunction(obj))

    return m, X
end
```

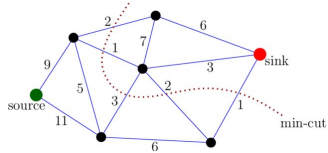


Figure: a mincut example

Here the model is much simpler, but the problem is still NP-hard. So we look for good enough solutions.

4 Outline

4 Conclusion

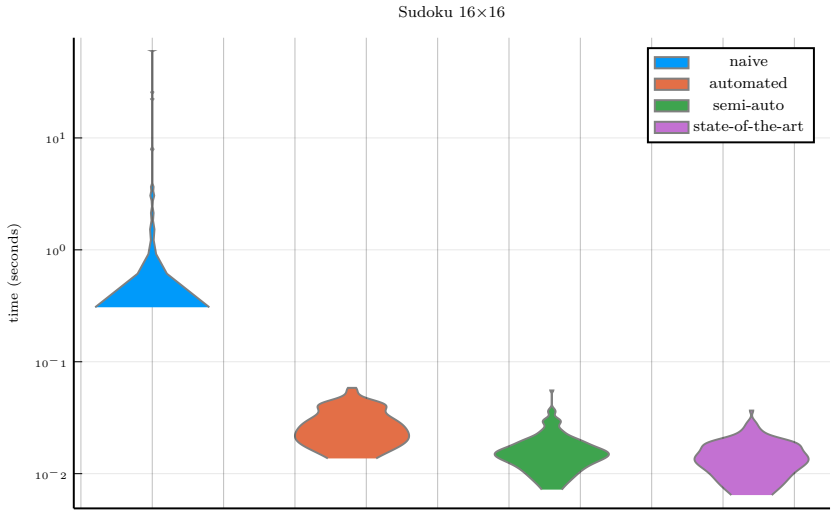
Most specialists of a field cannot do the following:

- Iterates over billions of years such as natural processes
- Write complex mathematical models that requires weeks of work for a scientist specialized in Optimization
- Fine-tune decision-making software to be able to make decisions in real-time

Model-as-you-speak framework

We propose a model-as-you-speak framework that allows non-specialists to solve complex decision-making problems.
We take care of the solving part, you can take care of making models of you field specialties!

4 Support you from the shadows



4 Framework

Our model-as-you-speak framework is available on GitHub
https://github.com/JuliaConstraints/LocalSearchSolvers.jl

- The framework is based on the JuMP (Julia for Mathematical Programming) ecosystem
- Works natively with multithreaded and distributed computing
- Can be used with any solver supported by JuMP (and we add more solvers every day ... or month maybe)

4 Try 1

4 Try 2

4 Try 3

4 Try 4 (really?)