TulipaEnergy

Faster Solving and Higher Detailed Large-Scale Energy System Models

Diego Tejada | JuMP dev 2024

TNO innovation for life

TulipaEnergy

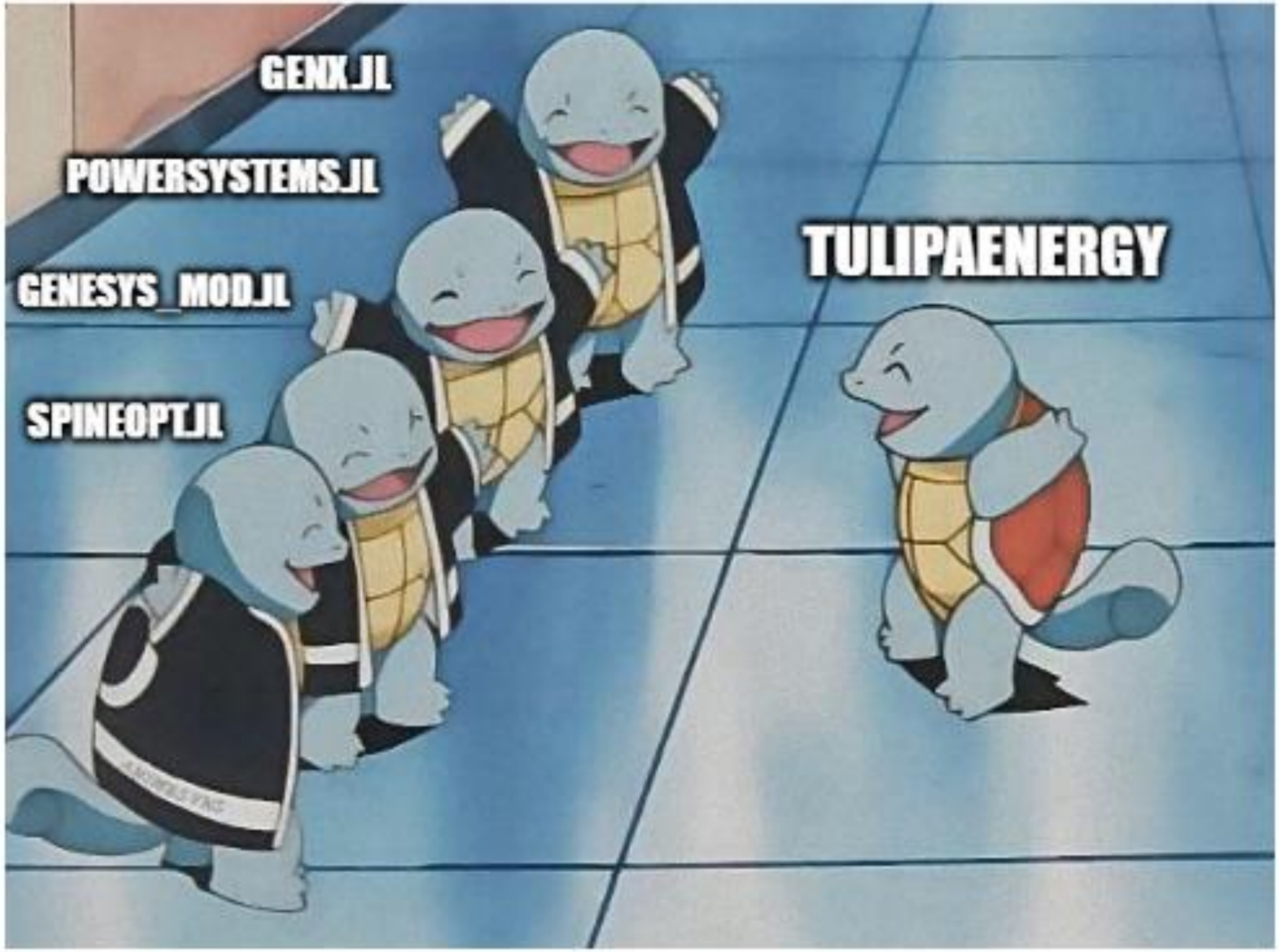
TNO innovation for life

Netherlands Organisation for Applied Scientific Research

Creating impactful innovations for the sustainable wellbeing and prosperity of society.

MOBILITY & BUILT ENVIRON...	DEFENCE, SAFETY & SECU...	ENERGY & MATERIALS TRA...
HEALTHY LIVING & WORK	HIGH TECH INDUSTRY	ICT, STRATEGY & POLICY

Yet Another Energy System Model in Julia...



The Challenge

- Models aid in integrating renewable energy and coupling energy carriers.
- Optimizing investments helps stakeholders understand system dynamics for energy transition.
- Existing models excel in either technological, operational, or spatial-temporal detail, but never all three simultaneously.
- Key challenge: including enough details while remaining computationally tractable.

Project Description

- New energy model from scratch (model design and coding)
- Sector coupling: e.g., electricity, H2 and heat
- The main objective is to determine the optimal investment and operation decisions
- Representation of different types of energy assets (e.g., producers, consumers, conversions, storages, and transports)

⭐ Focus on compact and efficient formulations and implementations suitable for expert energy system researchers

The TulipaEnergy Team

TNO innovation for life

Germán Morales

Diego Tejada

Lauren Clisby

Ni Wang

Wester Coenraads

netherlands eScience center

Abel Soares Siqueira
Julia Optimisation Expert

Suvayu Ali
Data-pipeline Expert

TU Delft

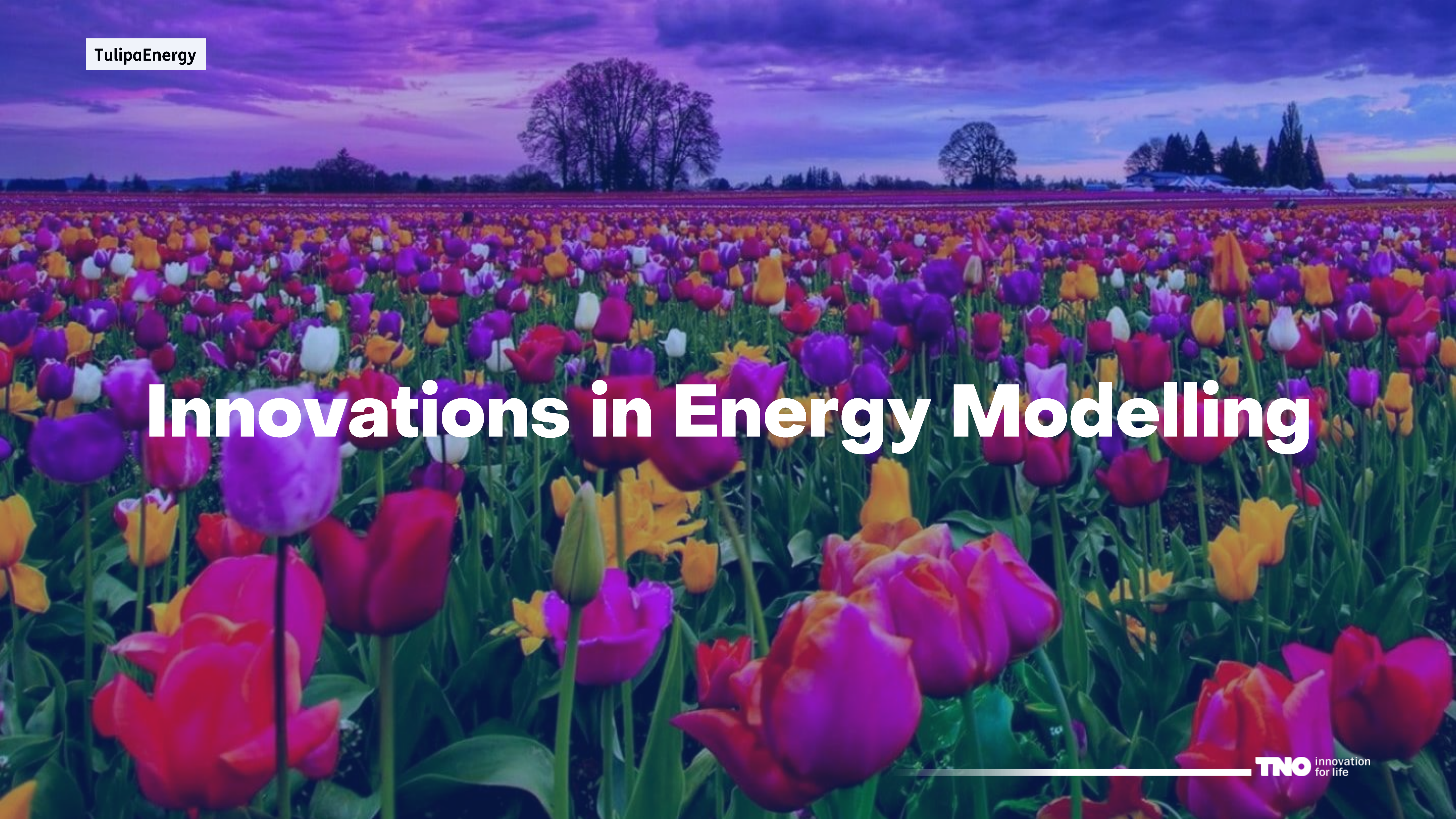
Greg Neustroev
Postdoc: Blended Rep. Periods

Maaike Elgersma
PhD: Accurate & Efficient Formulations

Utrecht University

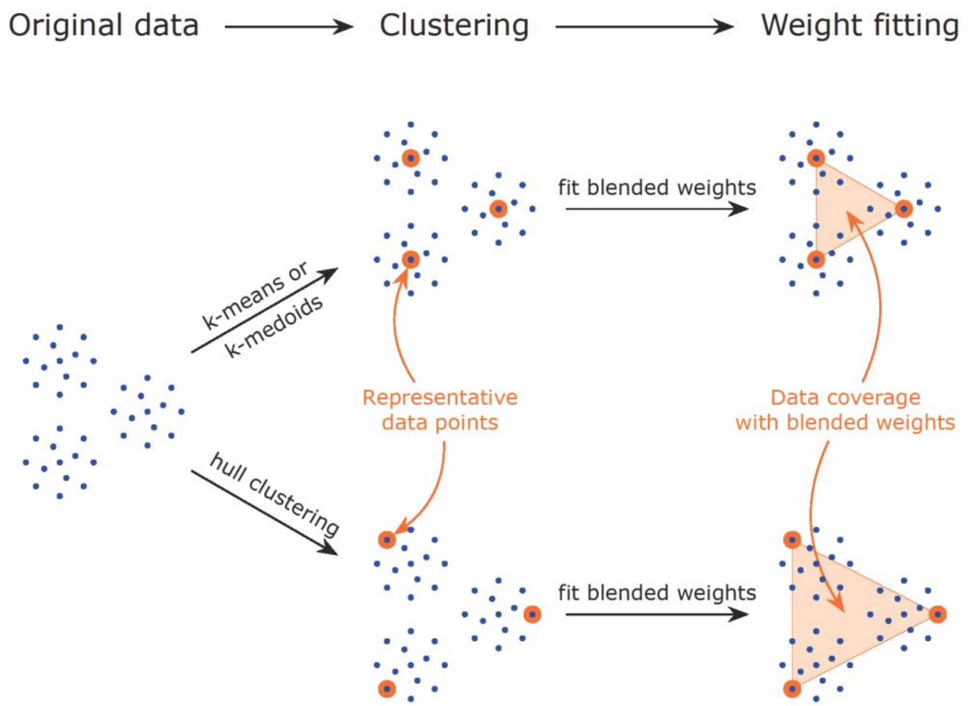
Zhi Gao
PhD: Fully Flexible Temporal Resolution

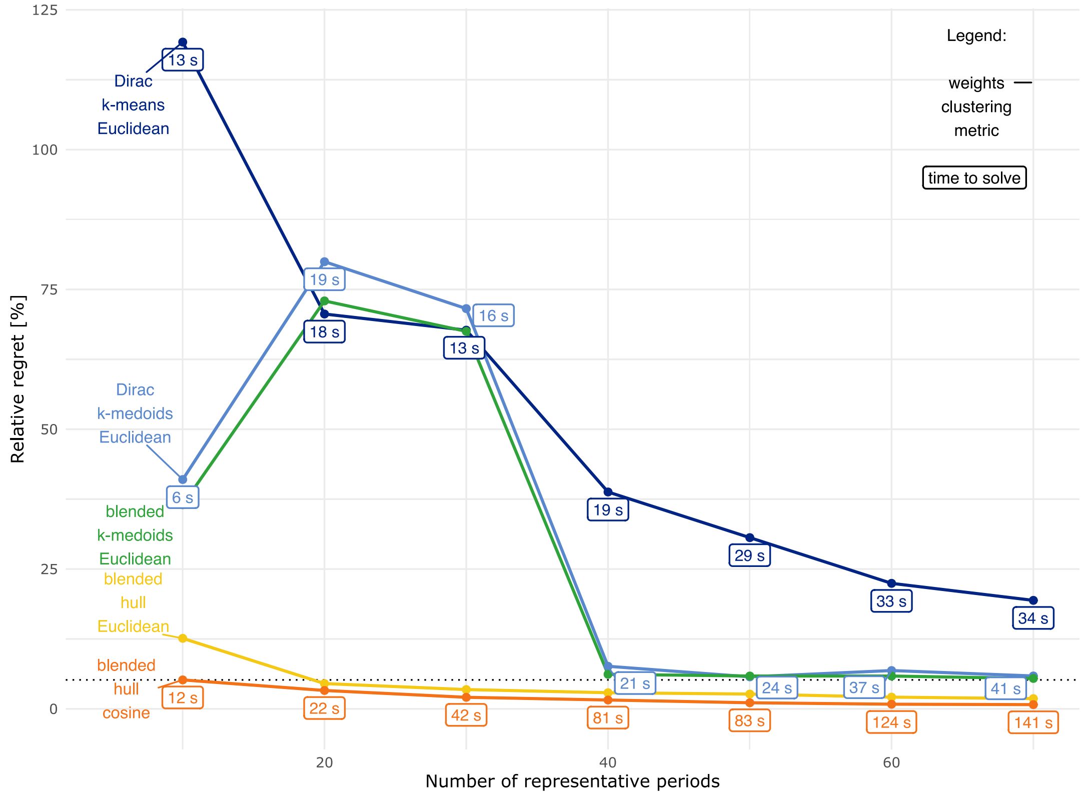
Matthijs Arnoldus
MSc: Modelling to Generate Alternatives


TulipaEnergy
Innovations in Energy Modelling
TNO innovation for life

Hull Clustering with Blended Representative Periods

- Method of hull clustering with blended representative periods (RPs)
- Advantages over k-means/medoids in data representation
- Faster performance with lower relative regret using fewer RPs

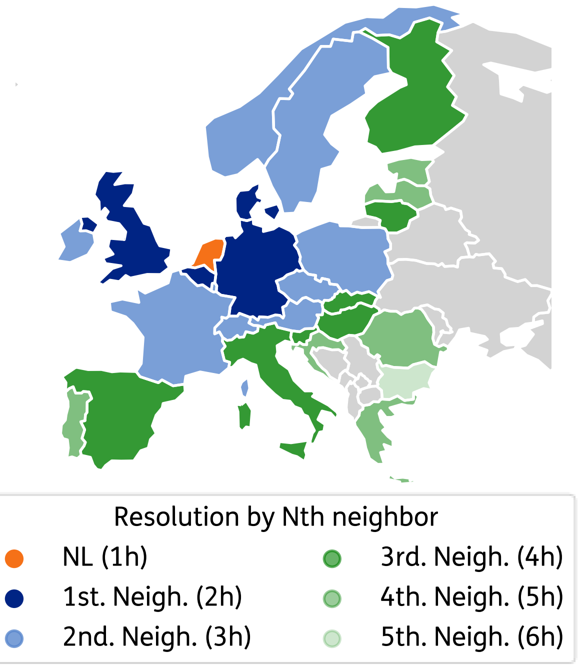




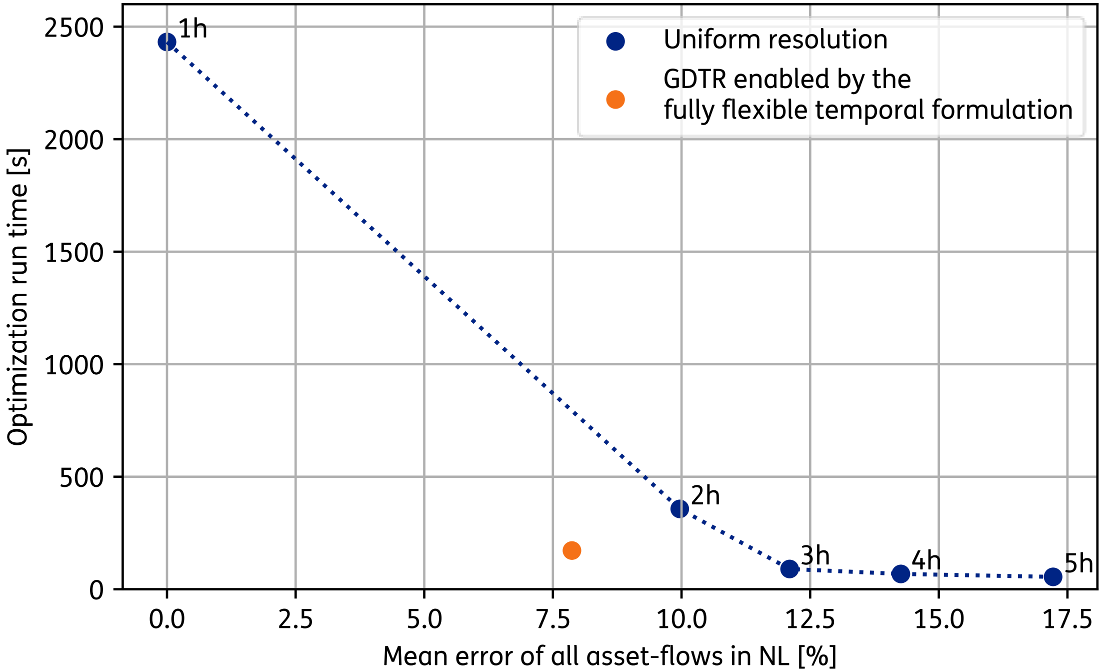
Fully Flexible Temporal Resolution

- Flexible formulation for temporal resolution
- Capability to mix independent resolutions across carriers, regions, and time horizons
- Example of geographical application in the Netherlands

Geographically Decreasing Temporal Resolution (GDTR)



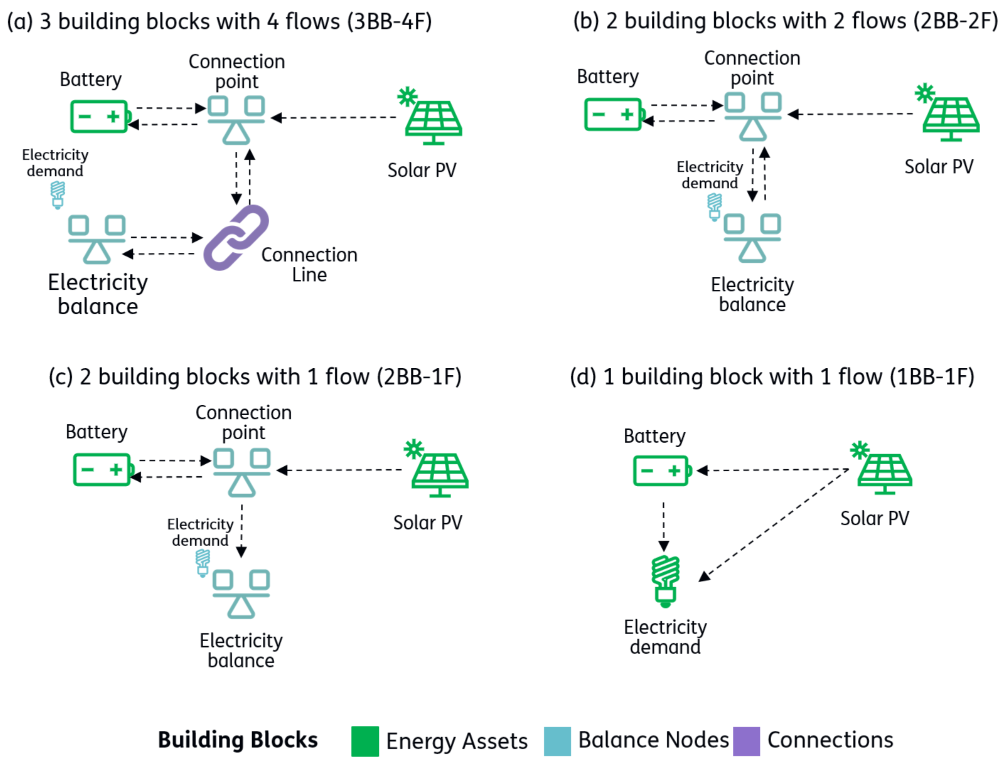
Computational cost vs solution accuracy

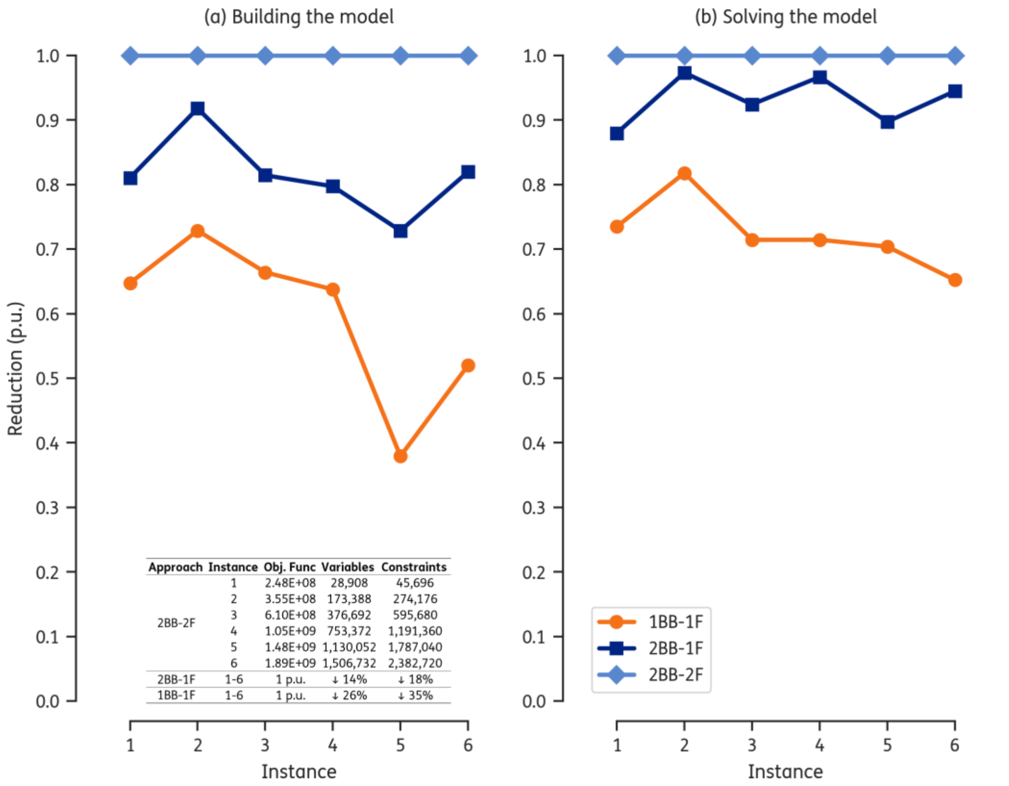


TulipaEnergy

Graph-Based System Representation

- Breaking the misconception of LP as the simplest representation
- Graph theory approach reducing problem size without losing accuracy
- Faster model building and solving with increasing model size





Approach	Instance	Obj. Func	Variables	Constraints
2BB-2F	1	2.48E+08	28,908	45,696
	2	3.55E+08	173,388	274,176
	3	6.10E+08	376,692	595,680
	4	1.05E+09	753,372	1,191,360
	5	1.48E+09	1,130,052	1,787,040
	6	1.89E+09	1,506,732	2,382,720
2BB-1F	1-6	1 p.u.	↓ 14%	↓ 18%
1BB-1F	1-6	1 p.u.	↓ 26%	↓ 35%

TulipaEnergy

TulipaEnergy and Julia/JuMP

TulipaEnergy Packages

TulipaEnergyModel.jl

Builds and runs the optimisation model

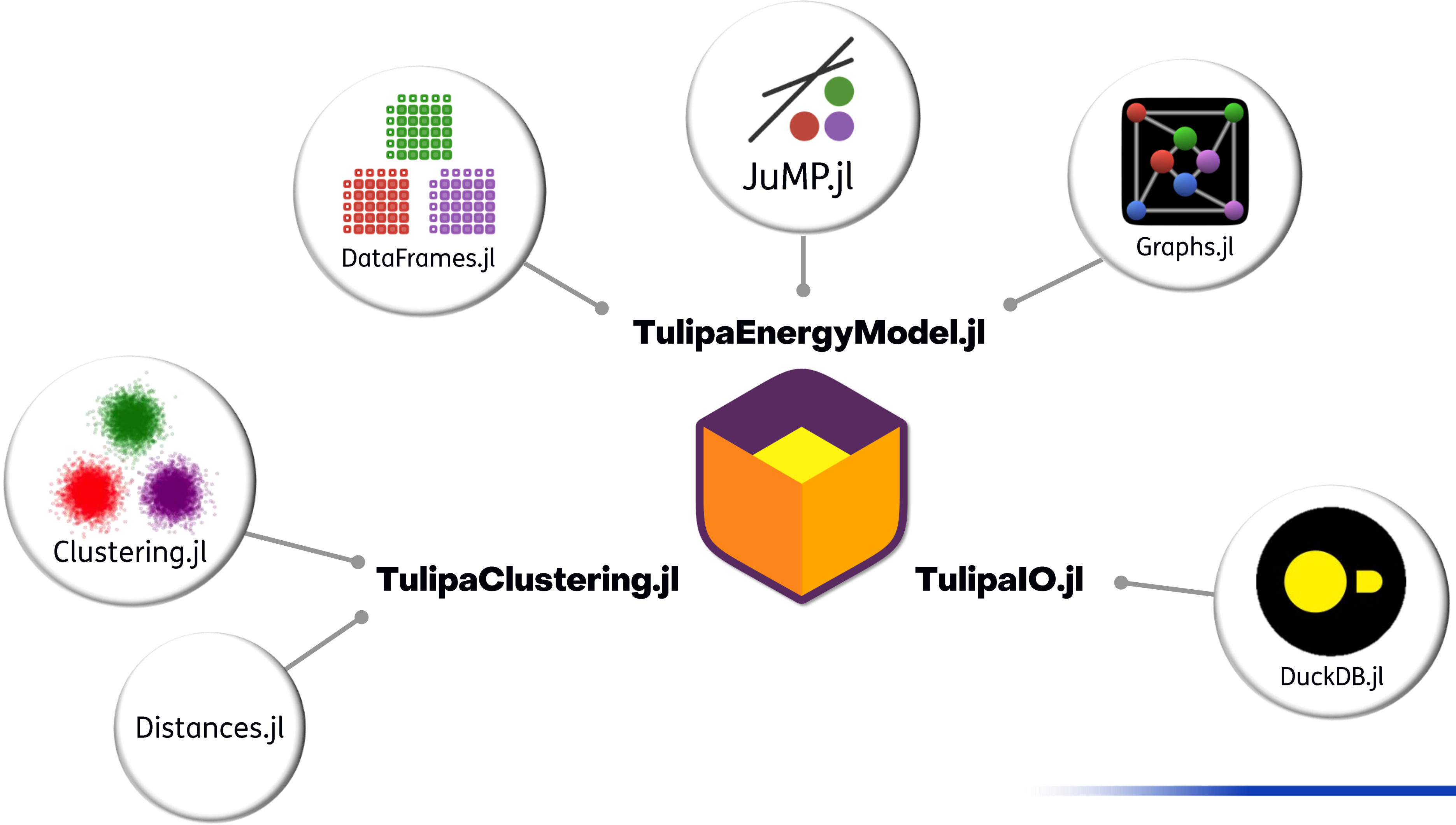
TulipaClustering.jl

Selects the Blended Representative Periods

TulipaIO.jl

Script-based IO for data manipulation

TulipaEnergy Main Dependencies



Tulipa

TulipaEnergy and Useful Tool for Development

TulipaEnergyModel.jl

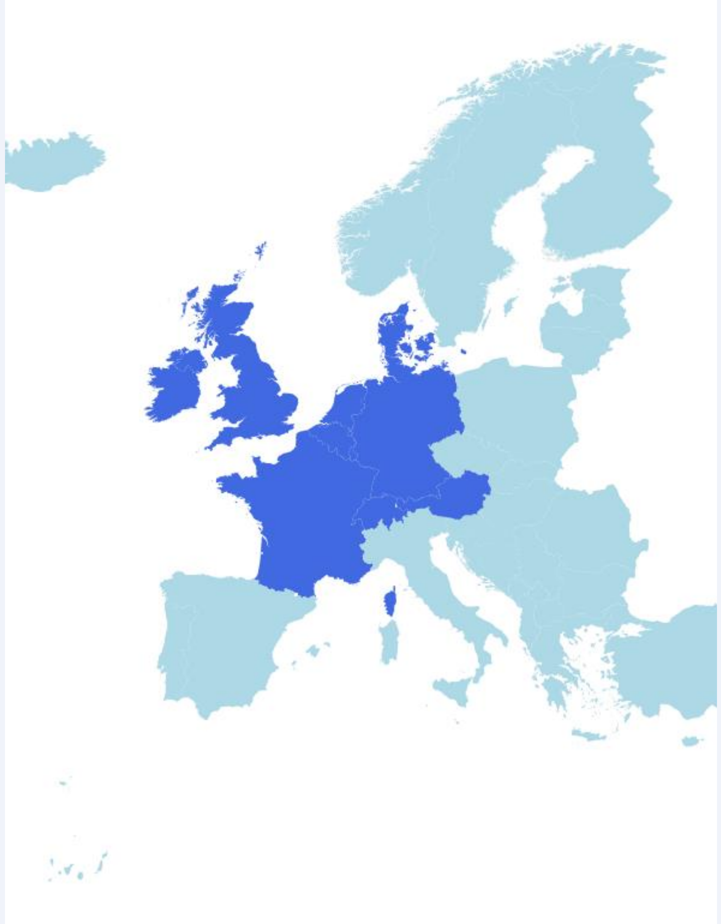
- Open-source Julia/JuMP package available on GitHub

- Timeline:
  - 2023 → Core features development and innovations
  - 2024 → Multi-year investment and power system operation constraints
  - 2025 → Operation constraints in other sectors (e.g., gas) and uncertainty

- Applying best practices for software development (e.g., atomic commits, semantic versioning, code review, tests, documentation)

Performance Results – Western European Countries

- 10 European Countries with an hourly resolution
- Minimize operating costs for one year
- Optimization problem size:
  - # variables: ≈ 1.2 million
  - # constraints: ≈ 2.5 million
- **TulipaEnergyModel.jl** building time† and memory usage:
  - Initial code: 314s and 32GB
  - Optimized code: 86s (**↓73%**) and 18GB (**↓44%**)



†First draft of the code: 8min for 2 EU countries

How did we achieve it?

Basic JuMP

```
@constraint(
    model,
    max_transport_flow_limit[f ∈ Ft, rp ∈ RP, B_flow ∈ graph[f...].partitions[rp]],
    duration(B_flow, rp) * flow[f, rp, B_flow] ≤ upper_bound_transport_flow[f, rp, B_flow]
)
```

Using DataFrames to linearise indices

```
df = filter(row -> (row.from, row.to) ∈ Ft, df_flows)
model[:max_transport_flow_limit] = [
    @constraint(
        model,
        duration(row.time_block, row.rp) * flow[row.index] ≤ upper_bound_transport_flow[row.index],
        base_name = "max_transport_flow_limit[($(row.from),$(row.to)),$(row.rp),$(row.time_block)]"
    ) for row in eachrow(df)
]
```

```
julia> energy_problem.dataframes[:flows]
648×9 DataFrame
 Row │ from    to      rep_period  timesteps_block  efficiency  index  flow
     │ Symbol  Symbol  Int64       UnitRange…       Float64     Int64  GenericV…
─────┼──────────────────────────────────────────────────────────────────────────────────
   1 │ ocgt    demand           1  1:1                     0.0      1  flow[(ocgt, demand), 1, 1:1]
   2 │ ocgt    demand           1  2:2                     0.0      2  flow[(ocgt, demand), 1, 2:2]
   3 │ ocgt    demand           1  3:3                     0.0      3  flow[(ocgt, demand), 1, 3:3]
```

https://discourse.julialang.org/t/help-improving-the-speed-of-a-dataframes-operation/107615

Some Final Thoughts

Good Things

- Speed & efficiency!
- Straight-forward syntax
- Great user community support!
- Others: DuckDB, Graphs, Clustering...
- Friendly to both
  researchers and software engineers

Some Final Thoughts

Good Things

- Speed & efficiency!
- Straight-forward syntax
- Great user community support!
- Others: DuckDB, Graphs, Clustering...
- Friendly to both researchers and software engineers

Room for Improvement

- Skill required for best speed/memory
- Add more tips for speed/efficiency improvement
- New Extensions?
  - Gather Update Solve Scatter (GUSS)
  - NearOptimalAlternatives.jl* for modelling to generate alternatives – MGA

Check out our GitHub!

TNO innovation for life