This project has received funding from the European Union's Horizon2020 research and innovation programme under grant agreement N° 863819

FlexPlan

16/02/2023 – EnergyVille (Genk, BE)

Network expansion planning with FlexPlan.jl

Matteo Rossini
RSE

Summary

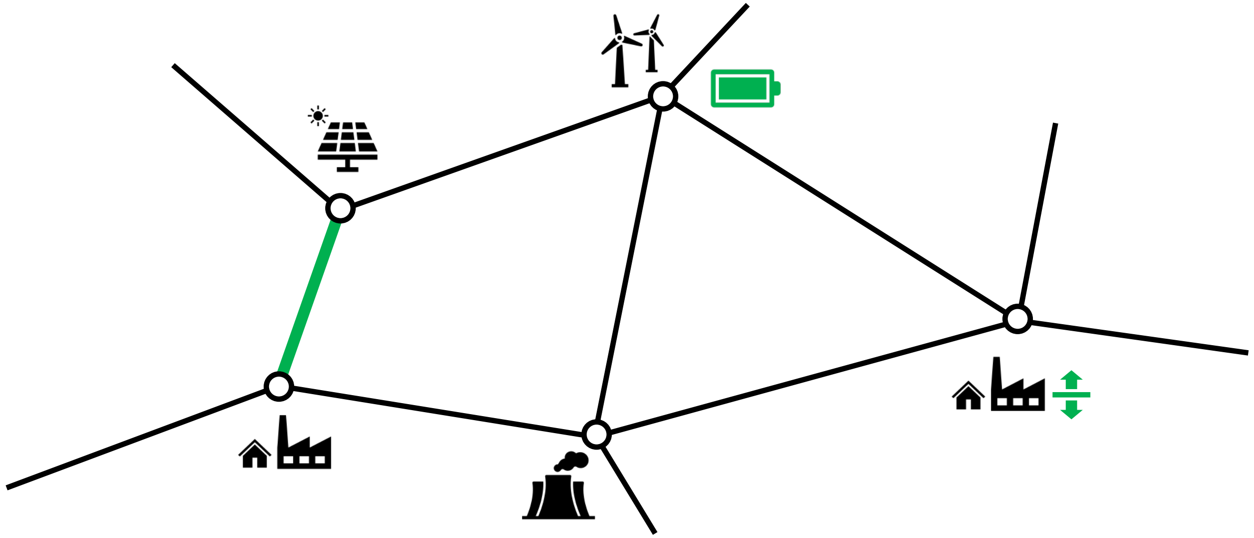
FlexPlan

1. The FlexPlan project and FlexPlan.jl
2. Overview of models
3. Insight into the implementation
4. T&D decoupling heuristics
5. Benders decomposition

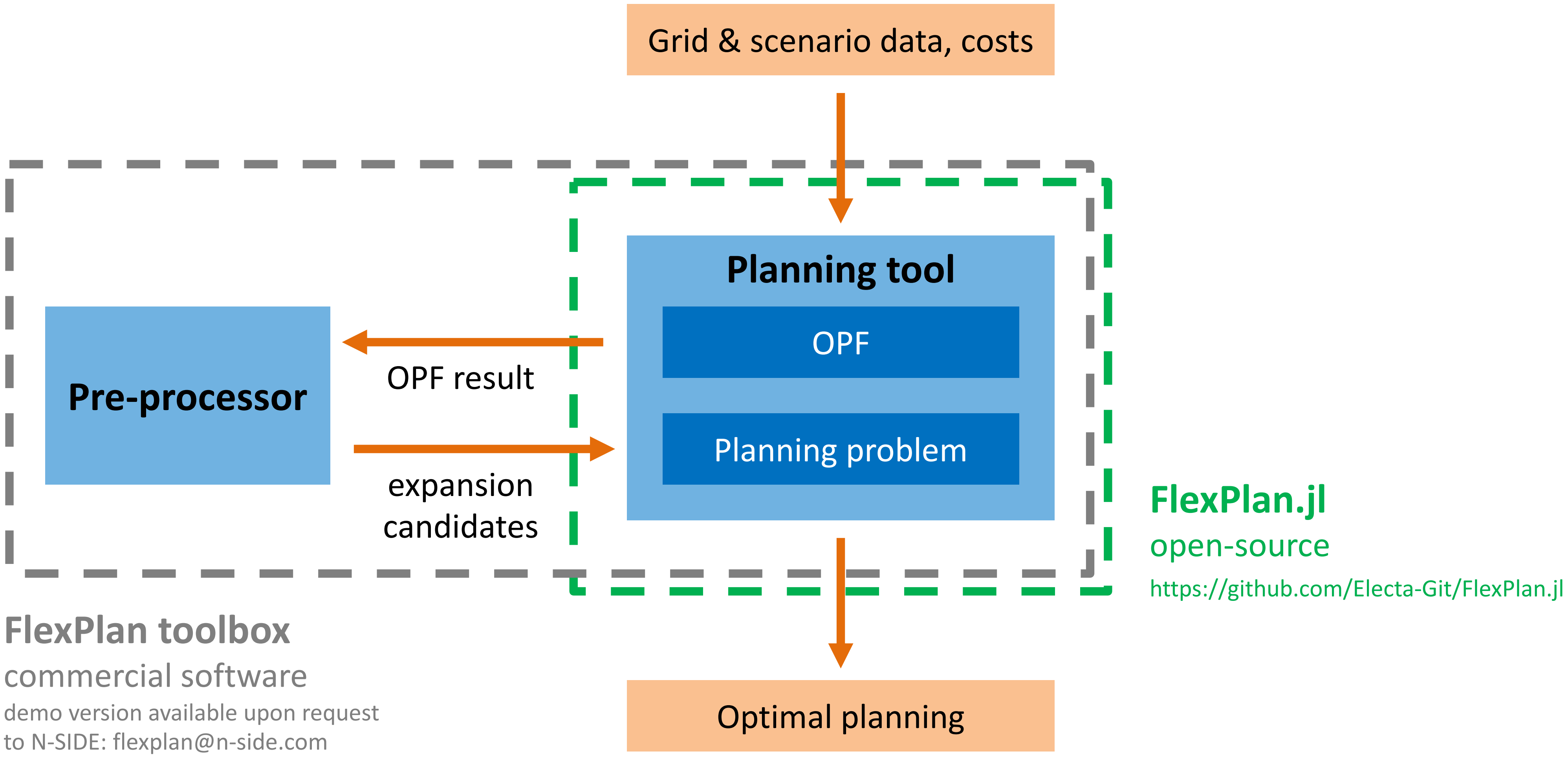
The FlexPlan project

FlexPlan

Aim: *establish a new **grid planning methodology** considering the opportunity to introduce new **storage** and **flexibility resources** in **T&D grids** as an alternative to conventional **network expansion**.*



FlexPlan software components



FlexPlan.jl is a "container" package

FlexPlan

Original content in FlexPlan.jl

problem definition	stochastic network expansion planning
network components	storage, flexible loads
power flow models	simplified DistFlow (w. OLTCs and linearized apparent power bounds)
data structures	for stochastic/multiperiod optimization
grid interconnection model	T&D coupling through generators
algorithms	T&D decoupling heuristic, Benders decomposition
test networks	2 transmission, 2 distribution (w. candidates and stochastic time series)
I/O functions	for data import/export and plotting

Cost minimization problem

FlexPlan

Sets

operation periods	`hour`	constant continuous variables (power, voltage, …)
scenarios	`scenario`	different timeseries for renewables and demand
planning periods	`year`	constant binary variables (investment decisions)

Objective function

$$\min \sum_{y \in Y} \left([\mathrm{CAPEX}]_y + \sum_{s \in S} \pi_s \sum_{t \in T} [\mathrm{OPEX}]_{t,s,y} \right)$$

$[\mathrm{CAPEX}]_y$: new **lines**, **transformers**, **converters**; new **storage** devices; flexibilization of **demand**

π_s: scenario probability

$[\mathrm{OPEX}]_{t,s,y}$: **generation**: cost, curtailment; **demand**: shifting, reduction, curtailment

Storage model

FlexPlan

$$E_{b,s,t,y} = (1-\lambda_b)^{\Delta t} E_{b,s,t-1,y} + \Delta t \left(\eta_b^{abs} P_{b,s,t,y}^{abs} - \frac{P_{b,s,t,y}^{inj}}{\eta_b^{inj}} + \xi_{b,s,t,y} \right)$$

- $E_{b,s,t,y}$: energy stored at time t
- $(1-\lambda_b)^{\Delta t}$: self-discharge
- $E_{b,s,t-1,y}$: energy stored at time $t-1$
- $P_{b,s,t,y}^{abs}$: power absorbed from network
- $P_{b,s,t,y}^{inj}$: power injected into network
- $\xi_{b,s,t,y}$: power of external process

Constraints

- bounds on stored energy and absorbed/injected power
- fixed energy at beginning of planning horizon
- lower bound for energy at end of planning horizon

⚠ No charge/discharge exclusivity constraint

Flexible load model

FlexPlan

$$P^{d}_{u,s,t,y} = P^{ref}_{u,s,t,y} + P^{sh,up}_{u,s,t,y} - P^{sh,dn}_{u,s,t,y} - P^{red}_{u,s,t,y} - P^{curt}_{u,s,t,y}$$

absorbed power	reference power	upward and downward demand shift	voluntary reduction	involuntary curtailment

Constraints

- non-negative absorbed power
- bounds on demand shift and voluntary reduction as fraction of reference power
- sum of upward and downward demand shifts rebalanced periodically

Simplified DistFlow model

FlexPlan

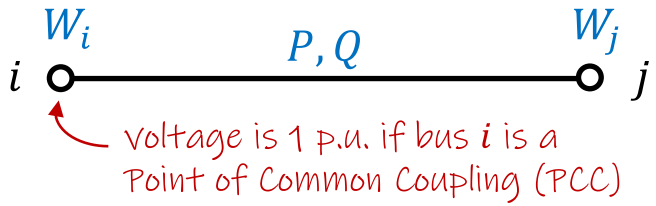
linear approximation of branch flow model for distribution networks (requires radial topology)

Variables

buses: squared voltage magnitude (W)

branches: active and reactive power (P, Q)

OLTCs: inverse squared transformation ratio (ρ)



Voltage drop constraint

line

$$W_i - W_j = 2(rP + xQ)$$

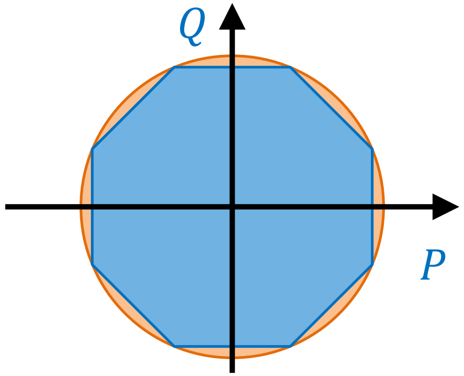
transformer connected to PCC

$$\frac{1}{\tau^2} - W_j = 2(rP + xQ)$$

OLTC connected to PCC

$$\rho - W_j = 2(rP + xQ)$$

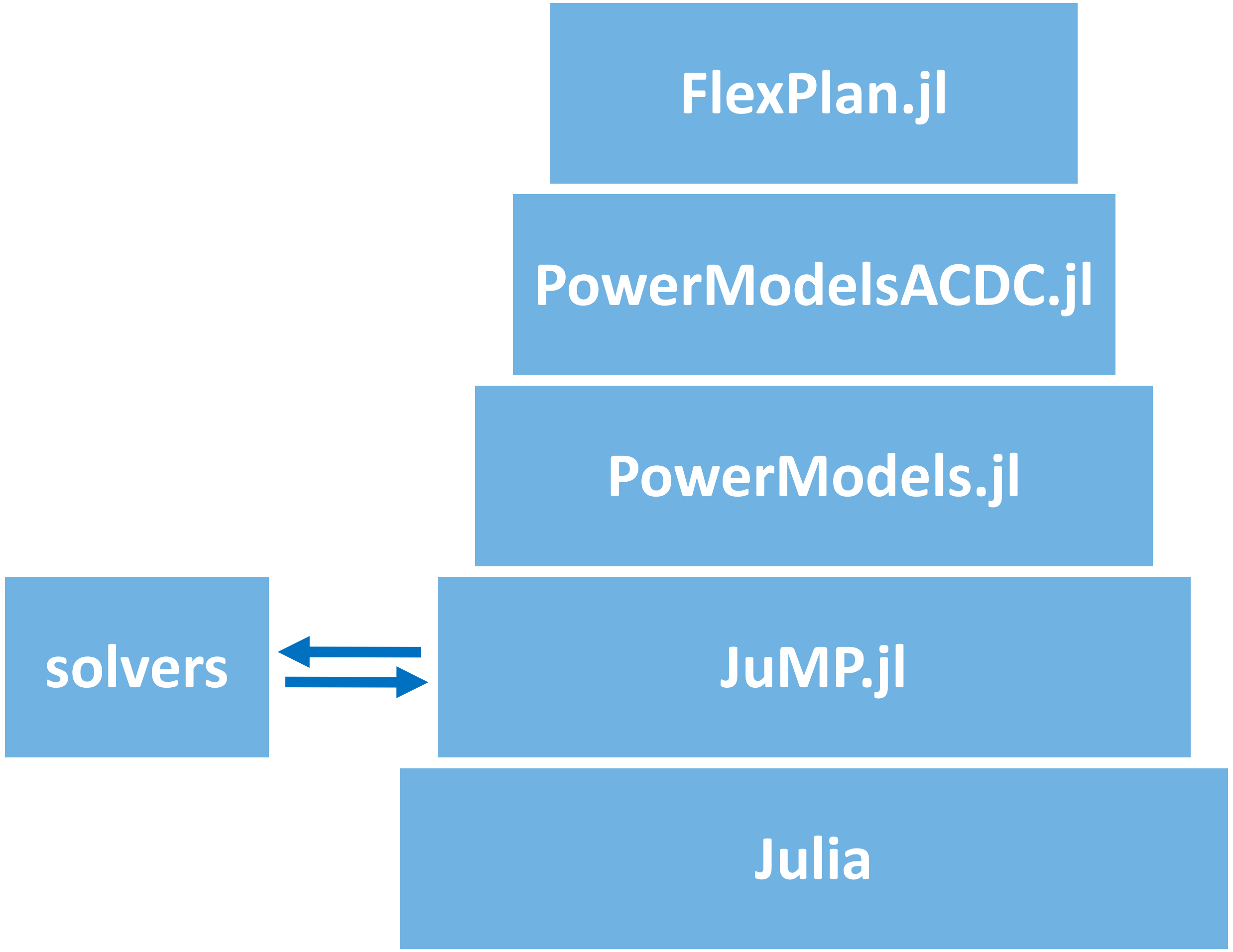
Branch apparent power bound



linearized as regular octagon inscribed in $P^2 + Q^2 \leq S^2$ circle

Software stack

FlexPlan



DC grid & components

AC grid & components, data structures

optimization

language

Input

FlexPlan

Static data

Matpower file
with custom extensions

Time series

Any file

```
data = parse_file(file; kwargs...)
scale_data!(data; kwargs...)
add_dimension!(data, name, ...)
```

user-provided function

Single-network dictionary (PowerModels-like)

Dictionary with vector values

```
make_multinetwork(data, time_series; kwargs...)
```

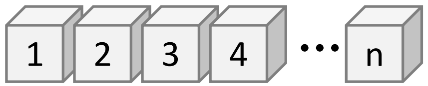
Multinetwork dictionary (PowerModels-like)

Sample networks: https://github.com/Electa-Git/FlexPlan.jl/tree/master/test/data

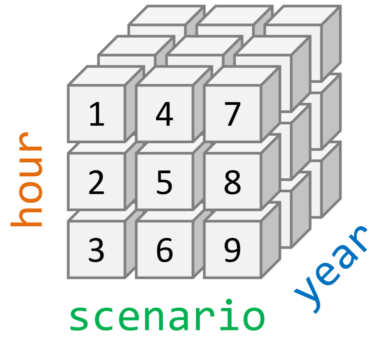
Dimensions

FlexPlan

Data is stored as `Dict{Int,Any}`…




```
add_dimension!(data, :hour, 24)

add_dimension!(data, :scenario, Dict(
               1 => Dict("probability"=>0.7),
               2 => Dict("probability"=>0.3)))

add_dimension!(data, :year;
               metadata=Dict("scale_factor"=>10))
```



…but accessed as a multidimensional array

```
nw_ids(pm; kwargs…)
similar_ids(pm, n; kwargs...)
similar_id(pm, n; kwargs...)
first_id(pm, n, dimension...)
last_id(pm, n, dimension...)
prev_id(pm, n, dimension)
prev_ids(pm, n, dimension)
next_id(pm, n, dimension)
next_ids(pm, n, dimension)
```

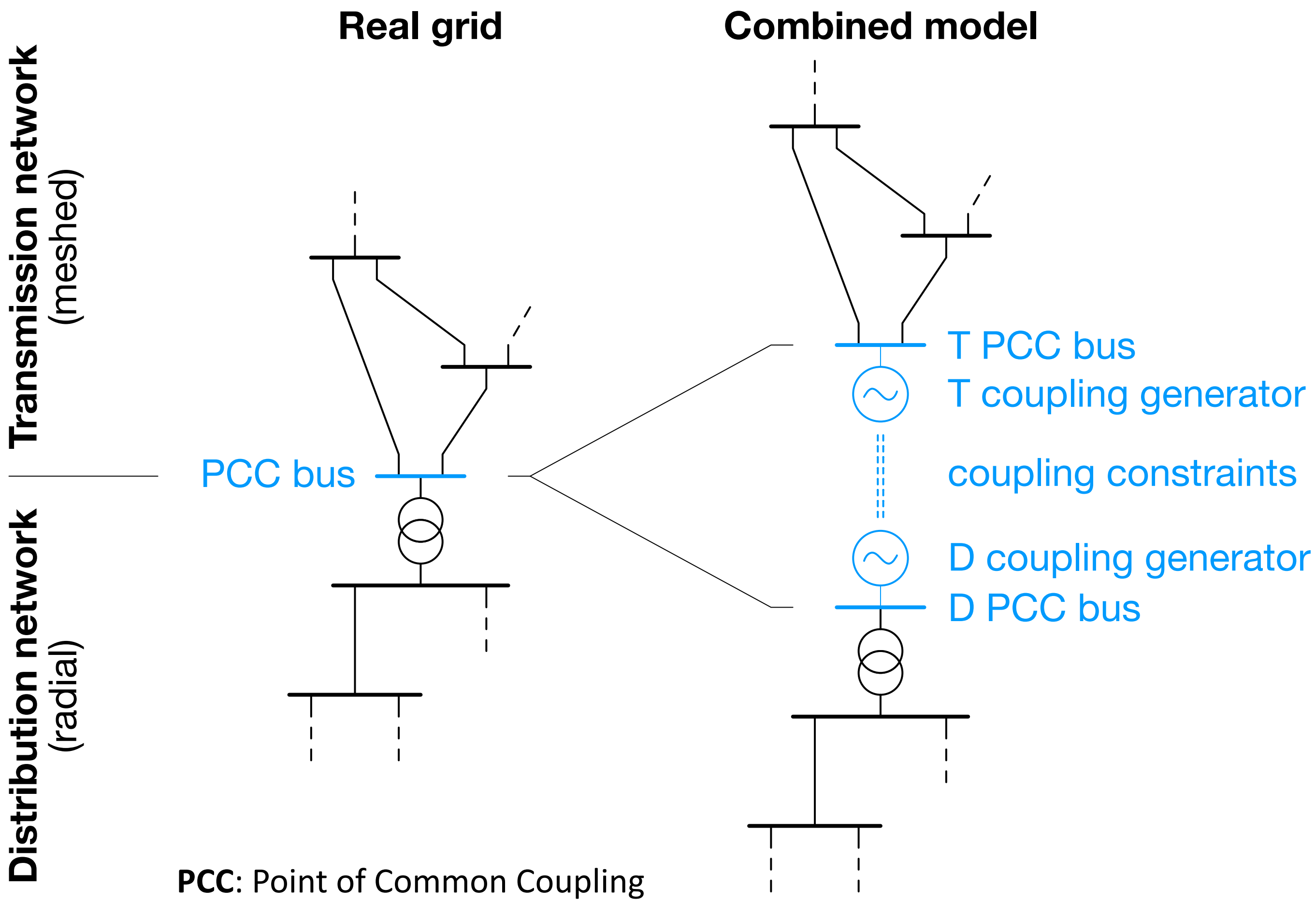
and many other functions

How to run optimizations

```
import FlexPlan as _FP
import HiGHS
optimizer = _FP.optimizer_with_attributes(HiGHS.Optimizer, "output_flag"=>false)

sn_data = _FP.parse_file("my_matpower_file.m")
_FP.scale_data!(sn_data)
_FP.add_dimension!(sn_data, :hour, 24)
_FP.add_dimension!(sn_data, :scenario, Dict(1=>Dict("probability"=>0.7), 2=>Dict("probability"=>0.3)))
_FP.add_dimension!(sn_data, :year; metadata = Dict("scale_factor"=>10))

time_series = my_function_to_load_time_series()
mn_data = _FP.make_multinetwork(sn_data, time_series)

setting = Dict("output" => Dict("branch_flows"=>true))
result = _FP.simple_stoch_flex_tnep(mn_data, _FP.BFARadPowerModel, optimizer; setting)
```

Combined T&D model

FlexPlan



PCC: Point of Common Coupling

- ✓ Different power flow models for T&D
- ✓ Uses only standard network components
- ✓ Coupling constraints do not depend on the power flow model

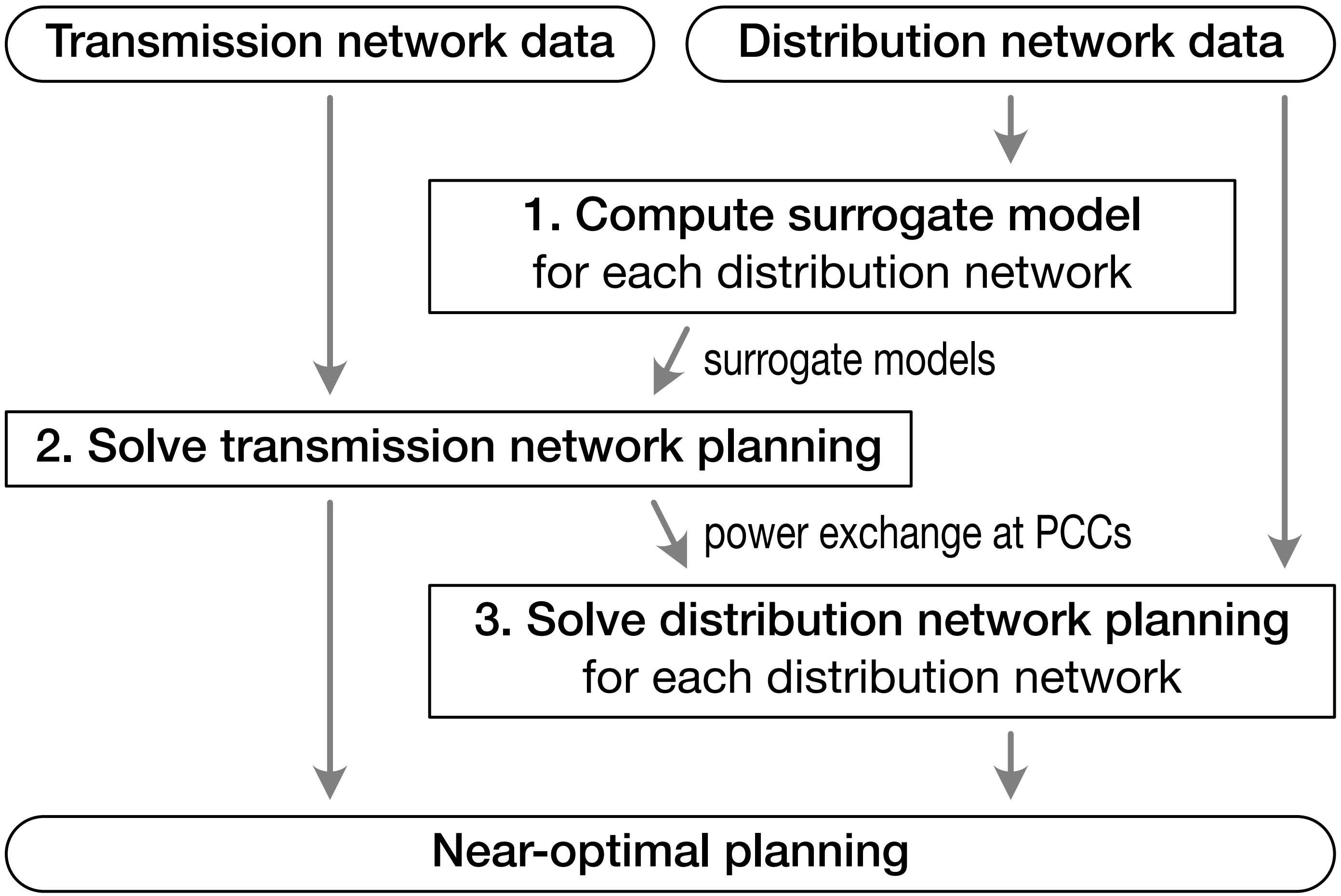
- ✗ Voltage variables are not coupled

Suitable for planning, not for operation

ℹ The combined system is passed to functions as two arguments: a `Dict` for transmission grid and a `Vector{Dict}` for distribution grids

T&D decoupling heuristic

FlexPlan



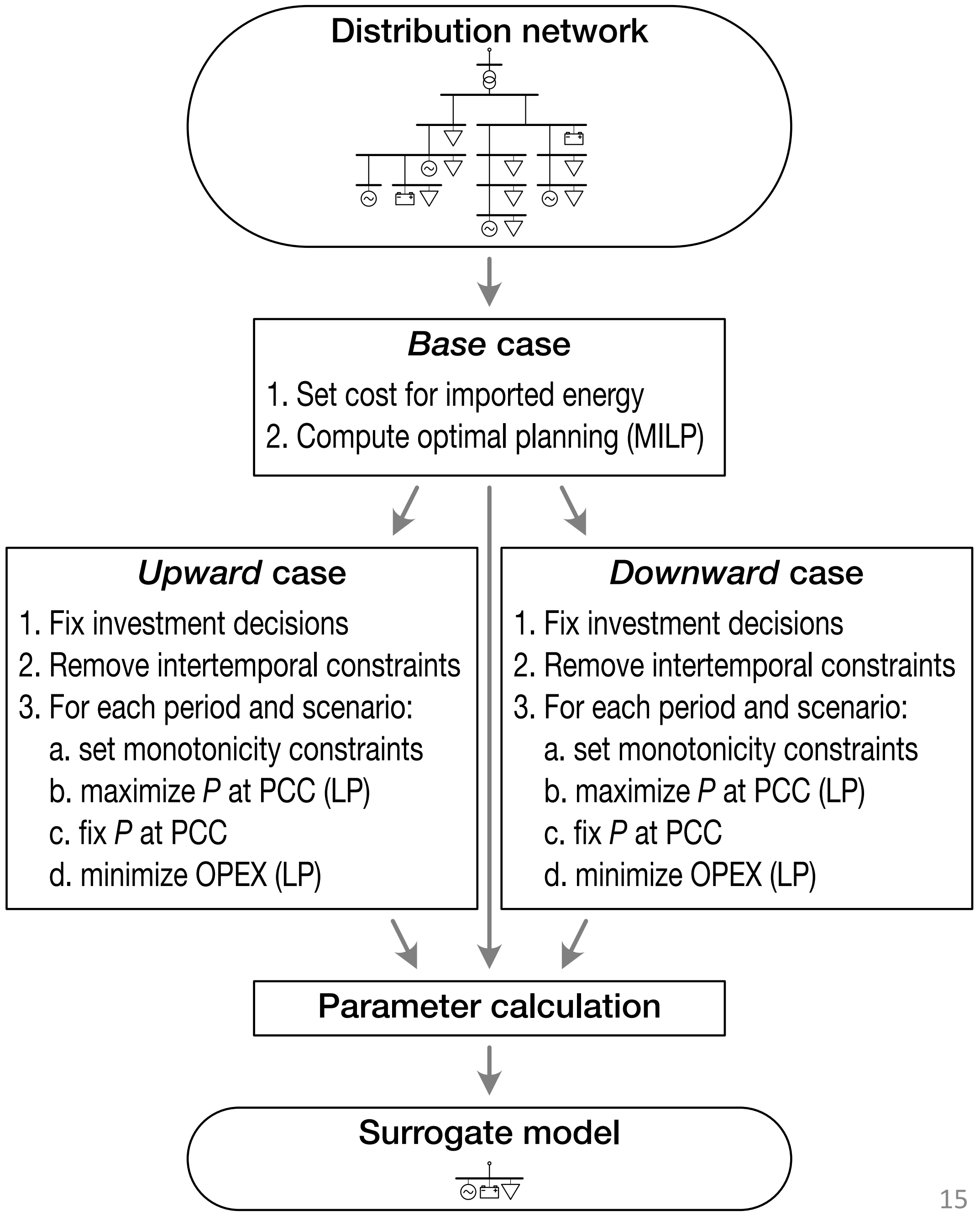
T&D decoupling heuristic

Surrogate model components:

- one generator
- one storage device
- one flexible load

with parameters such that:

- feasibility *implies* feasibility in original model
- cost *approximates* cost in original model



T&D decoupling heuristic

FlexPlan

Surrogate model assumptions

Independence of components

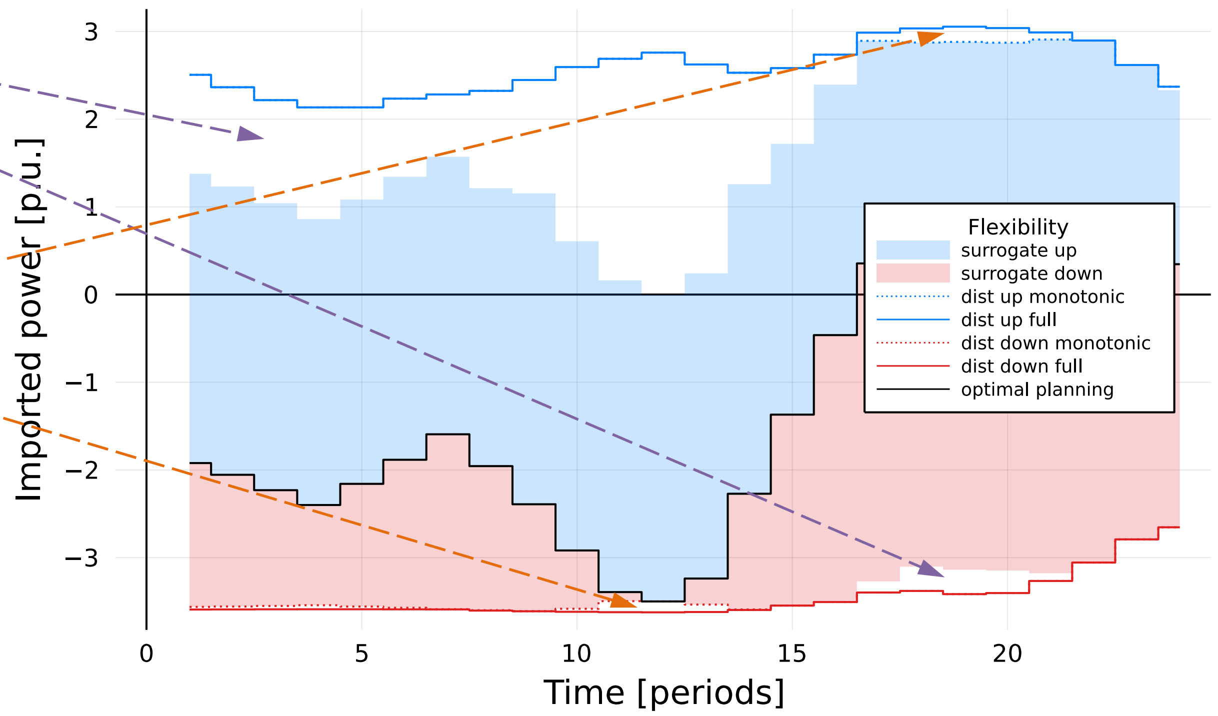
Generator, storage device and load can be used independently by transmission.

Monotonicity of power variations

A variation in power exchanged with transmission produces a variation of the same sign in components.

Example: if imported power increases, then the load cannot absorb less.

Effects on power exchange at PCC



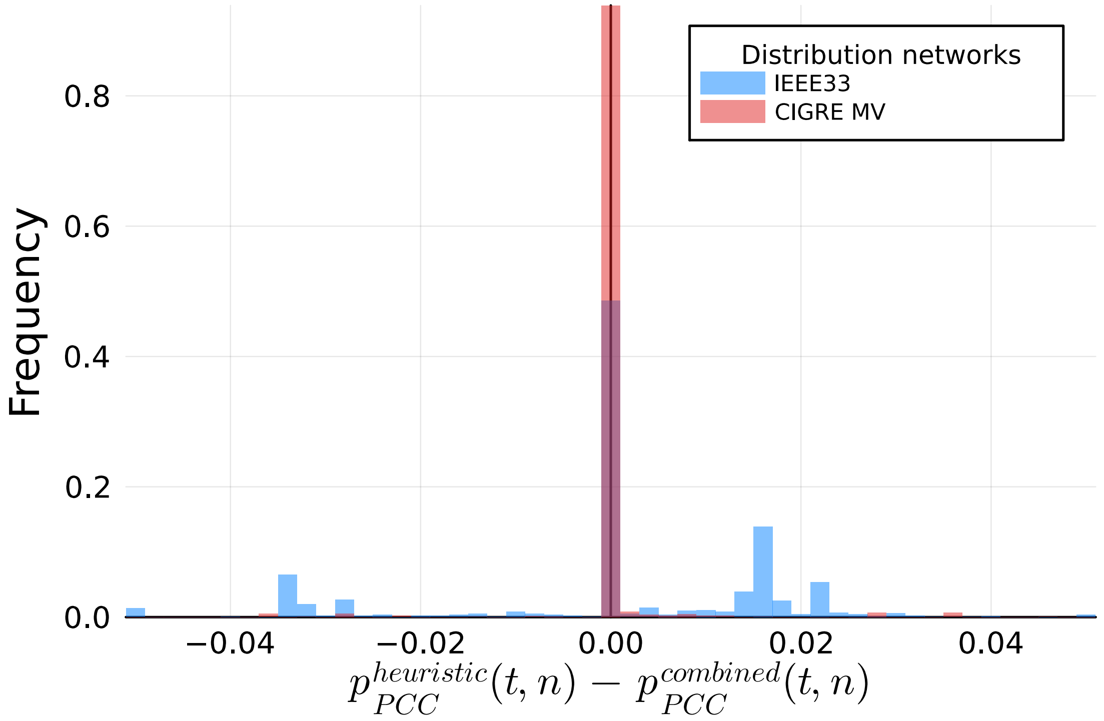
T&D decoupling heuristic

FlexPlan

Test: attach to a transmission network (case67) a variable amount N_d of distribution networks (either IEEE33 or CIGRE MV)

PERFORMANCE COMPARISON OF COMBINED MODEL AND DECOUPLING HEURISTIC

N_d	binary variables	CPU time: combined model [s]	CPU time: decoupling heuristic [s]	CPU time: ratio	relative cost increase
case67 with N_d IEEE33 distribution networks					
1	83	38	21	**0.553**	$-1.1{\cdot}10^{-5}$
4	158	148	25	**0.169**	$6.0{\cdot}10^{-7}$
16	458	1139	41	**0.036**	$1.4{\cdot}10^{-6}$
64	1658	4228	87	**0.021**	$6.6{\cdot}10^{-5}$
case67 with N_d CIGRE MV distribution networks					
1	88	39	23	**0.590**	$-5.3{\cdot}10^{-15}$
4	178	79	20	**0.253**	$6.2{\cdot}10^{-5}$
16	538	210	30	**0.143**	$5.3{\cdot}10^{-15}$
64	1978	6479	59	**0.009**	$4.1{\cdot}10^{-4}$



T&D decoupling heuristic

FlexPlan

Adopts a transmission-follows approach

(distribution investments decided in distribution, used also by transmission)

✓ Fast w.r.t. combined model
typical speedup: 10 ÷ 100 x

✓ Near-optimal result
typical relative cost increase: $< 10^{-4}$

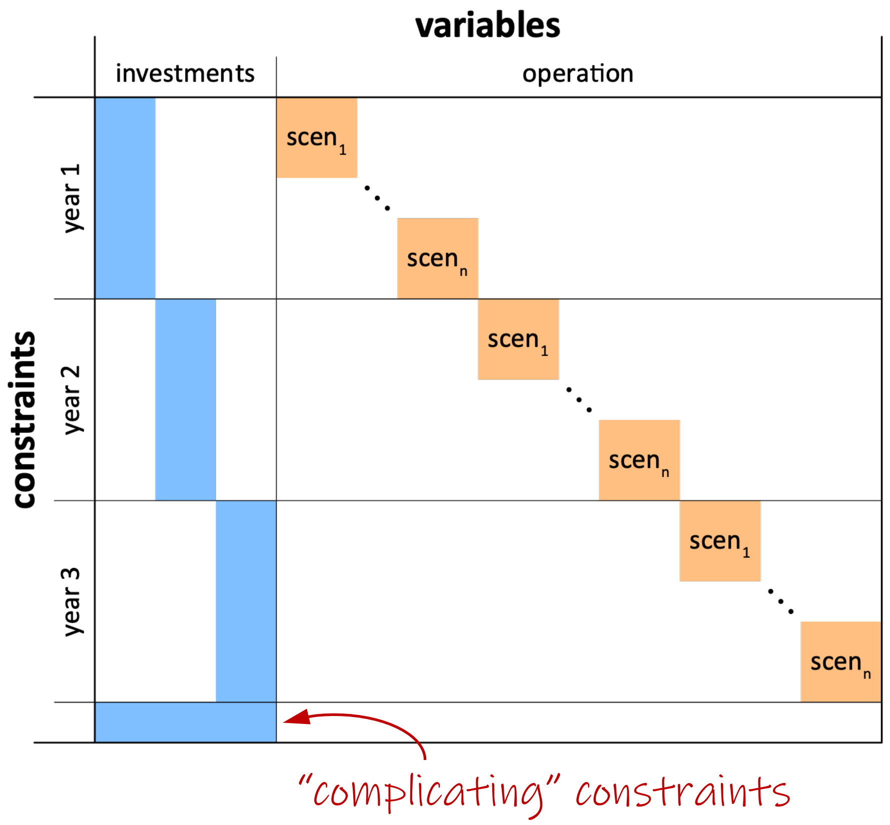
✓ Good solution quality
typical power deviation at PCCs: < 1%

✗ No voltage information shared between T&D
as in combined T&D model

Benders decomposition

FlexPlan

The problem matrix has a block structure



Benders decomposition

FlexPlan

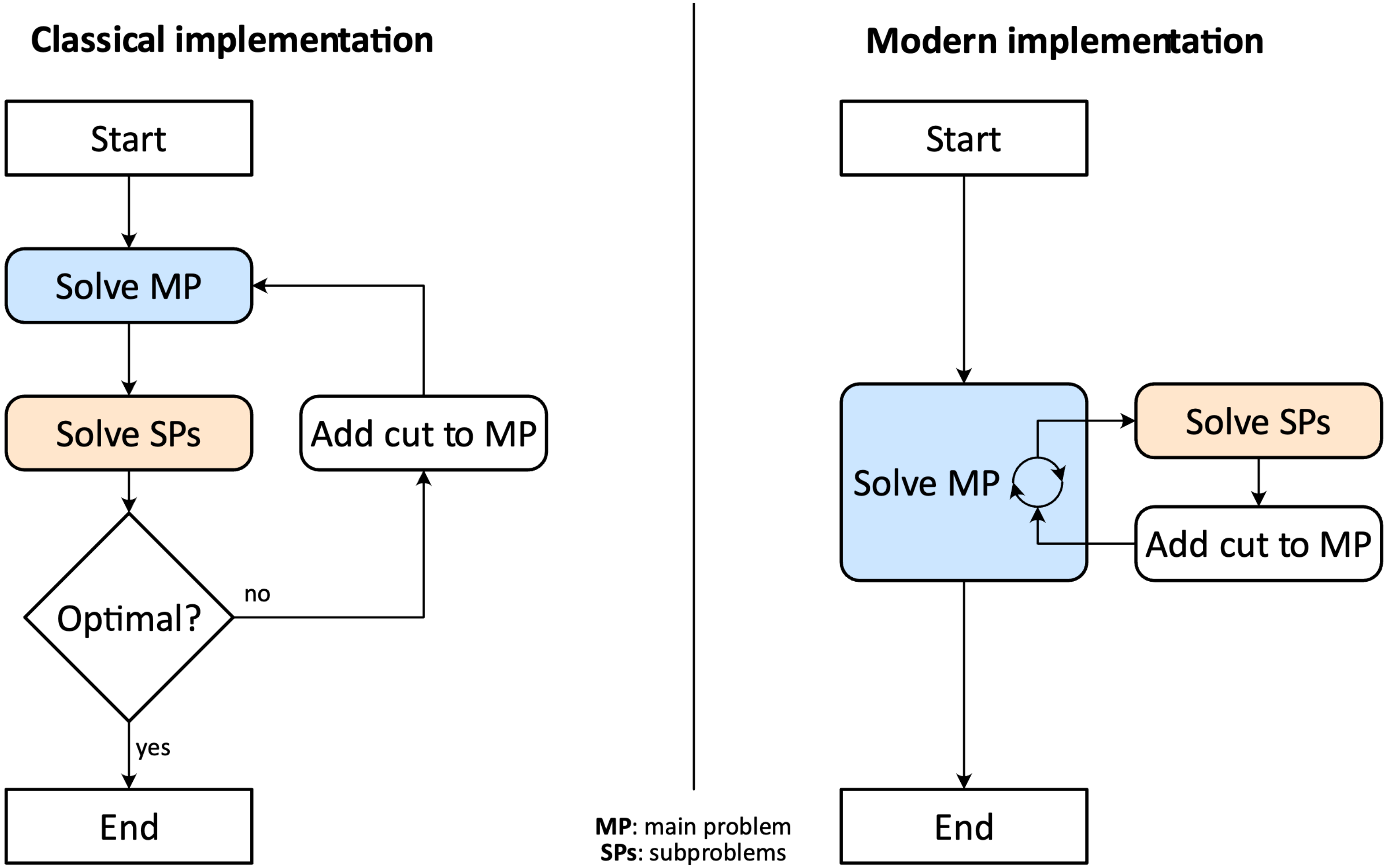
Benders algorithm (1962) applied to planning problem

Repeat until convergence:

1. Solve main problem
   - optimize investment (binary) variables
2. Solve subproblems (one per year/scenario)
   - fix investment decisions
   - optimize operation (continuous) variables
3. Add cut to main problem
   - cuts provide a lower bound of the operation cost

Benders decomposition

FlexPlan



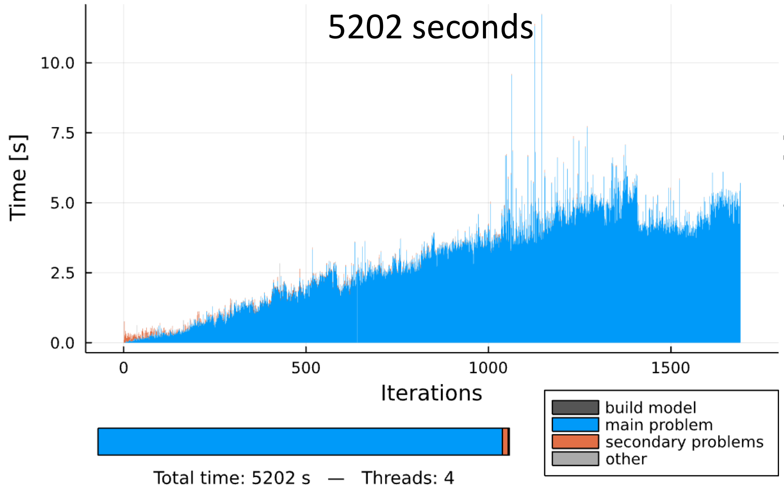
MP: main problem
SPs: subproblems

Benders decomposition

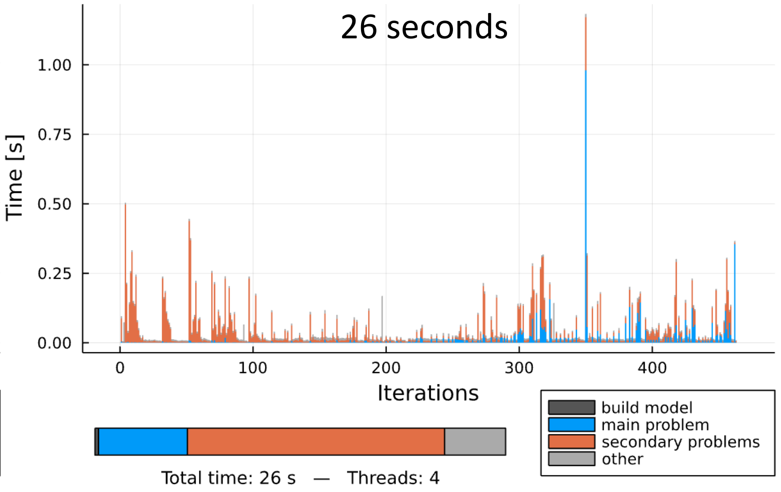
Test: comparison of solve time

Main problem: 357 binary variables and 705 constraints initially
9 subproblems: 2514 continuous variables and 8540 constraints each

Classical implementation



Modern implementation



Key features of implementation

✓ Subproblems are not dualized
- Variant of original Benders decomposition

✓ Multithreaded
- Main problem: multithreaded solver
- Subproblems: solved in parallel (multithread), with single-threaded solvers

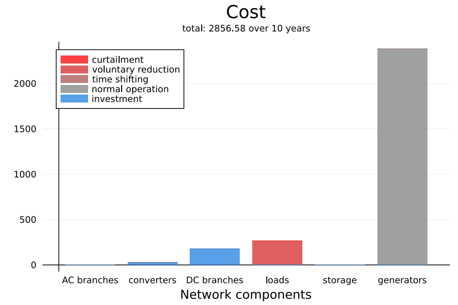
✗ Basic implementation
- No regularization applied

Performance of Benders decomposition implementation in FlexPlan.jl is very case-dependent.
Consider using the Benders decomposition frameworks included in some solvers (SCIP, CPLEX) instead.

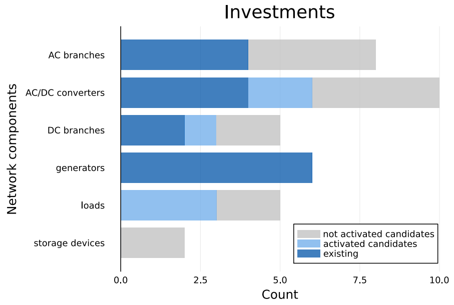
Functions for inspection of results

FlexPlan

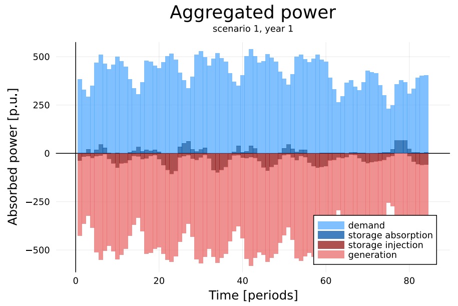
`sol_report_cost_summary(…)`



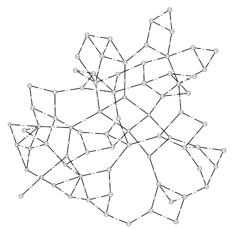
`sol_report_investment_summary(…)`



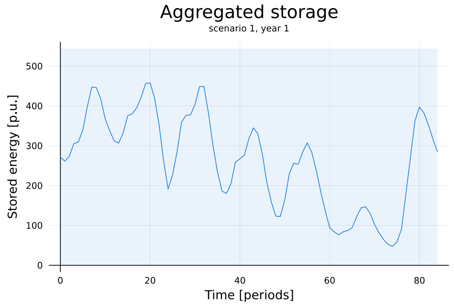
`sol_report_power_summary(…)`



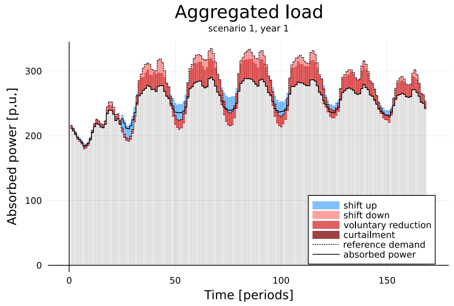
`sol_report_graph(…)`



`sol_report_storage_summary(…)`



`sol_report_load_summary(…)`



Outputs:

- **DataFrame** (returned value)
- numerical results as **CSV** files
- **plots** in any format supported by Plots.jl

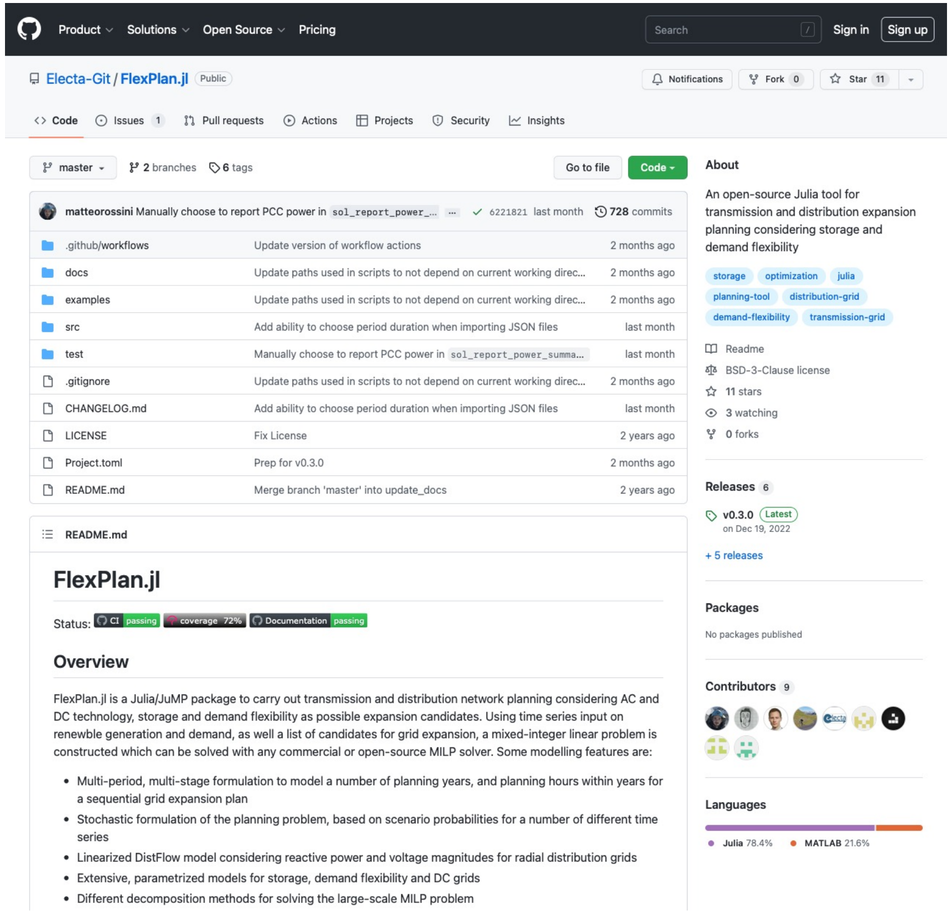
These functions are not part of FlexPlan module to avoid unwanted dependencies. Include them with:

```julia
import FlexPlan as _FP
const _FP_dir = dirname(dirname(pathof(_FP)))
include(joinpath(_FP_dir,"test/io/sol.jl"))
```

and import required packages.

References

FlexPlan



https://github.com/Electa-Git/FlexPlan.jl

To get started, see /examples/ and /test/scripts/ directories.
Documentation: https://electa-git.github.io/FlexPlan.jl/dev

FlexPlan.jl package

H. Ergun, M. Rossini, M. Rossi
FlexPlan.jl – An open-source Julia tool for holistic transmission and distribution grid planning
accepted to OSMSES 2023 – Aachen (DE), 27-29/03/2023

T&D decoupling heuristic

M. Rossini, M. Rossi, D. Siface
A surrogate model of distribution networks to support transmission network planning
submitted to CIRED 2023 – Rome (IT), 12-15/06/2023

Contact authors

Hakan Ergun – hakan.ergun@kuleuven.be
Matteo Rossini – matteo.rossini@rse-web.it

(full list of authors on repository website)

FlexPlan

FlexPlan-Project.eu

This presentation reflects only the author's view and the Innovation and Networks Executive Agency (INEA) is not responsible for any use that may be made of the information it contains.