Funded by the European Union

PyPSA-SMS++

Large-Scale Planning Under Uncertainties

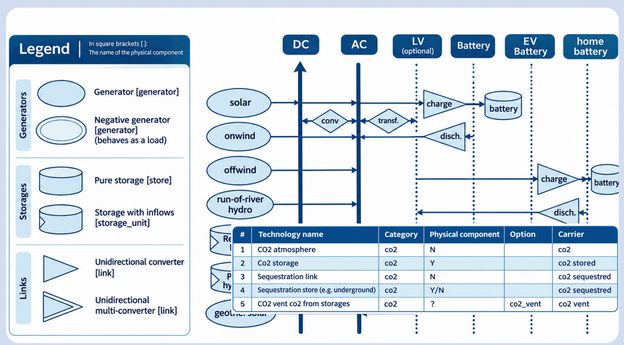
Objective

The **RESILIENT project** develops open-source tools for planning the future **European sector-coupled energy system under uncertainty**. Within the project, the **University of Pisa** elevates **PyPSA energy system models** with **mathematical decomposition offered by the SMS++** tool to make complex **stochastic models computationally tractable**. University of Pisa also supports EDF's French demonstration study

PyPSA for Effective Energy Modeling

PyPSA features

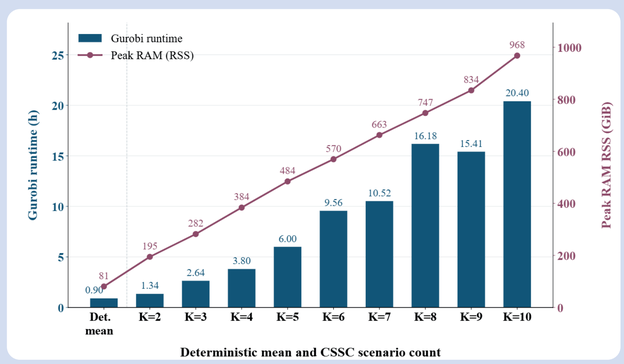
- **Flexible representation of energy systems**
  Energy systems represented by equations
- **Flexible system resolution**
  Flexible temporal and spatial resolution
- **Agnostic to solvers**
  Support for various standard solvers
- **User friendly**
  Effective interface with Python
- **Open-source:** https://github.com/PyPSA/pypsa



System characterization

Project activities

- **Characterization of Energy Systems**
  Prioritize component mapping into SMS++
- **Analysis of mathematical representation**
  Identify component-to-block mapping for SMS++
- **Case study development**
  Prepare and evaluate the role of new techniques



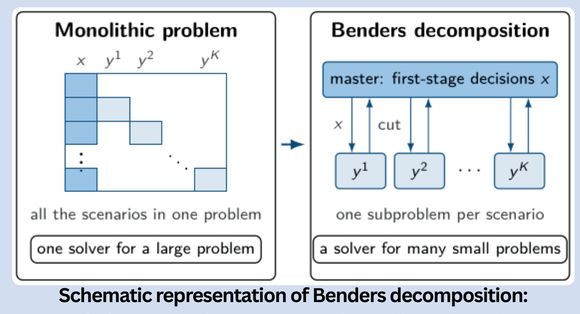
Increase in computational requirements for stochastic problems without decomposition

PyPSA brings accessible energy modelling to advanced algorithms in SMS++

SMS++ for Mathematical Optimization

SMS++ features

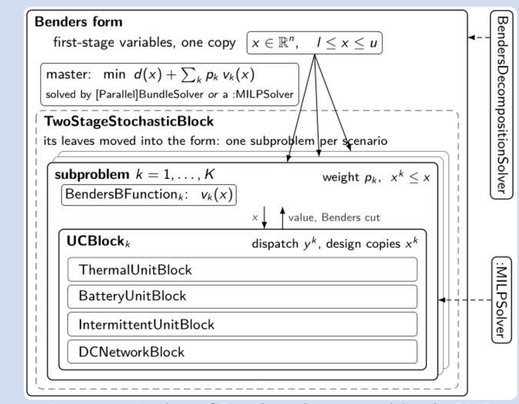
- **Expose structure of energy problems**
  Block-structured problem representation
- **Unlocks flexible nested decomposition**
  Decomposition algorithms (Benders, SDDP, ...)
- **Specialized high-performance solvers**
  Specialized solvers for some Block (DP, ...)
- **Fast, robust and parallel execution**
  High-performance multithreaded C++-20
- **Open-source:** https://gitlab.com/smspp/smspp-project



Schematic representation of Benders decomposition: the monolithic problem is decomposed in multiple subproblems that are solved iteratively multiple times till convergence.

Project activities

- **Implement major missing PyPSA features**
- **Support Two-Stage Stochastic Problems**
- **Implement Decomposition techniques**
  - Benders decomposition
  - Advanced InvestmentBlock
  - BundleSolver 2.0
  - PrimalProximalHeuristic
- **Easy installation** (conda , cmake, script, ...)
- Since 12/23, across the 16 repositories (8 new), +5,191 commits, +244k lines of code (190k to 435k)



A representation of Benders decomposition in SMS++ for a stochastic capacity-expansion and dispatch problem

SMS++ brings advanced decomposition to energy models in PyPSA

PyPSA–SMS++ Conversion Workflow

PyPSA model
the input model, with the solution parsed back

SMS++ as PyPSA solver
user interface: model creation, I/O, ... | standard optimization *or* **SMS++ interface**

Bidirectional conversion PyPSA to SMS++

pypsa2smspp
translates PyPSA objects into pySMSpp structures

pySMSpp
Python interface to edit and optimize SMS++ objects

Problem netCDF4
Configuration .txt
Solution netCDF4

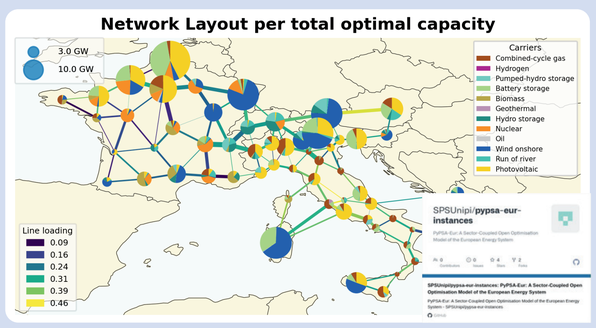
SMS++ optimization
solution approach set by the configuration file

Selecting the **"smspp" solver** in PyPSA automatically converts the model to SMS++, solves it using a user-configurable solution approach, and imports the results back into PyPSA.
The workflow relies on two open-source packages (pip/conda): **pypsa2smspp** for bidirectional model translation and **pySMSpp** for interfacing with SMS++ netCDF files, developed within the project

PyPSA–SMS++ Unlocks Unprecedented Decomposition Opportunities in Energy Modelling

Case PyPSA-Eur

-80% in computation time and -30% in memory usage

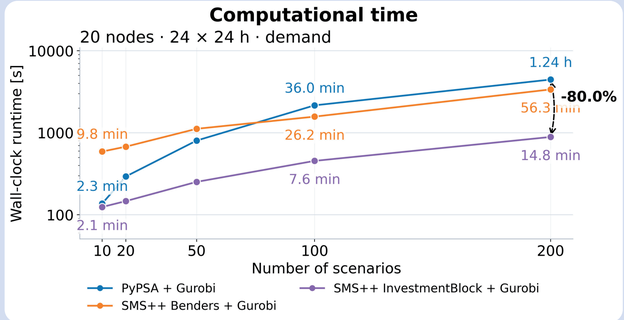


Tested on realistic stochastic instances from pypsa-eur-instances, a soft fork of PyPSA-Eur

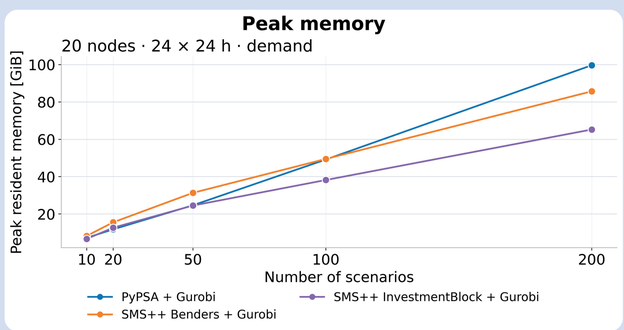
- up to 200 scenarios with renewable and load uncertainty
- up to 50 nodes
- up to 120 representative days per year

Pipeline at github.com/SPSUnipi/PyPSA-eur-instances

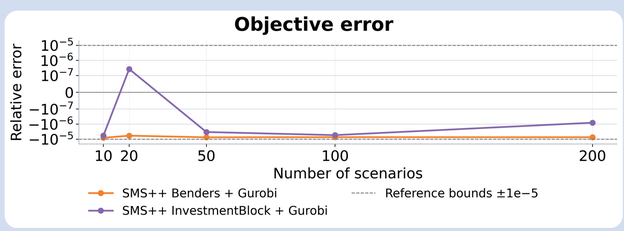
- **Up to 80% less computing time** than PyPSA-eur default Gurobi configuration (32 threads)
- **Decomposition outperforms the monolithic solve:** SMS++ decomposition techniques combined with Gurobi beat Gurobi alone
- **Fewer threads, still faster:** 15.6min on 8 threads vs 1.2h for Gurobi on 64 (200 scenarios)



- **Up to 30% less memory** than PyPSA-eur default Gurobi configuration (32 threads).
- **Memory-efficient as threads scale:** SMS++ keeps the lowest memory use at every thread count, with no loss of accuracy

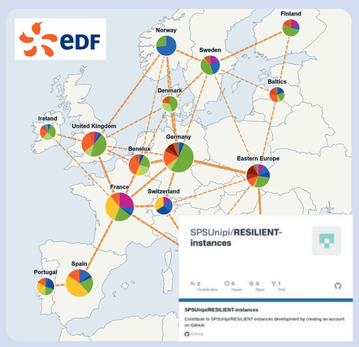


- **Accurate:** SMS++ solutions converge always within 1e-5 tolerance



Case ManOeuVRE

full-open-source SMS++ faster than Gurobi

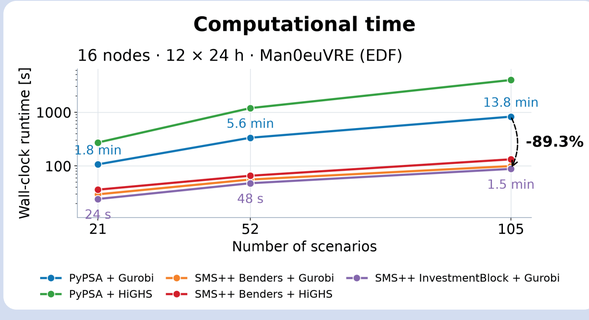


Tested on stochastic capacity expansion instances produced using the EDF instance generator using

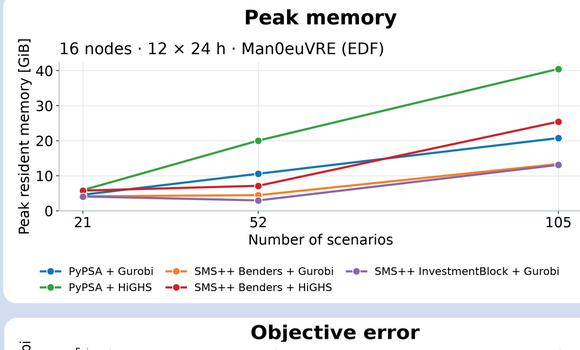
- 21, 52 and 105 scenarios
- 16 European regions
- 12 representative days

Pipeline at github.com/SPSUnipi/RESILIENT-instances

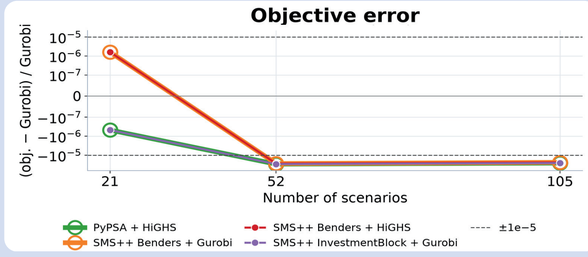
- **Up to 89% less computing time** than PyPSA + Gurobi: 88 s vs 826 s with 105 scenarios
- **Fully open source SMS++ is faster**: SMS++ Benders with HiGHS is 6× faster than PyPSA + Gurobi
- **Decomposition outperforms the monolithic solve:** SMS++ Benders with HiGHS is 30× faster than PyPSA + HiGHS (133 vs 3971 s)



- **Open-source SMS++ Benders + HiGHS** requires **similar memory** to PyPSA + Gurobi: 25 vs 20GiB
- **SMS++ InvestmentBlock + Gurobi saves 37% memory** compared to PyPSA + Gurobi: 13 vs 21GiB



- **Accurate**: SMS++ solutions converge always within 1e-5 tolerance



This work was supported by Italian Ministry of University and Research (MUR) under the RESILIENT project - CETPartnership – Joint Call 2022 TRI1 - Call module 1.1: PowerPlanningTools CUP I53C23002650007 co-funded by the European Union under grant agreement no. 101069750.