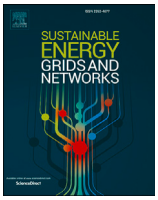
Sustainable Energy, Grids and Networks 44 (2025) 101893

Contents lists available at ScienceDirect

Sustainable Energy, Grids and Networks

journal homepage: www.elsevier.com/locate/segan



Understanding the performance impact of a massively parallel solver for energy system optimization models – a computational experiment using the PIPS-IPM++ solver for REMix instances

Manuel Wetzel*, Karl-Kiên Cao, Shima Sasanpour

German Aerospace Center (DLR), Institute of Networked Energy Systems, Stuttgart, Germany

HIGHLIGHTS

- Systematic application of the massively parallel solver PIPS-IPM++ to large-scale capacity expansion problems from the REMix framework reducing the time to solve by about one order of magnitude.
- Identification of the total number of capacity expansion decisions as the primary driver of the computational burden in linear ESOM instances.
- Generation of insights into the optimal decomposition strategy and configuration of PIPS-IPM++ depending on the problem instance to be solved.
- Enable researchers to leverage massively parallel solver on high-performance computing infrastructure, avoid limitations from shared memory, and allow for larger optimization models.

ARTICLE INFO

Keywords:
Energy system analysis
Capacity expansion planning
High performance computing
Linear programming
Massively parallel solvers
Decomposition strategies

ABSTRACT

The complexity in the design of future integrated energy systems is reflected in the modeling tools used to analyze the interactions and synergies between energy technologies. As real-world systems become more interconnected, the complexity of model representations increases, resulting in a greater computational burden to obtain optimal solutions. Several approaches can address this challenge, including making trade-offs between model dimensions, using methods to reduce complexity, performing mathematical decomposition, and applying massively parallel solvers. While most of these approaches have been extensively studied, the application of massively parallel solvers has barely been explored due to their novelty. The main advantage of this approach is that it is well-suited to take advantage of modern high-performance computing infrastructure. Therefore, a more systematic evaluation of the types of model instances and decomposition strategies that can benefit from massively parallel solvers remains necessary. In this study, we identify capacity expansion decisions as the primary driver of computational complexity, particularly within the REMix framework, and demonstrate how to effectively leverage the underlying problem structure of these models. In a computational experiment we evaluate the performance of PIPS-IPM++ against a state-of-the-art interior-point solver. The results show a significant reduction in the total required wall-clock time of about one order of magnitude, as well as a reduction in required computational resources. Our findings provide modelers with the necessary methods and capabilities to solve previously intractable, large-scale problems, thereby increasing the level of detail and explanatory power of energy system optimization models.

1. Introduction

Energy system optimization models (ESOMs) have become widely used tools to gain insight into the interactions between energy technologies and flexibility options in future energy systems. However, accurate modeling of these systems requires a sufficient temporal and spatial resolution to capture the variability of renewable energy sources [1], transmission bottlenecks [2] and need for electrical energy storage expansion [3]. Hence, contemporary large-scale ESOMs translate into mathematical optimization problems involving up to hundreds of millions of variables and constraints, resulting in a high computational

* Corresponding author at: Curiestraße 4, 70563 Stuttgart, Germany.
Email addresses: manuel.wetzel@dlr.de (M. Wetzel), karl-kien.cao@dlr.de (K.-K. Cao), shima.sasanpour@dlr.de (S. Sasanpour).

https://doi.org/10.1016/j.segan.2025.101893
Received 29 January 2025; Received in revised form 23 June 2025; Accepted 1 August 2025
Available online 5 August 2025

List of abbreviations

AC	alternating current
AC-OPF	alternating current optimal power flow
ESOM	energy system optimization model
FLOP	floating point operation
GPU	graphics processing unit
HPC	high-performance computing
IP	interior-point
IAM	integrated assessment model
JSC	Jülich Supercomputing Centre
LP	linear programming
MILP	mixed-integer linear programming
NUMA	non-uniform memory access
RAM	random access memory
UC	unit commitment
VRE	variable renewable energy

burden. This computational burden is evident in the fact that most modelers opt for clustering and aggregation techniques in either the spatial or temporal resolution [4] as well as the large field of research to systematically improve complexity reduction techniques [5]. The maximum size of ESOMs that can still be solved in a reasonable time frame is mainly determined by the performance of the utilized algorithm and the available hardware. This combination gives an overall threshold, which determines the maximum computational complexity that can still be addressed. However, this reasonable time frame is subjective for each modeler and depends on additional factors like the overall number of runs for a given analysis [6]. For an individual optimization run, there are various approaches that affect the computational burden – trade-offs between model dimensions, aggregation techniques and heuristics, mathematical decomposition, and application of massively parallel solvers. We start by discussing the different influences each of these approaches has on the computational burden and drawbacks that need to be considered by the modeler.

Trade-offs between model dimensions allow one dimension's level of detail to increase at the expense of another dimension's level of detail. For example, simple aggregation techniques can be used to achieve this goal [7]. When using state-of-the-art interior-point (IP) solvers, such shifts allow focusing on a specific aspect of the model while staying within the threshold for computational complexity. More specialized heuristics allow more efficient trade-offs through intelligent reductions in model dimensions [5]. However, like the trade-offs, this approach is also subject to the same overall computational threshold imposed by the available algorithms and hardware. In contrast to the aforementioned approaches, mathematical decomposition allows the computational threshold to be increased by iteratively solving multiple smaller subproblems [8]. Similarly, massively parallel solvers exploit a fundamental block structure of the optimization matrix. This also effectively results in the iterative solving of subproblems but has the advantage that no explicit reformulation of the model logic is required. Both mathematical decomposition and massively parallel solvers can be deployed to distributed memory architectures, enabling the efficient use of high-performance computing (HPC) infrastructure.

1.1. The increasing complexity of ESOMs

Energy system modeling has been widely used to inform policy makers about decisions for a successful energy system transition [9]. The fundamental challenge of decarbonizing energy systems lies in integrating renewable energy resources while ensuring an affordable, secure and sustainable energy supply. All three aspects can be modeled using ESOMs, which enable capacity expansion planning and economic dispatch to be integrated into a single optimization problem [10]. While this integration of capacity expansion planning significantly increases computational complexity [11], it also offers the most valuable insights for planning the energy transition.

One of the most prevalent challenges in modeling large-scale energy systems is the trade-off between two fundamental requirements – maintaining a comprehensive system perspective while at the same time providing a sufficient level of detail. This level of detail can affect either the spatial, the temporal or the technological resolution of the model [4]. Reducing the spatial resolution, for example, can affect the preservation of critical bottlenecks in the network topology, such as high-voltage transmission lines. Consequently, the model may produce overly optimistic solutions. Conversely, increasing the spatial resolution increases computational complexity and potentially leads to time constraints for real-time applications, such as market clearing or network redispatch. There are a large number of different approaches and various degrees of complexity for different formulations of optimization models, but all of them are constrained by the limitation of computational complexity when applying standard IP solvers [8].

Several crucial factors must be considered for future energy systems, requiring a minimum level of detail to be addressed by the ESOM across the different dimensions. In terms of the temporal dimension the correct representation of the availability of variable renewable energy (VRE) sources plays a fundamental role in future energy systems [1]. Due to their variability, hourly and sub-hourly resolutions are required to accurately estimate the need for additional flexibility options, such as transmission networks and storage technologies. Consequently, many ESOMs discretize the temporal dimension of the model into hourly time steps [12].

Similarly, exploring transformation pathways requires the integral consideration of long-term time horizons. For example, this enables the endogenous modeling of optimal CO_2 budget allocation or the inclusion of learning rates for energy technologies [13]. Therefore, a fundamental design choice for ESOMs is whether to include transformation pathways as multi-year optimizations or to use (successive) single-year optimizations. While single-year optimization models focus on the hourly interactions of technologies providing system flexibility, multi-year models typically use representative time slices [8].

At a spatial level, the increasing number of distributed generators and new power consumers requires more accurate modeling of power flows. This modeling must bridge the gap between modeling grid congestion at a regional level and accounting for long-distance energy transmission due to the spatial balancing effects of renewable energy sources [14]. This requires a broad geographical scope, detailed spatial resolution, and accurate representation of network losses, in order to adequately assess energy transmission and identify the need for network expansion [15].

In terms of energy technologies, the expansion of the modeled system is driven by the coupling of energy sectors to decarbonize the heating and transport sectors, primarily through electrification or the provision of synthetic fuels based on renewable energy sources [16]. This introduces a wide range of new technologies and energy vectors to be additionally considered in ESOMs. Similarly, considering hydrogen blending in natural gas pipelines [17] or hydrogen offshore hubs [18] further increases the complexity of capacity expansion decisions in ESOMs.

While reducing the spatial and temporal resolution can often be achieved with limited loss of detail, introducing more sectors and technologies can significantly impact the design of the overall energy system. However, there is still the possibility of aggregating similar technologies [19].

1.2. Heuristics and complexity reduction

As a consequence of this growing complexity in all three dimensions, the size of the model in terms of variables and constraints, as well as the time required to solve it, increases. Many approaches to dealing with the resulting computational limitations can be summarized as heuristics. In this context, heuristics aim to provide either simplifications across one or more dimensions, or multi-stage approaches, where an initial approximation is performed at a low resolution, followed by a consecutive run at a higher resolution. This trade-off between modeling capacity expansion pathways and ensuring a high temporal resolution can be addressed by running two models sequentially. For instance, in Gerbaulet et al. [20], first the capacity expansion is modeled using a low temporal resolution to estimate the overall energy system design. Then, the system's operations are validated by conducting an hourly economic dispatch of power plants at a high temporal resolution.

For models with an hourly resolution in particular, the time dimension is one of the main drivers of complexity and therefore typically the first dimension to be addressed using complexity reduction. This involves reducing the temporal dimension sufficiently (e.g., by identifying representative time periods or merging consecutive hours), while preserving critical information from the input data with an hourly resolution. Kotzur et al. evaluate the impact of several time series aggregation techniques on energy system design (by comparing the resulting objective values) and computational times compared to an ESOM using the full time series data [21]. They conclude that the choice of a specific algorithm does not have a significant impact on the accuracy of a model. However, the authors emphasize that the approaches studied are not recommended for modeling energy systems that rely heavily on energy storage. This finding is also confirmed by Buchholz et al., who use complexity reduction approaches that do not explicitly consider long-term energy storage [22]. The authors point out that investments in flexibility options and thus total system costs are systematically underestimated. Raventos et al. study two different time series aggregation techniques, reporting that chronological clustering better accounts for the operational patterns of energy storage [23].

Similar findings apply to the spatial scale. In a previous study, we demonstrated how such reductions can lead to an underestimation of the share of investment costs for transmission investments, as well as a shifts towards electricity generation with low marginal costs [4]. Although spatial reduction techniques are state-of-the-art in power systems engineering, where network equivalents are used to simplify areas around the study area (i.e., the area of responsibility of a network operator) [24], such approaches are of limited value for ESOMs, which often do not focus on a specific distribution network. The key challenge, therefore, is to define clusters that can be treated as interconnected regions with no internal bottlenecks. Therefore, network clustering approaches are usually used to identify the transmission lines that are most important to maintain in a reduced network representation [25].

In summary, heuristics that aim to reduce the complexity of ESOMs, particularly with regard to the temporal scale, have the potential to significantly reduce computational effort while providing results that are sufficient for specific use cases. Buchholz et al. report reductions in computation time of up to 98 % when ignoring long-term energy storage options [22]. However, systematic evaluations in the literature have mainly been performed with comparatively small ESOMs, since two conditions must be met for such studies. Firstly, a full-scale model must be solved as a benchmark for the complexity reduction method. Secondly, tuning the algorithm and demonstrating scaling effects require a sufficiently large number of runs. To ensure that an algorithm's performance is comparable to the reference solution, these studies must be performed on identical hardware. Therefore, the cumulative computation time for an appropriate benchmark study remains an issue, particularly for systematic studies involving large-scale models [26]. For a broader overview of the various methods of reducing the complexity of energy system models, the reader is referred to [5].

1.3. State of running ESOMs on HPCs

All of the approaches presented so far use single-node, shared-memory architectures to solve the optimization problem. Historically, this approach has worked quite well since available hardware has mainly offered speed increases based on processor clock rates. However, this trend shifted in the early 2000s with the introduction of multi-core processors.[1] Consequently, algorithms had to utilize multi-core architectures to enhance performance. Further scaling of computing resources was achieved by switching from single-node to multi-node computing clusters. This utilization of multiple cores across one or multiple compute nodes can be classified as either parallelization on shared memory architectures (e.g., via OpenMP, Open Multi-Processing, a standard interface for parallel computing on shared memory, multi-core architectures) and parallelization across distributed memory architectures, also referred to as distributed computing (e.g., via MPI, Message Passing Interface, a standard interface for coordinating processes across distributed memory, multi-node architectures). On distributed computing architectures, modelers take advantage of the large number of identical processors for parallel computing, but at the cost of requiring algorithms that are able to utilize the distributed memory. While distributed computing is a quite broad term, modern HPC clusters additionally provide fast communication networks and data storage.

Even without parallel software, HPC can still be used to scale up the optimization of multiple independent model instances in an *embarrassingly parallel* workflow. While this approach is useful for analyzing a broad scenario space, individual model instances still must fit within the hardware limitations of a single compute node. For instance, Sharma et al. [27] perform benchmark analyses with models of the TIMES family [28] and study optimal strategies to allocate multiple independent model instances across compute nodes. The authors also highlight advantages of applying HPC-typical software, such as workload management tools, to perform multiple parameter variations and solve a set of scenarios.

However, solving an ESOM that exceeds the memory capacity of a single compute node requires decomposing the optimization problem into a set of subproblems. Accordingly, such applications must handle the additional communication overhead between subproblems. Various approaches have been presented for operations scheduling [29], often using well-known mathematical decomposition strategies such as Benders decomposition [30] or Dantzig–Wolfe decomposition [31]. Gong et al. use an Augmented Lagrangian Relaxation applied to a combined unit commitment (UC), optimal power flow, and transmission switching problem [32]. They also decompose a coordinated operations planning problem for the gas and electricity sectors in this manner [33]. Göke et al. apply Benders Decomposition to a stochastic two-stage energy system optimization problem where additional enhancement strategies are used to improve the performance of the algorithm [34]. Similarly, to improve the parallelization of Benders Decomposition, Pecci et al. decompose the subproblems along the temporal dimension [35].

In the special case of two-stage stochastic optimization the communication overhead is minimal, making such problems early candidates for the application of HPC. One of the main advantages of stochastic optimization problems is their fundamental structure: the objective is to identify a set of decisions applicable to a broad range of uncertain boundary conditions. If a different solution is allowed for each boundary condition, solving the entire optimization problem is equivalent to solving multiple small optimization problems. The goal, however, is to find a single solution that applies to all problems, so the decision variables span across multiple stochastic scenarios. These variables link the problem structure, preventing the solution of parts individually. Therefore, they are referred to as "linking" or "complicating" variables. Linking

[1] https://github.com/karlrupp/microprocessor-trend-data

variables automatically emerge when the entire problem is decomposed in a particular way.

In terms of application, Papavasiliou et al. decompose a short-term UC problem with a time horizon of 24 hours by Lagrangian relaxation and solve 1,000 independent stochastic scenarios [36]. The authors report reducing the computation time from 120 hours down to 17.5 hours by exploiting the parallelization potential of 1000 processors. In a more recent study Ávila et al. [37] introduce a stochastic optimization problem for transmission system expansion in the European electricity network combining two different decomposition approaches based on the L-shaped method, and bring it to HPC. In contrast to classical decomposition techniques, Lubin et al. present a novel massively parallel interior point solver, PIPS-IPM, which decomposes the stochastic optimization problem along the scenario dimension [38]. This solver is used by Petra et al. to determine the optimal dispatch of power plants in the federal state of Illinois to adequately reflect the probability distribution for electricity generation from wind turbines [39]. The major advantage of these approaches is their scalability, which is one of the most important aspects for distributed memory architectures. Similar scalability was demonstrated by Wang et al. [40], who presented the HiOP solver for addressing nonlinear stochastic optimization problems. This solver was used in one of the first successful attempts at exascale computing with a large number of graphics processing units (GPUs) in the field of power system analysis. More recently, Shin et al. [41] and Swirydowicz [42] both applied massively parallel solvers on GPUs for the optimization of large-scale nonlinear alternating current optimal power flow (AC-OPF) problems. Also outside the field of optimization, novel approaches addressing the increasing complexity of transmission expansion planning are emerging, using convolutional neural networks [43].

However, most of the previously mentioned approaches require either a limited number of linking variables or a limited number of linking constraints for the decomposition of the mathematical formulation. To solve a more generalized type of optimization problem that includes both linking variables and constraints, Rehfeldt et al. [44] extended the PIPS-IPM solver by Lubin et al. to handle both types. For various model instances, they report a speedup factor of up to 67 compared to the best standard IP solver. However, the instances addressed in the study are based on very stylized energy system models and real-world systems for pure economic dispatch models. The authors stress the need to further investigate the solver's performance with model instances showing stronger interconnection between individual subproblems. Similarly, Panos and Hassan apply PIPS-IPM++ to the TIMES model family for two instances and observe a speedup of 4 compared to a standard IP solver [45]. This wide range already indicates a strong connection between the types of optimization problems and the potential for speedups, and a lack of understanding of decomposition approaches and their resulting performance. To date, the PIPS-IPM++ solver has also not yet been applied to large-scale hourly resolved ESOMs, which include a large number of capacity expansion decisions.

1.4. Towards a systematic understanding of computational performance in ESOMs

In summary, solving large-scale ESOMs, which comprise a multitude of capacity expansion decisions is one of the most challenging aspects of outlining pathways for the energy transition. The models aim to identify potential investment opportunities in generation, storage and transmission capacities. This in turn requires methods that can efficiently handle both linking variables and linking constraints. Although several heuristics have been applied to reduce the problem size overall, these heuristics either lead to information loss or require highly model-specific tuning, and they are usually not generalizable. Massively parallel solvers have proven useful in bringing ESOMs to HPC and avoiding hardware limitations. However, their use has been limited to models with a small number of coupling elements. Here, PIPS-IPM++ presents an option to accelerate or even enable the solution of large-scale ESOMs that combine both operational and capacity expansion planning. To better understand the types of problems for which the PIPS-IPM++ solver can be effectively applied, we present its first application to a broad set of model instances to study the overall scaling behavior using real-world instances involving a large number of investment decisions. Thus, our study raises the general question of whether we can obtain similar benefits from applying PIPS-IPM++ to capacity expansion problems and provides the following contributions:

1. We analyze the underlying problem structure for a scalable REMix model instance with varying degrees of capacity expansion planning and economic dispatch, and assess the impact of its complexity on the time and memory needed to solve the problem.
2. We outline the methodological steps that enable modelers to decompose generic ESOMs and solve them using PIPS-IPM++ on distributed memory hardware. To this end, we discuss how different decomposition strategies impact linking elements and the potentially achievable parallelization.
3. We perform a systematic computational experiment on various scalable model instances derived from the REMix framework to compare the performance of PIPS-IPM++ against a state-of-the-art standard IP solver with respect to obtainable solving time speedup, memory reduction, and scaling behavior.

2. Methods

Our research is fundamentally based on a computational experiment. In Sections 2.1 and 2.2, we first present the REMix framework used in this study as well as the parallel solver PIPS-IPM++. Section 2.3 then outlines the necessary block structures, while Section 2.4 presents a systematic approach for the implementation of the decomposition. Section 2.5 then provides a model-specific example and highlights different decomposition strategies for ESOMs.

2.1. The REMix framework

REMix is an open-source framework for energy system optimization modeling.[2] It optimizes the expansion and operation of the energy system from a central planner's perspective with perfect foresight by minimizing the total system costs. The framework consists of a few generalized modules that can represent various technologies across multiple sectors. For most applications, one target year with hourly resolution is formulated as linear programming (LP). However, more advanced features are available such as mixed-integer linear programming (MILP) for UC and investment planning, Pareto fronts, myopic and pathway optimization [46]. Examples of different use-cases of the REMix framework have been published in previous studies [47,48].

As discussed in Section 1.1 from an abstract perspective, REMix is likewise structured in various model dimensions:

$t \in T$	time steps
$y \in Y$	year intervals
$n \in N$	model nodes
$l \in L$	transfer links
$z \in Z$	network cycles
$q \in Q$	energy carriers
$a \in A$	converter activities

[2] https://dlr-ve.gitlab.io/esy/remix/framework

$p \in P$ technologies

$P_C \subset P$ converter technologies

$P_S \subset P$ storage technologies

$P_L \subset P$ transfer technologies

$i \in I$ system indicators

$I_{obj} \subset I$ indicator declared as objective function

The main model variables can be divided into two categories: capacity expansion decisions, which relate to infrastructure investment decisions, and dispatch decisions, which describe how available infrastructure is utilized throughout the year:

$x \geq 0$ number of available units / links

$\Delta x^+ \geq 0$ units / links expanded

$\Delta x^- \geq 0$ units / links decommissioned

$d \geq 0$ dispatch of converter

$e \geq 0$ level of storage

$f^+ \geq 0$ flow along transfer link

$f^- \geq 0$ flow against transfer link

$g^+ \geq 0$ import of energy carriers

$g^- \geq 0$ export of energy carriers

j accounting indicator

Several additional parameters are needed to describe various characteristics of different technologies and the connectivity between transfer links and model regions:

D rated dispatch coefficients

E rated storage coefficients

F rated transfer coefficients

H rated transfer loss coefficients

M incidence matrix between transfer links and model regions

K minimum cycle basis matrix between transfer links and network cycles

J indicator coefficient matrix

V activity matrix for converter technologies

W weighting factor between year intervals

α availability factor

γ loss adjusted transfer correction factor

δ self-discharge rate of storage

λ technical lifetime

χ link-specific reactance

To define an objective function for optimization, various system indicators can be used, which are linked to both the capacity expansion and economic dispatch parts of the model. These indicators can be subdivided into those referring to individual years and those integrating results across multiple years, the latter of which is controlled by the weighting factor W for individual year intervals. Additional bounds can be defined for indicators that are not defined as the objective function to enforce system constraints, such as a limit on carbon emissions.

$$\min j_i \qquad \forall i \in I_{obj} \tag{1}$$

s.t.

$$\underline{j}_i \leq j_i \leq \overline{j}_i \qquad \forall i \in I \setminus I_{obj} \tag{2}$$

$$j_i = \sum_{y \in Y} W_{y,i} \cdot \Bigg(\sum_{n \in N, p \in P_C \cup P_S} J^{\text{build}}_{i,n,y,p} \cdot \Delta x^+_{n,y,p} + J^{\text{fix}}_{i,n,y,p} \cdot x_{n,y,p} + J^{\text{decom}}_{i,n,y,p} \cdot \Delta x^-_{n,y,p} \tag{3a}$$

$$+ \sum_{l \in L, p \in P_L} J^{\text{build}}_{i,l,y,p} \cdot \Delta x^+_{l,y,p} + J^{\text{fix}}_{i,l,y,p} \cdot x_{l,y,p} + J^{\text{decom}}_{i,l,y,p} \cdot \Delta x^-_{l,y,p} \tag{3b}$$

$$+ \sum_{t \in T, n \in N, p \in P_C, a \in A} J^{\text{var}}_{i,t,n,y,p,a} \cdot d_{t,n,y,p,a} \tag{3c}$$

$$+ \sum_{t \in T, l \in L, p \in P_L} J^{\text{flow}}_{i,t,l,y,p} \cdot \left(f^+_{t,l,y,p,q} + f^-_{t,l,y,p,q} \right) \tag{3d}$$

$$+ \sum_{t \in T, n \in N, q \in Q} J^{\text{import}}_{i,t,n,y,q} \cdot g^+_{t,n,y,q} + J^{\text{export}}_{i,t,n,y,q} \cdot g^-_{t,n,y,q} \Bigg) \qquad \forall i \in I \tag{3e}$$

The capacity expansion model of REMix can be described using the lower and upper bounds of the converter and storage units and of the transfer links,

$$\underline{x}_{n,y,p} \leq x_{n,y,p} \leq \overline{x}_{n,y,p} \qquad \forall n \in N, y \in Y, p \in P_C \cup P_S \tag{4}$$

$$\underline{x}_{l,y,p} \leq x_{l,y,p} \leq \overline{x}_{l,y,p} \qquad \forall l \in L, y \in Y, p \in P_L \tag{5}$$

the continuity equation considering existing units from the previous year's intervals and changes in the number of units due to new investment and decommissioning decisions,

$$x_{n,y,p} = x_{n,y-1,p} + \Delta x^+_{n,y,p} - \Delta x^-_{n,y,v} \qquad \forall n \in N, y \in Y, p \in P_C \cup P_S \tag{6}$$

$$x_{l,y,p} = x_{l,y-1,p} + \Delta x^+_{l,y,p} - \Delta x^-_{l,y,p} \qquad \forall l \in L, y \in Y, p \in P_L \tag{7}$$

and the equations enforcing the decommissioning of units and links after the end of their respective lifetime λ:

$$\sum_{y' \leq y - \lambda_v} \Delta x^+_{n,y',p} \leq \sum_{y' \leq y} \Delta x^-_{n,y',p} \qquad \forall n \in N, y \in Y, p \in P_C \cup P_S \tag{8}$$

$$\sum_{y' \leq y - \lambda_l} \Delta x^+_{l,y',p} \leq \sum_{y' \leq y} \Delta x^-_{l,y',p} \qquad \forall l \in L, y \in Y, p \in P_L \tag{9}$$

The number of available units limits the dispatch of different activities for converters,

$$\sum_{a \in A} d_{t,n,y,p,a} \leq V_{p,a} \cdot \alpha_{t,n,y,p} \cdot x_{n,y,p} \qquad \forall t \in T, y \in Y, n \in N, p \in P_C \tag{10}$$

the maximum level of storage reservoirs,

$$e_{t,n,y,p} \leq \alpha_{t,n,y,p} \cdot x_{n,y,p} \qquad \forall t \in T, y \in Y, n \in N, p \in P_S \tag{11}$$

and the flows of energy carriers throughout the network:

$$f^+_{t,l,y,p} \leq \alpha_{t,l,y,p} \cdot \gamma_{l,p} \cdot x_{l,y,p} \qquad \forall t \in T, y \in Y, l \in L, p \in P_L \tag{12}$$

$$f^-_{t,l,y,p} \leq \alpha_{t,l,y,p} \cdot \gamma_{l,p} \cdot x_{l,y,p} \qquad \forall t \in T, y \in Y, l \in L, p \in P_L \tag{13}$$

In addition to the flow constraints based on the network capacity, a direct current optimal power flow (DC-OPF) formulation is used to linearly approximate alternating current (AC) networks:

$$\sum_{l \in L} K_{z,l} \cdot \chi_{l,p} \cdot (f^+_{t,l,y,p} - f^-_{t,l,y,p}) = 0 \qquad \forall t \in T, y \in Y, p \in P_L, z \in Z \tag{14}$$

Both the availability of fuels and the exogenous demand for energy carriers are defined through the bounds on imports and exports across

the system boundary:

$$\underline{g}^{+}_{t,n,y,q} \le g^{+}_{t,n,y,q} \le \overline{g}^{+}_{t,n,y,q} \qquad \forall t \in T, n \in N, y \in Y, q \in Q \quad (15)$$

$$\underline{g}^{-}_{t,n,y,q} \le g^{-}_{t,n,y,q} \le \overline{g}^{-}_{t,n,y,q} \qquad \forall t \in T, n \in N, y \in Y, q \in Q \quad (16)$$

The nodal conservation of energy carriers is enforced by the balancing equation for all model regions and time steps. This equation includes inputs and outputs into the converters (17a), flows across transfer links in the network (17c) and associated losses, which are evenly distributed to both connected model regions (17d), the change in storage levels and their respective self-discharge (17b), and imports and exports of energy carriers across the system borders (17e). Note that for the storage level the reference to the previous time step $t-1$ behaves in a cyclical fashion in which the predecessor of the first time step is equal to the last time step of the set T:

$$\sum_{p \in P_C, a \in A} D_{p,a,q} \cdot d_{t,n,y,p,a} \quad (17a)$$

$$+ \sum_{p \in P_S} E_{p,q} \cdot \left(e_{t,n,y,p,q} - (1-\delta_p) \cdot e_{t-1,n,y,p,q} \right) \quad (17b)$$

$$+ \sum_{l \in L, p \in P_L} M_{n,l} \cdot F_{l,q} \cdot \left(f^{+}_{t,l,y,p} - f^{-}_{t,l,y,p} \right) \quad (17c)$$

$$- \sum_{l \in L, p \in P_L} \frac{1}{2} \cdot |M_{n,l}| \cdot H_{l,q} \cdot \left(f^{+}_{t,l,y,p} + f^{-}_{t,l,y,p} \right) \quad (17d)$$

$$+ \left(g^{+}_{t,n,y,q} - g^{-}_{t,n,y,q} \right) = 0 \qquad \forall t \in T, n \in N, y \in Y, q \in Q \quad (17e)$$

Applying the optimization to a single target year significantly simplifies the Eqs. (4)–(9). Similarly, if the expansion and decommissioning of power plants are not permitted, the available units $x_{n,y,p}$ and links $x_{l,y,p}$ can be treated as fixed variables and are reduced to constant factors in Eqs. (10)–(13), effectively resulting in a pure economic dispatch model. The equations shown are the minimum requirement to model the basic functionality of the general ESOMs and are even in the reduced form for a single target year sufficient to demonstrate the increase in computational complexity of the model with a fixed number of time steps $|T|$ and increasing the number of model regions $|N|$ and transfer links $|L|$.

2.2. The parallel solver PIPS-IPM++

As mentioned in Section 1, the original implementation of PIPS-IPM was specifically designed to solve stochastic optimization problems on HPCs. The algorithm effectively leverages the block structure of the underlying problem quite well and is adept at handling the linking variables that connect the stochastic dimension of optimization problems. For instance, in [38], these are the decision variables related to the power generation limits, which must be consistent across all scenarios. In contrast, the decomposition of arbitrary optimization problems generated by ESOMs leads not only to linking variables, but also to linking constraints. For instance, incorporating limits on annual CO_2 emissions as constraints in such models yields constraints across all model regions and time steps. For this particular reason, Rehfeldt et al. extended their version of PIPS-IPM, called PIPS-IPM++, to apply to linear programs with both linking variables and linking constraints and presented the improvements in [44]. Fundamentally, the solver uses a primal-dual IP method to solve the augmented system of an LP with arrowhead structure. This arrowhead structure can be created via a permutation of the optimization matrix, resulting in a structure in which most of the non-zero entries are close to the diagonal of the matrix and the remaining non-zero entries are close to the left and top borders of the matrix. This property is intrinsic to almost all ESOMs with a high temporal resolution, as discussed further in Section 2.3. The solver's increased parallelism and computational performance stem from the distributed computation of the local Schur complement for each block in the structure. For a more detailed description of the algorithm, see the initial introduction in [39] and [44].

Since PIPS-IPM++ can also use information about the adjacency of blocks, it is necessary to distinguish between globally linking constraints and variables and locally linking constraints and variables. Constraints that are contained in fewer than four blocks are considered locally linking and can be treated differently by the solver. All other constraints are considered globally linking. The same metrics apply to both local and global linking variables. Building upon this, Kempke et al. extended this version with a hierarchical approach that splits the linear system, allowing it to solve larger models with a large number of local constraints [49]. Initial benchmark analyses have demonstrated that PIPS-IPM++ can outperform standard IP solver software for both generic and real-world ESOMs [50]. Moreover, the reported results show that PIPS-IPM++ is even able to obtain solutions where other solvers cannot. However, the degree of parallelization and thus the ability to accelerate ESOMs depends on several parameters, such as the choice of underlying problem structure. As we demonstrate during the computational experiment, there are some fundamental aspects to consider based on the model instance at hand in order to obtain reasonable performance when applying PIPS-IPM++.

In contrast to standard IP solver, a slightly different workflow is required in order to benefit from distributed memory architectures and to obtain optimal performance with PIPS-IPM++. Fig. 1 shows a comparison of the two workflows and highlights the necessary considerations for parallelism. The HPC cluster used in the computational experiment has dual-socket compute nodes with 2 CPUs per node. Since each socket has its own memory controller, only half of the memory is locally accessible by each CPU. Accessing the remaining memory requires cross-socket communication, which decreases access speeds. This concept of non-uniform memory access (NUMA) needs to be considered when distributing any computations on HPC systems. In the case of the standard IP solver on shared memory (left side), only one compute node can be used. Furthermore, using more than half of the total memory can decrease the solver's overall performance due to cross-socket communication. For PIPS-IPM++ additional preparation steps are required. First, in this case, annotation describes the assignment of variables and equations to blocks, a topic discussed further in Section 2.4. Then, splitting and distributing the matrix is required to prevent parallel threads from simultaneously reading the same input file, which could cause a bottleneck. Up to 12 cores can be used for each block with the fastest memory access, or up to 24 cores can be used while still preventing cross-socket communication. Please note that the exact number of cores for each case depends on the CPU architecture and memory layout specific to the HPC cluster.

2.3. Matrix structures in ESOMs

The objective function of linear programs is typically formulated in the form $c^T x$ representing either a minimization or maximization of the decision variables x multiplied by a cost vector c. In the case of ESOMs this objective function is usually a minimization of total system costs as shown in Eq. (1). In addition to the decision variables the model contains equations in the form of $Ax \le b$ which limit the dispatch of power plants to their rated capacity, for example. The non-zero entries of the matrix A therefore provide useful insights into the connectivity between variables and equations. These non-zero elements can be used to represent any optimization problem visually (see Fig. 2(a)). In this visualization, each row represents an equation, and each column represents a variable. Black dots indicate non-zero entries and show whether a given constraint contains a given variable. The rows and columns of the matrix can be permuted without altering the underlying optimization problem, creating a block-structured matrix (see Fig. 2(c)). This block structure is required by the parallel solver PIPS-IPM++ and allows us to treat each of the blocks P_1 to P_k as a partial optimization problem. The linking

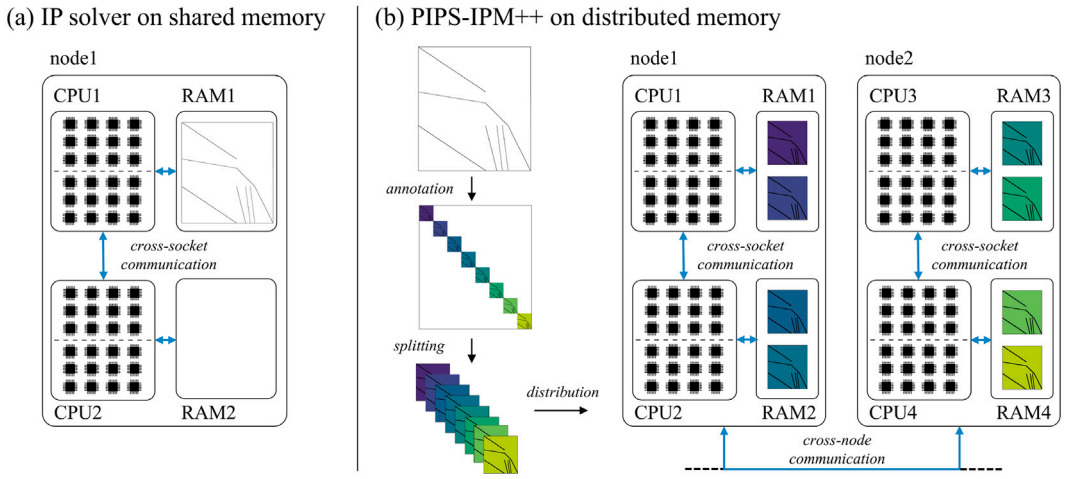


Fig. 1. Stylized representation of the differences in the workflow and hardware utilization between PIPS-IPM++ operating on distributed memory architectures and standard IP solvers. Blue arrows indicate communication pathways at the CPU, node, and cluster levels. The dashed line indicates CPU-internal core subdivisions. (a) IP solver on shared memory: Ideally, the matrix can be stored in the memory addressable by a single socket, otherwise cross-socket communication can decrease performance due to non-uniform memory access. (b) PIPS-IPM++ on distributed memory. The problem is first annotated, then split and finally distributed and solved on several nodes and CPUs. The blocks can be evenly distributed throughout the system, allowing for fast access and mitigating memory limitations.

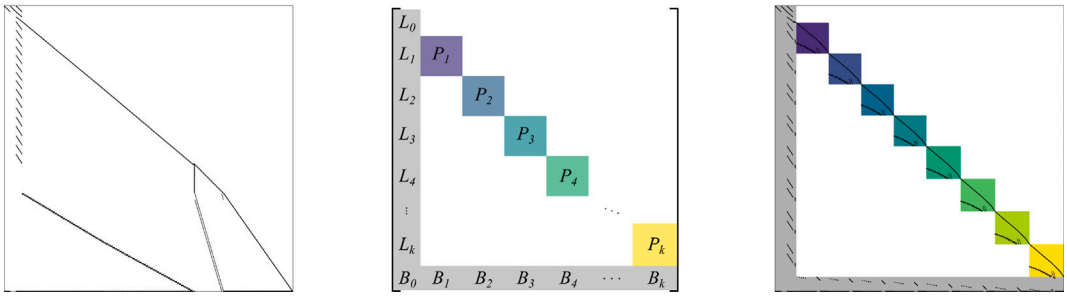


Fig. 2. Matrix structure representations for a small scale instance with 5 model regions and 16 time steps. (a) Representation of the matrix structure using the non-zero coefficients. (b) Required matrix structure for PIPS-IPM++ with colored blocks highlighting the diagonal sub-matrices and gray blocks highlighting sub-matrices for linking elements. (c) Permutation of the original matrix structure to match the matrix structure required by PIPS-IPM++.

variables and equations in L_0 to L_k and B_0 to B_k are used between parallel iteration steps to update the bounds for each partial optimization problem. In other words, each block represents a subproblem computed by one of many processes running in parallel. Thus, the greater the number of blocks, the greater the number of parallel processes that can be used.

However, the number of blocks – and thus the degree of parallelism – is limited by the additional elements connecting the blocks. Some variables appear in equations across multiple blocks, creating additional interlinkages. These linking variables are shown on the left-hand side of Fig. 2(b) as L_1 to L_k. Similarly, equations can contain variables from multiple blocks. These linking equations can be found at the bottom of the arrowhead matrix in blocks B_0 to B_k. Both types of linking structures impose an overhead cost on the parallel solver. In extreme cases, this communication overhead can outweigh the performance gain from parallel computing. Therefore, the goal of pre-structuring the matrix is to find appropriate decomposition strategies and optimize the number of blocks.

2.4. Block structure generation

Depending on the chosen decomposition strategy, this pre-structuring can result in various arrowhead structures. Despite the large number of possible decompositions, there are several points to consider for the effective application of PIPS-IPM++. First, the number of blocks corresponds to the maximum achievable parallelism. Therefore, we try to generate structures with a large number of diagonal blocks. Second, the more linking elements there are in the block structure, the greater the communication overhead during the non-parallel phase of the solver. These two effects create a trade-off between the speedup gained from parallelization and the slowing effect of communication, which is further explored in the computational experiment presented in Section 3.

In order to generate a specific block structure for PIPS-IPM++, a decomposition strategy must first be defined. This strategy comprises the mapping between two sets: the set of decomposition elements $\mathcal{D}$, which are related to the model variables based on the decisions the variables represent, and the set of blocks $\mathcal{K}$ we want to generate. In a temporal decomposition strategy, for example, the set of decomposition elements is the set of all time steps considered in the model. The mapping can then, for example, assign hours 1 to 24 to block 1, hours 25 to 48 to block 2, and so on. For the partitioning of the variables, each variable is checked to see if it is associated with a decomposition element. For example, a variable can represent the dispatch of a power plant in hour 12. If a variable is associated with an element of the decomposition strategy, it belongs to the parallel blocks P_1 to P_k and the corresponding linking equations B_1 to B_k, with the block k depending on the mapping from

decomposition elements to blocks. If a variable is not associated with an element in the strategy, then it belongs to the linking variables of the blocks L_0 to L_k and B_0. To determine the partitioning of the constraints, the annotation of the variables must be evaluated. If a constraint contains only linking variables, then it is an L_0-constraint contained in the top-left block, L_0. If a constraint contains only variables from a single block and optionally linking variables, it is assigned to the respective parallel block, P_1 to P_k. If a constraint contains variables from multiple parallel blocks, then it is a linking constraint contained in blocks B_0 to B_k. More formally, it can be described via the following partitioning algorithm.

1. **Definition of the decomposition strategy**
   With $\mathcal{D} = \{1, 2, \dots, D\}$ as the set of decomposition elements and $\mathcal{K} = \{1, 2, \dots, K\}$ as the set of blocks, $s : \mathcal{D} \to \mathcal{K}$ is the user-defined decomposition strategy assigning each decomposition element to a block.
2. **Annotation of the variables**
   With $\mathcal{X} = \{x_1, x_2, \dots, x_j\}$ as the set of all decision variables, the variables can be partitioned as follows:

$$\mathcal{X}_k = \{x \in \mathcal{X} \mid x \text{ is associated with } d \text{ such that } s(d) = k\} \quad \forall k \in \mathcal{K}$$
$$\mathcal{X}^{\text{linking}} = \mathcal{X} \setminus \bigcup_{k \in \mathcal{K}} \mathcal{X}_k$$

3. **Annotation of the constraints**
   With $\mathcal{C} = \{c_1, c_2, \dots, c_i\}$ as the set of all constraints and $\mathcal{X}_c \subseteq \mathcal{X}$ as the subset of decision variables with non-zero coefficients in constraint c, the constraints can be partitioned as follows:

$$\mathcal{C}^{L_0} = \left\{c \in \mathcal{C} \mid \mathcal{X}_c \subseteq \mathcal{X}^{\text{linking}}\right\}$$
$$\mathcal{C}_k = \left\{c \in \mathcal{C} \setminus \mathcal{C}^{L_0} \mid \mathcal{X}_c \subseteq \mathcal{X}_k \cup \mathcal{X}^{\text{linking}}\right\} \quad \forall k \in \mathcal{K}$$
$$\mathcal{C}^{\text{linking}} = \mathcal{C} \setminus \left(\mathcal{C}^{L_0} \cup \bigcup_{k \in \mathcal{K}} \mathcal{C}_k\right)$$

4. **Assignment of variables and constraints to matrix blocks**
   With $A_{c,x}$ as the coefficients in the original matrix A for constraint c and variable x, the matrix can be partitioned for the structure required by PIPS-IPM++ as follows:

$$L_0 = \left\{A_{c,x} \mid c \in \mathcal{C}^{L_0} \wedge x \in \mathcal{X}^{\text{linking}}\right\}$$
$$L_k = \left\{A_{c,x} \mid c \in \mathcal{C}_k \wedge x \in \mathcal{X}^{\text{linking}}\right\} \quad \forall k \in \mathcal{K}$$
$$P_k = \left\{A_{c,x} \mid c \in \mathcal{C}_k \wedge x \in \mathcal{X}_k\right\} \quad \forall k \in \mathcal{K}$$
$$B_0 = \left\{A_{c,x} \mid c \in \mathcal{C}^{\text{linking}} \wedge x \in \mathcal{X}^{\text{linking}}\right\}$$
$$B_k = \left\{A_{c,x} \mid c \in \mathcal{C}^{\text{linking}} \wedge x \in \mathcal{X}_k\right\} \quad \forall k \in \mathcal{K}$$

2.5. Decomposition strategies

Due to the high spatial and temporal resolution of the ESOMs, decomposing along these dimensions is the most promising way to generate a large number of blocks k:

- In the case of spatial decomposition, individual model regions can be treated as parallel blocks. However, these blocks are still connected by hourly variables that representing energy transmission between model regions and by constraints affecting multiple model regions, such as a global CO_2 budget.
- In the case of temporal decomposition, multiple time steps can be grouped into blocks, allowing for a uniform sizing of the parallel blocks. This type of decomposition establishes investment decisions as the linking variables and creates additional linking constraints for the storage level continuity and annual limitations, such as CO_2 budgets.

Table 1 shows the commonly used constraints and variables in ESOMs and the number of linking variables and constraints generated depending on the decomposition strategy. The last row provides an estimation of the scaling behavior of the linking elements. When using a spatial decomposition approach, the number of linking elements depends primarily on the size of the time dimension ($|T|$) and the total number of lines ($|L|$) between model regions. For a temporal decomposition, the number of linking constraints and variables depends mainly on the number of model regions ($|N|$) times the number of transmission and storage technologies ($|P_C| + |P_S|$) and the number of transfer links ($|L|$) times the number of transfer technologies ($|P_L|$). Note that if capacity expansion and decommissioning decisions are not allowed for a specific model region or technology, these decisions do not factor into the number of linking variables. Therefore, in the special case of a pure economic dispatch model, as described in Section 2.1, combined with a temporal decomposition, there are no linking variables.

A decomposition by time steps can be beneficial if the level of detail on the temporal scale outpaces the spatial scale. Additionally, when using HPC, it is essential to consider load balancing across multiple compute nodes and, consequently, to create similarly sized blocks. This is similar to two of the requirements of Lagrangian relaxation, in which the quality of the diagonal structure can be determined by granularity (a large number of blocks) and homogeneity (equally sized blocks) [51]. In the case of a spatial decomposition, each model region is usually connected to a various number of generation and demand technologies, resulting in unequally sized blocks. In contrast, the overall number of time steps can be equally and more flexibly assigned to the number of blocks chosen, allowing for a more thorough assessment of the scaling behavior between solver performance and the number of blocks. For this reason, and because the number of time steps is significantly greater than the number of model regions, we select the temporal decomposition strategy for the computational experiment in Section 3.

Table 1
Estimation of linking constraints and variables for spatial and temporal decomposition of a generic ESOM. The entries correspond to the factor by which the the number of linking elements increases per constraint or variable. k denotes the number of blocks, $|T|$ the number of time steps, $|N|$ the number of regions, $|L|$ the number of network links, $|P_{C,L,S}|$ the number of converter, network, and storage technologies, respectively.

		spatial decomposition	temporal decomposition
constraint	global annual constraint, e.g. CO_2 limit	1	1
	regional annual constraint, e.g. CO_2 limit	–	$\|N\|$
	conservation of energy carriers	–	$\|N\| \cdot \|P_S\| \cdot k$
	limitation of transfer flows	$\|T\| \cdot \|L\| \cdot \|P_L\|$	–
variable	available converter units	–	$\|N\| \cdot \|P_C\|$
	available storage units	–	$\|N\| \cdot \|P_S\|$
	available transfer links	$\|L\| \cdot \|P_L\|$	$\|L\| \cdot \|P_L\|$
	transfer flows	$\|T\| \cdot \|L\| \cdot \|P_L\|$	–
	scaling of linking elements	$\sim (2 \cdot \|T\| + 1) \cdot \|L\| \cdot \|P_L\|$	$\sim (1 + k) \cdot \|N\| \cdot \|P_S\| + \|N\| \cdot \|P_C\| + \|L\| \cdot \|P_L\|$

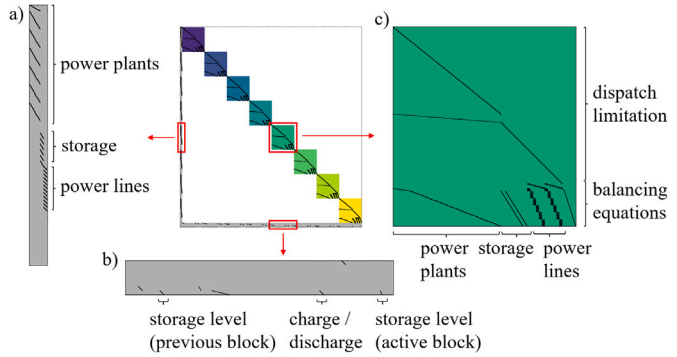


Fig. 3. Structure of a temporally decomposed optimization matrix which includes the following sub-matrices: (a) L_1 to L_k containing linking variables, (b) B_1 to B_k containing linking constraints, (c) P_1 to P_k containing the block-specific independent part of the decomposed optimization problem.

Fig. 3 shows an ESOM temporally decomposed into 8 blocks. Most of the equations and variables are non-linking, so they can be easily separated into parallel blocks, as indicated in the block matrix in (c). However, a few equations significantly impact the number of linking variables and constraints. These equations are:

- Globally linking constraint (e.g., CO_2-limit): An annual limitation on carbon emissions $\bar{J}_{CO_2}$ results in a globally linking constraint, since the emissions resulting from the utilization of fossil fuels in each time step and model region are summed up as modeled in Eq. (3a)).
- Globally linking variable (e.g., capacity expansion): If investment decisions are considered in the optimization the capacities of each technology represent linking variables. For example, they have an impact on the dispatch in each time step, as modeled in Eqs. (10)–(13) (see Fig. 3 (a)).
- Locally linking constraint (e.g., storage level): Storage technologies introduce locally linking constraints to the optimization. Equation (17b) enforces conservation of energy carriers and therefore also links the storage level of any time step to the storage level of the previous one (see Fig. 3 (b)).

3. Computational experiment

In this section, we will first provide a summary of the case study, followed by more details on the model instances for the computational experiment in Section 3.1. Second, in Section 3.2 we motivate the selection of performance indicators that are relevant in the context of assessing ESOMs in the context of HPC. Third, in Section 3.3 we describe the used hardware, software, and solver-specific configurations.

3.1. Model instance

This paper uses a model instance based on the work of Cao et al. [52] that focuses on the capacity expansion in the power sector. The model includes existing capacities for renewable and conventional power plants, as well as lithium-ion battery, pumped hydro storage, and the transmission grid. An annual CO_2 budget limits the operation of conventional power plants. Each year is modeled with full hourly resolution, resulting in 8,760 sequential time steps. At the highest spatial resolution, Germany is considered with 477 model regions representing substations and grid junctions including a total of 643 transmission lines. Additionally, the 11 neighboring countries are included with one model region each, and their electricity imports and exports to Germany are fixed based on historical time series. The base model with the highest spatial resolution and full expansion planning has been published as open data.[3] Although minor data updates have been implemented compared to the model instances used in this study, they do not affect the findings regarding computational performance.

Starting from the base model instance, we use hierarchical clustering [4] to derive a subset of spatial aggregations to investigate the performance of the solver for a broad set of model instances. For our computational experiment, we use eight levels of aggregation with 18 to 488 model regions. Additionally, we vary the degree of complexity by defining three levels of capacity expansion planning typical for energy systems research: i) disabled (economic dispatch optimization, *disp*), ii) enabled only for renewable power generators (generation expansion planning, *expRE*), and iii) enabled for renewable power generators, storage units, and transmission lines (full expansion planning, *expAll*). Combining these three levels with eight different spatial aggregations results in a total of 24 model instances. Table 2 lists the number of constraints, variables and non-zeros for each of these instances.

To decide which decomposition strategy to use, the rule of thumb from the Table 1 for estimating the number of linking elements can be applied to the full-size model. In the case of a spatial decomposition, the large number of time steps significantly increases the number of linking variables for power flows to about 22 million. With a temporal decomposition and an estimate of 100 time blocks, however, the number of linking constraints for the storage level is only about 0.1 million. This confirms our expectation from Section 2.4, that for REMix and other hourly resolved ESOMs, a temporal decomposition is advantageous when using PIPS-IPM++.

3.2. Performance indicators

For energy system modelers, one of the most important metrics is the wall-clock time required to solve a given optimization problem. A short wall-clock time significantly reduces the overall analysis workflow time because solving the problem is usually the most time-consuming step, except for initial data collection. During the setup of a given case study, for example, several workflow cycles are required to calibrate the model correctly and to detect and correct possible errors in the input data. Even if computation across multiple compute nodes is possible for a large-scale scenario analysis, the minimum run time of the analysis as a whole is limited by the longest single run.

Another key metric to consider is the amount of random access memory (RAM) required to store the optimization problem matrix on a shared memory system. Although there is some overhead caused by the optimization method itself, it scales quite linearly within a given problem class. As a result, energy system modelers face an upper bound on maximum problem sizes depending on available hardware. One way to circumvent this limitation is to simplify the problem's overall complexity through aggregation on the temporal, spatial, or technological scale, as previously discussed in Section 1.2. If simplifications cannot be made, or if the goal is to solve a reference run at the highest possible resolution, memory can still be a limiting factor for standard IP solvers using shared memory hardware. In contrast, decomposition approaches and block-structure solvers allow smaller chunks of the optimization matrix to be stored in memory.

From an HPC usage perspective, another key metric is the number of core hours required to solve the problem, in the following also referred to as compute. The number of core hours gives an indication of how much computational effort was required to solve any specific problem. In the context of HPCs, core hours usually also represent the budget that is granted in publicly funded compute clusters or paid for in commercial clusters. Therefore, it is important to develop an awareness of how much compute is required in order to run in a highly parallel environment.

Due to the overall scaling behavior of the optimization problem, all three factors – wall-clock time, RAM demand, and core hours – are highly correlated, but they offer trade-offs when using parallel solvers.

[3] https://gitlab.com/dlr-ve/esy/remix/projects/powger

Table 2
Model instances for the computational experiment and sizes of their corresponding optimization matrices.

regions	constraints [mio]			variables [mio]			dense columns [tsd]			non-zeros [mio]		
	disp	*expRE*	*expAll*	*disp*	*expRE*	*expAll*	*disp*	*expRE*	*expAll*	*disp*	*expRE*	*expAll*
18	2.60	3.00	3.24	2.44	2.83	3.07	0.00	0.10	0.16	8.57	10.46	12.07
30	3.96	4.63	4.98	3.71	4.37	4.72	0.00	0.15	0.24	13.02	16.07	18.48
60	6.90	8.23	8.85	6.44	7.77	8.38	0.00	0.26	0.44	22.35	28.13	32.48
90	9.17	11.15	12.03	8.53	10.51	11.39	0.00	0.35	0.60	29.52	37.80	43.96
120	11.49	14.13	15.27	10.70	13.34	14.48	0.00	0.44	0.76	36.74	47.54	55.49
240	18.99	24.13	26.32	17.71	22.85	25.04	0.00	0.74	1.33	59.65	79.84	94.52
360	23.85	31.03	34.19	21.66	28.84	32.00	0.00	0.91	1.74	73.78	101.36	122.30
488	31.11	40.75	45.04	28.51	38.15	42.44	0.00	1.19	2.29	95.94	132.81	160.67

Decomposing the problem into more blocks allows for a higher degree of parallelism, reducing wall-clock time and RAM requirements per block. Conversely, more blocks increase the number of link variables and constraints generated per block, in turn leading to increased communication overhead when exchanging information between different blocks. To investigate this trade-off, we use the temporal decomposition strategy introduced in Section 2.5 to demonstrate scaling across a range of different block numbers.

3.3. Software setup and hardware

All runs of the computational experiments were performed on the JUWELS cluster at Jülich Supercomputing Centre (JSC) with 2,271 standard nodes, each with two Intel Xeon Platinum 8168 and 96 GB of DDR4 memory, and 240 large memory nodes with the same CPUs but 192 GB of memory.[4] All compute nodes are connected via an InfiniBand EDR network.

The runs with the standard IP solver as the reference case were performed using GAMS 39.1 and CPLEX 22.1. For all runs we used an interior-point method with crossover disabled, which generally seems to provide the best performance for large and complex ESOMs [53]. Other additional settings have been applied based on previous experiences with the REMix model.[5] Sharma et al. report the highest reductions in overall solve time for their model until up to 12 cores, and a slightly less steep reduction between 12 and 35 cores [27]. In order to make use of both a high solver performance and limit the number of cores to allow for running multiple instances on the same machine we utilize 8 cores for the runs with the standard IP solver and take into account the theoretical farming factor. This factor describes how many similar jobs could be run simultaneously on a single compute node given the limitation of 48 physical cores and 192 GB of RAM. When running small-scale models 8 cores are taken into account for the calculation of the compute because memory is not limiting, and 6 instances can be run simultaneously. However, for large model instances, when shared memory is limited, the additional cores cannot be allocated for parallel runs, resulting in a farming factor of 1.

Another option to consider for the solvers is the stopping criterion used by the optimization algorithm. For interior point methods this is usually the duality gap between the primal and dual objectives of the LP. For ESOMs solved as linear problems a sufficient gap tends to be in the range of 10^{-8} to 10^{-4} depending on the optimization problem at hand. For many ESOMs the area around the optimal solution tends to be rather flat due to the large number of marginally different decisions. Therefore, lowering the precision is a reasonable strategy to reduce wall-clock time, albeit at the expense of being slightly off the global optimum. In all runs of the computational experiment, the stopping criterion is set to a tolerance of 10^{-5} between the primal and dual objectives.

The parallel solver PIPS-IPM++ was compiled using the Intel Compiler and ParaStationMPI. The optional library PARDISO 7.2 [54] was used as a sparse direct solver, though other options, such as MA57 [55], can also be used. For PIPS-IPM++ the default settings for the convergence criteria have been used, requiring a μ value below 10^{-6} and a relative residual norm below 10^{-4}. Additional PIPS-IPM++-specific options are required to set PARDISO as the direct solver when multiple direct solvers are available during compilation.[6] As discussed in Section 2.4, the two main drivers of the overall performance of PIPS-IPM++ are the number of blocks into which the problem is decomposed into and the utilized hardware per block. To fully utilize all available 48 cores per compute node and account for NUMA cache locality for the dual-socket CPU nodes, we split the number of blocks into multiples of 12 and allocate 4 cores for each block. This results in a total of 72, 96, 144, and 192 blocks, while utilizing 6, 12 and 24 compute nodes respectively.

4. Results and discussion

The results of this paper are structured as follows: Section 4.1 analyzes the scaling behavior of the model instances with respect to different model sizes and levels of complexity using a standard IP solver. Section 4.2 derives the ideal configurations of the parallel solver PIPS-IPM++ in terms of optimal number of blocks for decomposition and utilized hardware nodes. Section 4.3 then brings together the model instances and the optimal configuration for PIPS-IPM++ from the previous sections to compare PIPS-IPM++ to the standard IP solver in terms of key performance indicators.

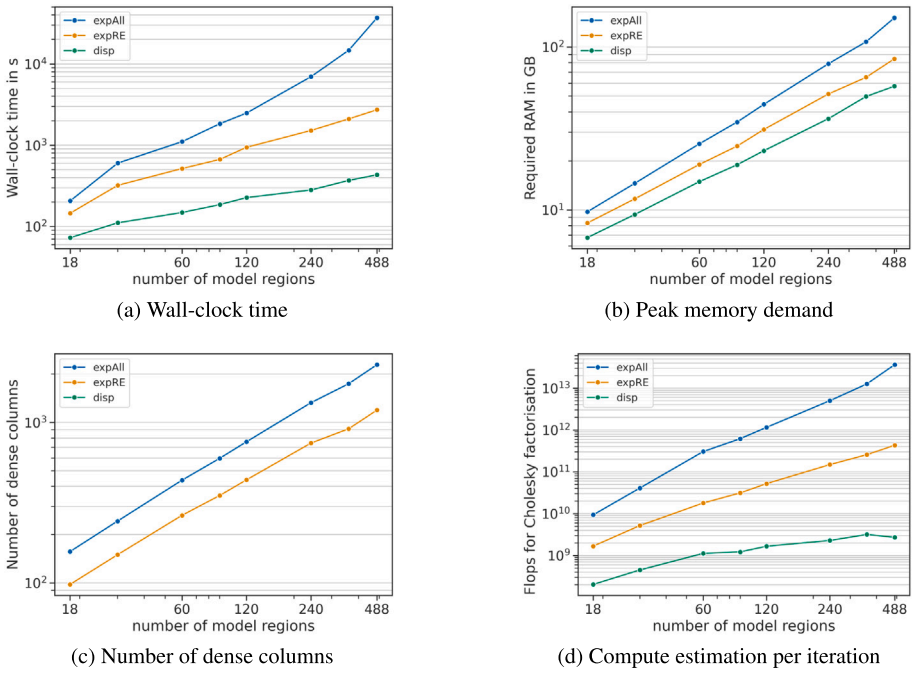
4.1. Complexity of ESOMs

Fig. 4 shows the comparison of the runs performed using a standard IP solver for the 24 instances described in Section 3.1. As expected, Fig. 4(a) shows an overall increase in the wall-clock time as the number of model regions increases. Similarly, the overall complexity of the problem in terms of capacity expansion decisions significantly impacts the results. The difference between the types of capacity expansion becomes more pronounced when a larger number of model regions is considered. The increasing difference in solving times can be explained by the additional complexity due to capacity expansion decisions indicated by the number of dense columns (see Fig. 4(c)). Dense columns in the coefficient matrix result from variables that appear in many equations, such as limitations on dispatch relative to capacity expansion. These variables

[4] https://www.fz-juelich.de/en/ias/jsc/systems/supercomputers/juwels

[5] The following CPLEX solver options have been used for the computational experiment:
lpmethod 4 (select interior point method)
predual −1 (pass only the primal problem to the pre-solver)
barorder 3 (use nested dissection for the matrix ordering)
barepcomp 1e-5 (set stopping criteria to 10^{-5})
solutiontype 2 (disable crossover, as no basic solution is required).

[6] The following PIPS-IPM++ solver options have been used to set PARDISO as the direct sparse solver:
LINEAR_ROOT_SOLVER 3 int
LINEAR_SUB_ROOT_SOLVER 3 int
LINEAR_LEAF_SOLVER 3 int



(a) Wall-clock time (b) Peak memory demand

(c) Number of dense columns (d) Compute estimation per iteration

Fig. 4. Scaling across the different instances while using a standard IP solver. Note that both x and y axis are displayed with a logarithmic scaling. (a) Wall-clock time: For all expansion variations, the wall-clock time increases exponentially with more model regions. The *expAll* variation has the steepest ascent. (b) Peak memory demand: The higher the complexity of the expansion variation, the higher the memory demand and the exponential increase. (c) Number of dense columns: This metric is only applicable for the two variations with expansion variables. (d) Compute estimation via FLOPs for Cholesky factorization: Especially the *expAll* variation shows the highest exponential increase with a higher spatial resolution.

are comparable to linking variables and are identified in standard IP solvers to be treated differently during presolve. For the *disp* instance, the presolver can remove all dense variables, resulting in a reduced complexity. For the other instances, *expAll* and *expRE*, the curves show clear scaling behavior with respect to the number of model regions. However, the curves are shifted relative to each other, as *expAll* allows investments in storage technologies and power lines, in addition to renewable power plants, whereas *expRE* only allows investments into renewable power plants. For the largest analyzed instance this amounts to 2290 dense columns for *expAll*, 1194 for *expRE* and 0 for *disp*.

The number of dense columns also affects the total number of variables, constraints and non-zeros in the optimization problem because each investment decision results in dispatch decisions for every hour of the model. The overall size of the optimization matrix is one of the main drivers of the required memory (see Fig. 4(b)). For the largest instance size the total memory requirement ranges from 57 GB to 151 GB depending on the complexity of investment decisions. This memory demand corresponds to the number of variables after presolve, which ranges from 14.4 million to 37.1 million. We consider the number of variables because it is larger than the number of constraints. This allows us to derive an estimate of memory per variable, ranging from 3.96 to 4.07 GB per million variables after presolve, indicating a theoretical limitation of a few hundred million variables on current shared memory architectures.

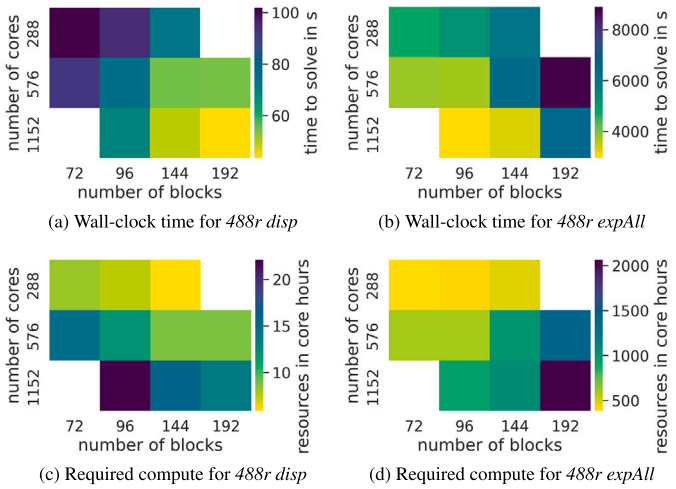
Another helpful metric for estimating the total runtime of the optimization in barrier algorithms is the number of floating point operations (FLOPs) for the Cholesky factorization. This value provides a rough estimate of how much time each iteration will take. However, the total time required to solve the problem depends on the maximum achievable FLOPs of the hardware and the total number of barrier iterations required for the solution. Fig. 4(d) shows the number of FLOPs required per iteration across all instances, indicating a clear difference in computational effort for each barrier iteration depending on the instance's complexity.

In summary, we can clearly establish a connection between the overall size of the optimization problem and the number of capacity expansion decisions as the primary drivers of model complexity. A comparison of the scaling behaviors for memory and wall-clock time indicates that, with a larger number of investment decisions, the increase in wall-clock time significantly outpaces the increase in memory requirement. Therefore, even if the shared memory suffices for the computation, the theoretical example with several hundred million variables is expected to require multiple days of wall-clock time to solve.

4.2. Solver tuning for PIPS-IPM++

To identify the ideal configurations for the model decomposition and the hardware, we run the same model instances using a range of different configurations as outlined in Section 3.3. Fig. 5 shows a clear difference between the two large-scale instances (*disp* on the left, *expAll* on the right). The top figures show the time to solve for different configurations in terms of chosen number of blocks and cores. As expected, a larger number of cores results in faster solution times at the expense of additional computing power. However, the figure also highlights the risk of choosing non-optimal configurations, which can result in up to three times longer solution times compared to the best configuration.

For the economic dispatch instance the fastest time to solution is achieved with both a large number of cores and a large number of blocks. This suggests that the model complexity, due to linking variables and constraints, is not the limiting factor and even higher speedups would be possible if more blocks and cores are utilized. Conversely, the best performance when dealing with a large number of investment decisions is achieved with a high number of cores and a low number of blocks. This indicates an additional limitation of the algorithm's overall parallelism. In this case, the bottleneck for the performance is not in solving the individual blocks, but rather the computation of the global Schur complement and overall communication due to the additional linking structures of the underlying problem.



(a) Wall-clock time for *488r disp*
(b) Wall-clock time for *488r expAll*
(c) Required compute for *488r disp*
(d) Required compute for *488r expAll*

Fig. 5. Time to solve and required compute for the different model instances and hardware configurations using PIPS-IPM++. Lower time to solve and lower compute resources are favorable. (a) The *disp* variation achieves the lowest wall-clock time with more cores and a higher number of blocks. (b) The *expAll* variation benefits from more cores when minimizing wall-clock time, however, fewer blocks are favorable. (c) The compute of the *disp* variation is the lowest with more blocks and a lower number of cores. (d) The *expAll* variation requires the least compute with fewer blocks and cores.

These findings provide a good initial estimate for the configuration of PIPS-IPM++ depending on the optimization problem to be solved. The optimal configuration also depends on the overall goal of deploying PIPS-IPM++: decreasing wall-clock time, increasing efficiency in terms of computation, or preventing limitations from the shared memory architectures. Furthermore, these results provide an explanation for the significant differences in speedup observed in the previous two applications of PIPS-IPM++ by Rehfeldt et al. and Panos and Hassan, as reported in their respective studies [44,45]. Rehfeldt et al. observed the highest speedup, a factor of 67, for the *ELMOD_EU16* instance with 876 blocks and 1,752 CPU cores. While the model instances do have around 300,000 linking constraints, they have only very few global linking variables. Therefore, they are similar to the large *disp* instances presented in this study, which also benefited from a larger number of CPU cores. Panos and Hassan, on the other hand, report a speedup of factor 4 for their model instance *288_22_8*, which contains 162,169 linking constraints and 16,783 linking variables which is higher than the largest *expAll* instance considered here. Due to the different model structure, they used 88 blocks and 88 CPU cores.

4.3. Solver comparison with PIPS-IPM++

Bringing together the results of the complexity analysis using standard IP solvers in Section 4.1 and the assessment of different configurations for PIPS-IPM++ in Section 4.2, we can evaluate the relative performance between the two. While the reference solution using the standard IP solver provides only one data point for time to solve and compute (Fig. 4), the different configurations for PIPS-IPM++ provide a range of data points for solution time and compute (Fig. 5). For the following comparison, we will assume prior knowledge of the optimal configuration for PIPS-IPM++ in terms of the shortest time to solve and the least compute, respectively.

Fig. 6 shows the time required to solve and the necessary compute for all instances considered in this study. It illustrates the effect of using the most suitable PIPS-IPM++ configuration instead of the standard IP solver. A significant reduction in time to solve, roughly one magnitude, can be observed across all instances. The greatest reductions are possible for the economic dispatch instances with small numbers of model regions. For instances with the largest number of model regions, regardless of model complexity, the time required to solve can be reduced by one order of magnitude by using PIPS-IPM++ instead of the standard IP solver. However, this speedup comes at the cost of slightly increased compute and furthermore requires prior knowledge of the optimal PIPS-IPM++ configuration. Only for the instances *expAll* with 488 model regions is it roughly on par. This is mainly due to the farming factor that we considered, which significantly favors the standard IP solver if memory is not a limiting factor.

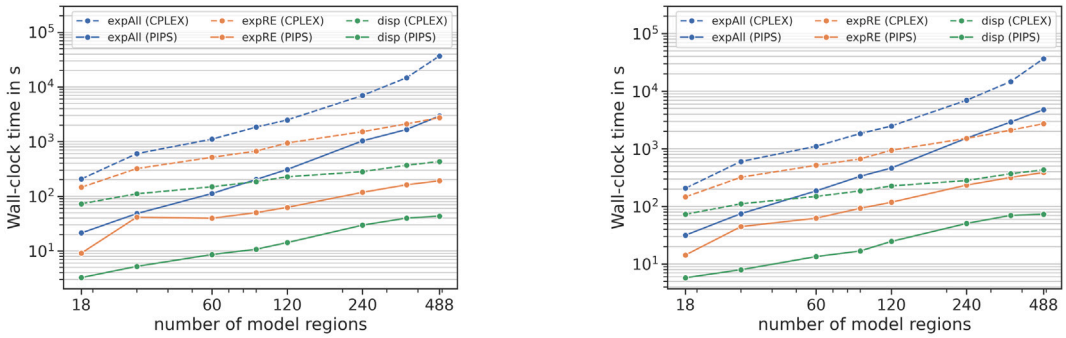
The speedup achieved via the application of PIPS-IPM++ can also be expressed in terms of solving larger models within a similar time frame. For the model instance *disp*, the greatest difference is observed when solving the 488 region instance faster than the 18 region instance using the standard IP solver. However, since the absolute solution time is still in the range of minutes, the real-world savings are negligible. In contrast, the savings in wall-clock time for the instance *expRE* allow one to increase the number of regions from 60 to 240 or from 90 to 360, respectively. On average, model instances can increase fourfold with a similar wall-clock time to solve, significantly pushing the limits of computational complexity.

Switching to a comparison of the most efficient solution in terms of required compute, we observe similar findings to those of the shortest time to solve. Fig. 6(b) shows that all instances can be solved faster using PIPS-IPM++. Additionally, the required compute to solve the same instances is reduced across all model instances. This is especially relevant for the full capacity expansion instances. For the 488 model regions instance, we observe a 13 % reduction in solve time, while saving 22 % of compute compared to the standard IP solver. These results further support the conclusion that applying PIPS-IPM++ can provide the greatest benefits for large, complex models. Table 3 shows the solving time and compute for all scenarios and regions, as well as for both solvers and the different PIPS-IPM++ configurations.

5. Limitations

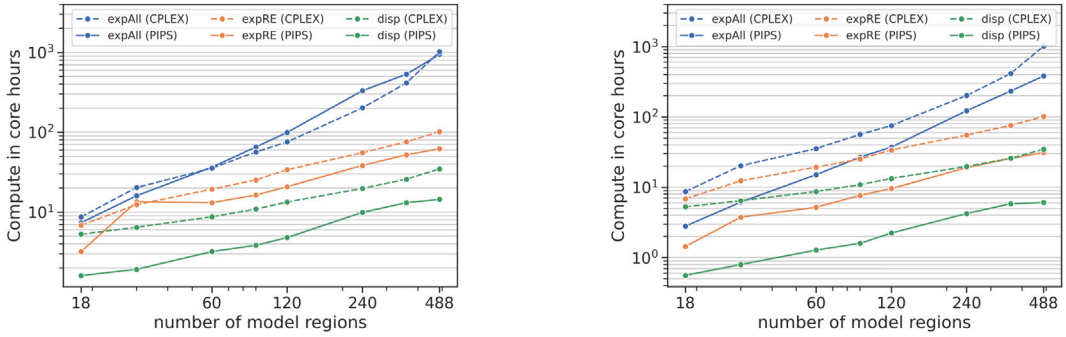
Strictly speaking, the findings of this study are only applicable to the REMix framework. However, the results align with findings from previous applications, suggesting some degree of transferability to other ESOMs. Considering all applications thus far, the performance of PIPS-IPM++ highly depends on a suitable algebraic model formulation, the total number of blocks and the linking structure between blocks. In the case of non-optimal configurations for PIPS-IPM++, the computational benefits can quickly diminish. Similarly, access to hardware with sufficient performance to take advantage of the high degree of parallelism can be a limiting factor.

Based on the reported findings for ELMOD [56], the Swiss TIMES model [57], and REMix [46], we can formulate some general expectations for other ESOMs: For models with a high temporal resolution, chronologically ordered time steps, and limited interlinkages between time steps, such as PyPSA [58], oemof [59], Backbone [60], Balmorel [61], and AnyMod [62], we expect similar performance benefits from PIPS-IPM++, under the assumption of a similar ratio between time steps and investment decisions, and of a well-performing configuration. When it comes to models using representative time slices, such as OSeMOSYS [63], GENeSYS-MOD [64], and models from the TIMES family, the ratio between investment decisions and the number of time steps makes it more challenging to apply PIPS-IPM++. Furthermore, aspects such as international energy trade can make a sufficient decomposition into blocks challenging, since they are usually represented on an annual rather than sub-annual timescale. Applying different decomposition strategies may therefore be more appropriate for these types of models. Regarding integrated assessment models (IAMs), such as Message$_{ix}$ [65] and REMIND [66], we expect similar challenges, because these models focus on additional aspects, such as the overall economy and environmental impact. IAMs also typically consider more



(a) Wall-clock time for least time to solve

(b) Wall-clock time for least compute



(c) Required compute for least time to solve

(d) Required compute for least compute

Fig. 6. Time to solve and required compute for the three model instances solved with the standard IP solver and PIPS-IPM++. Sub-figures (a) and (c) utilize the PIPS-IPM++ configuration optimized for the least time to solve. Sub-figures (b) and (d) utilize the PIPS-IPM++ configuration optimized for the least compute. Lower time to solve and lower compute resources are better. Note that both x and y axis are displayed with a logarithmic scaling.

Table 3
Wall-clock time and compute for all scenarios and regions. Comparison between CPLEX and different configurations for PIPS-IPM++. Depending on the configuration either the solving time or the compute is minimized for PIPS-IPM++.

scenario	regions	CPLEX		PIPS-IPM++ (shortest time)		PIPS-IPM++ (least compute)	
		time [s]	compute [core h]	time [s]	compute [core h]	time [s]	compute [core h]
disp	18	73	5.28	3.25	1.60	5.25	0.56
	30	111	6.45	5.23	1.92	7.95	0.80
	60	149	8.72	8.62	3.20	13.48	1.28
	90	186	10.91	10.73	3.84	16.81	1.60
	120	227	13.36	14.22	4.80	24.68	2.24
	240	282	19.79	29.61	9.92	50.62	4.24
	360	369	25.81	39.90	13.12	69.93	5.84
	488	433	34.80	43.46	14.40	73.33	6.08
expRE	18	146	6.85	9.12	3.20	14.34	1.44
	30	321	12.37	41.47	13.44	44.55	3.76
	60	517	19.33	39.69	13.12	62.45	5.20
	90	670	25.28	50.25	16.32	92.92	7.68
	120	943	33.95	62.65	20.80	117.99	9.60
	240	1519	55.33	118.06	38.40	234.74	19.04
	360	2109	75.87	162.19	52.16	318.73	25.76
	488	2733	101.55	192.44	62.08	387.57	31.20
expAll	18	207	8.69	21.33	7.36	31.65	2.80
	30	603	20.29	48.27	16.00	74.69	6.16
	60	1108	35.44	112.32	36.48	185.92	15.12
	90	1839	56.59	202.60	65.28	333.10	26.80
	120	2489	75.52	309.26	99.52	464.43	37.28
	240	6973	201.81	1030.94	330.56	1530.5	122.56
	360	14723	414.11	1665.62	533.44	2915.69	233.36
	488	36897	1015.47	2944.78	942.72	4791.47	383.44

nonlinear and nonconvex interactions, which may prevent the application of PIPS-IPM++ altogether. Ultimately, future research is needed to demonstrate the generalized transferability of our findings and show concrete performance benefits across a wide range of ESOM frameworks and model instances with different levels of complexity.

The performance of PIPS-IPM++ can be significantly affected not only by the software, but also by the available hardware, such as the CPU and memory per compute node, as well as by proper software configuration, such as thread pinning. Therefore, the findings of this study should be considered in the context of the used HPC until the solver has been applied to a wider range of clusters. Additionally, computational experiments may be influenced by other factors, such as simultaneous computational loads on the same nodes or general loads on the communication network. Although we applied simple countermeasures, such as allocating complete compute nodes for each run, variations caused by simultaneous load can affect the results. Ideally, computational experiments are performed with multiple runs per test and on different hardware to ensure the reliability of the results. For this study, however, such a design was not possible due to limited access to HPC clusters and the prohibitively large number of computational resources required to obtain statistical significance for the large model instances.

It should also be noted that, in its current state, PIPS-IPM++ has strict requirements regarding the provided optimization problems. These requirements must be addressed by the modeler until adequate heuristics are incorporated into the solver itself. For example, this concerns rank-deficient matrices, which standard IP solvers can automatically presolve but which PIPS-IPM++ currently cannot handle due to challenges in preserving the block structure during the presolve. This can render PIPS-IPM++ impossible to apply to some of the aforementioned problems.

6. Conclusion and outlook

This study evaluates a broad spectrum of scalable energy system model instances generated with the REMix framework. It shows the direct link between the size and complexity of the model instances and their corresponding computational requirements. Specifically, the number of investment decisions significantly impacts the required wall-clock time, while the total number of variables and constraints primarily drives the memory demand. These findings also suggest that, as the size increases, pure economic dispatch problems are more likely to be limited by available shared memory, while capacity expansion problems are more likely to be limited by the time required to obtain a solution.

We further demonstrate that decomposing the problem and utilizing the massively parallel solver PIPS-IPM++ can significantly reduce solution times and prevent limitations imposed by shared memory architectures, given both access to sufficient HPC hardware and a well performing configuration. However, the optimal configuration of PIPS-IPM++ depends heavily on the specific model instance to be solved and the emphasis placed on either the least wall-clock time or the least compute time. The number of blocks into which the optimization problem is decomposed and the number of cores used for the parallel solving process must be adjusted accordingly. For instances with few investment decisions, a large number of blocks is advisable. For instances with a large number of investment decisions, a smaller number of blocks should be used. If the primary goal is to reduce the wall-clock time to solve an instance, utilizing more cores is beneficial. Conversely, reducing the required compute can be achieved by reducing the number of utilized cores.

The observed speedups are in line with previous applications of PIPS-IPM++ which demonstrated significant speedups for pure economic dispatch models [44] and a limited speedups for models with a greater emphasis on capacity expansion planning [45]. In our application, the greatest benefits are obtained for large, complex instances, where solving with standard IP solvers on a shared memory architecture is challenging or impossible. First, the speedup achieved by PIPS-IPM++ allows more scenarios to be solved in the same amount of time. Thus, the challenge of multiple uncertainties associated with scenario studies of large energy systems can be addressed more effectively [6]. Second, the ability to switch to distributed memory machines avoids the memory limitations previously encountered on shared memory machines, as only part of the full optimization matrix needs to be stored in local memory [44]. This is particularly relevant for using modern HPC infrastructure, as the emphasis is generally on high parallelization and throughput, and individual compute nodes have limited memory.

When it comes to the question of whether to adopt the use of PIPS-IPM++, several key conditions must be met. First, access to a multi-node computing cluster is recommended or, alternatively, cutting-edge CPUs with more than 256 cores on shared memory systems. This translates to approximately 120 blocks with two cores per block or 60 blocks with four cores per block, which seems like a reasonable lower entry point. Furthermore, due to the additional overhead of annotating and splitting the problem, using PIPS-IPM++ is only recommended for models that exceed several hours of solve time with standard IP solvers or that exceed the available shared memory. Additionally, extra time must be factored in to determine well-performing configurations in terms of the number of blocks and cores used. Regarding the trade-off between configurations optimized for the shortest time or the least compute, the decision depends heavily on the specific usage scenario. The former is especially relevant for academic HPC systems, which often have constraints on the maximum time allowed for each computing job. The latter is potentially more suitable for users running PIPS-IPM++ in cloud computing environments, where computing costs are charged based on usage. In both cases, we expect users to benefit most from significant reductions in the time it takes to solve large linear optimization problems.

Against the background of ever-increasing model complexity and the prevalent design of modern HPC infrastructure, massively parallel solvers for general optimization problems, such as PIPS-IPM++, offer clear advantages compared to other current approaches. While many of these solvers are still in the early stages of research, they have the potential to significantly benefit many different ESOM frameworks in the long term. However, additional development efforts are required for PIPS-IPM++ specifically in terms of numerical stability, testing across multiple platforms, and lowering the entry barrier for using the solver. Regarding the development of ESOM frameworks, different annotations and classes of problem instances must be explored in the future. Some of the most promising approaches that could significantly benefit from the application of solvers like PIPS-IPM++ are capacity expansion planning problems with stochastic economic dispatch, as well as the computation of long-term transformation pathways with perfect foresight. These approaches share a decomposition into capacity expansion and economic dispatch along the time dimension, offering a large number of blocks while keeping the total number of linking variables and constraints relatively low. Going forward, this allows for the efficient solution of increasingly large model instances, such as the assessment of resource adequacy for renewable capacities [37,67] and the outlining of long-term transitions toward systems with high shares of renewable energy technologies and high degrees of sector integration at high temporal resolution [17].

CRediT authorship contribution statement

Manuel Wetzel: Writing – original draft, Visualization, Software, Methodology, Formal analysis, Conceptualization. **Karl-Kiên Cao:** Writing – review & editing, Project administration, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Shima Sasanpour:** Writing – review & editing, Visualization, Investigation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The research for this paper was performed within the projects 'BEAM-ME' and 'UNSEEN' supported by the German Federal Ministry for Economic Affairs and Energy under grant numbers 03ET4023A and 03EI1004A.

The authors gratefully acknowledge the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) for funding this project by providing computing time through the John von Neumann Institute for Computing (NIC) on the GCS Supercomputer JUWELS at Jülich Supercomputing Centre (JSC).

Furthermore, the authors thank the developers of PIPS-IPM++ D. Rehfeldt and N. Kempke for providing insights into the solver and debugging encountered errors, F. Fiand for establishing the GAMS/PIPS++ Solverlink and supporting the annotation and optimization of the REMix model, and T. Breuer for his support in utilizing the JUWELS HPC. In addition, we would like to thank all current and former members of the Energy Systems Modeling group at DLR for their continuous contributions to the modeling framework REMix and the input data for the model instances, in particular G. Recht and H. C. Gils, who reviewed the manuscript, J. Buschmann, who contributed to the input data collection for the model instance, and F. Borggrefe, who contributed to the funding acquisition.

Data availability

Model instances from the computational experiment, as well as HPC-specific scripts and configurations, can be provided by the corresponding author upon request.

References

[1] K. Phillips, J.A. Moncada, H. Ergun, E. Delarue, Spatial representation of renewable technologies in generation expansion planning models, Applied Energy 342 (121092) (2023) https://doi.org/10.1016/j.apenergy.2023.121092.

[2] M.M. Frysztacki, J. Hörsch, V. Hagenmeyer, T. Brown, The strong effect of network resolution on electricity system models with high shares of wind and solar, Appl. Energy 291 (116726) (2021) https://doi.org/10.1016/j.apenergy.2021.116726.

[3] J. Haas, F. Cebulla, K. Cao, W. Nowak, R. Palma-Behnke, C. Rahmann, P. Mancarella, Challenges and trends of energy storage expansion planning for flexibility provision in low-carbon power systems – a review, Renew. Sustain. Energy Rev. 80 (2017) 603–619, https://doi.org/10.1016/j.rser.2017.05.201.

[4] K.-K. Cao, K. von Krbek, M. Wetzel, F. Cebulla, S. Schreck, Classification and evaluation of concepts for improving the performance of applied energy system optimization models, Energies 12 (24) (2019) 4656.

[5] L. Kotzur, L. Nolting, M. Hoffmann, T. Groß, A. Smolenko, J. Priesmann, H. Büsing, R. Beer, F. Kullmann, B. Singh, A. Praktiknjo, D. Stolten, M. Robinius, A modeler's guide to handle complexity in energy systems optimization, Adv. Appl. Energy 4 (100063) (2021) https://doi.org/10.1016/j.adapen.2021.100063.

[6] K.-K. Cao, U. Frey, T. Breuer, M. Wetzel, S. Sasanpour, J. Buschmann, K. von Krbek, A. Böhme, A multi-perspective approach for exploring the scenario space of future power systems, in: International Conference on Operations Research - OR 2022, Springer, Aug 2022. URL https://elib.dlr.de/188202/.

[7] B.A. Frew, M.Z. Jacobson, Temporal and spatial tradeoffs in power system modeling with assumptions about storage: an application of the POWER model, Energy 117 (2016) 198–213, https://doi.org/10.1016/j.energy.2016.10.074.

[8] M. Hoffmann, B.U. Schyska, J. Bartels, T. Pelser, J. Behrens, M. Wetzel, H.C. Gils, C.-F. Tang, M. Tillmanns, J. Stock, A. Xhonneux, L. Kotzur, A. Praktiknjo, T. Vogt, P. Jochem, J. Linßen, J.M. Weinand, D. Stolten, A review of mixed-integer linear formulations for framework-based energy system models, Adv. Appl. Energy 16 (100190) (2024) https://doi.org/10.1016/j.adapen.2024.100190.

[9] G. Savvidis, K. Siala, C. Weissbart, L. Schmidt, F. Borggrefe, S. Kumar, K. Pittel, R. Madlener, K. Hufendiek, The gap between energy policy challenges and model capabilities, Energy Policy 125 (2019) 503–520, https://doi.org/10.1016/j.enpol.2018.10.033.

[10] J. DeCarolis, H. Daly, P. Dodds, I. Keppo, L. Francis, W. McDowall, S. Pye, N. Strachan, E. Trutnevyte, W. Usher, M. Winning, S. Yeh, M. Zeyringer, Formalizing best practice for energy system optimization modelling, Appl. Energy 194 (2017) 184–198, https://doi.org/10.1016/j.apenergy.2017.03.001.

[11] J. Wales, A. Zolan, T. Flamand, A. Newman, Decomposing a renewable energy design and dispatch model, Optim. Eng. 26 (1) (2025) 613–653.

[12] A. Zerrahn, W.-P. Schill, Long-run power storage requirements for high shares of renewables: review and a new model, Renew. Sustain. Energy Rev. 79 (2017) 1518–1534.

[13] C.F. Heuberger, E.S. Rubin, I. Staffell, N. Shah, N.M. Dowell, Power capacity expansion planning considering endogenous technology cost learning, Appl. Energy 204 (2017) 831–845, https://doi.org/10.1016/j.apenergy.2017.07.075.

[14] P.V. Gomes, J.T. Saraiva, State-of-the-art of transmission expansion planning: a survey from restructuring to renewable and distributed electricity markets, Int. J. Electr. Power Energy Syst. 111 (2019) 411–424.

[15] F. Neumann, V. Hagenmeyer, T. Brown, Assessments of linear power flow and transmission loss approximations in coordinated capacity expansion problems, Appl. Energy 314 (118859) (2022) https://doi.org/10.1016/j.apenergy.2022.118859.

[16] T. Brown, D. Schlachtberger, A. Kies, S. Schramm, M. Greiner, Synergies of sector coupling and transmission reinforcement in a cost-optimised, highly renewable European energy system, Energy 160 (2018a) 720–739.

[17] J. Hanto, P. Herpich, K. Löffler, K. Hainsch, N. Moskalenko, S. Schmidt, Assessing the implications of hydrogen blending on the European energy system towards 2050, Adv. Appl. Energy 13 (100161) (2024) https://doi.org/10.1016/j.adapen.2023.100161.

[18] R. Martínez-Gordón, M. Sánchez-Diéguez, A. Fattahi, G. Morales-España, J. Sijm, A. Faaij, Modelling a highly decarbonised North Sea energy system in 2050: a multi-national approach, Adv. Appl. Energy 5 (100080) (2022) https://doi.org/10.1016/j.adapen.2021.100080.

[19] J. Priesmann, L. Nolting, A. Praktiknjo, Are complex energy system models more accurate? An intra-model comparison of power system optimization models, Appl. Energy 255 (113783) (2019).

[20] C. Gerbaulet, C. von Hirschhausen, C. Kemfert, C. Lorenz, P.-Y. Oei, European electricity sector decarbonization under different levels of foresight, Renew. Energy 141 (2019) 973–987.

[21] L. Kotzur, P. Markewitz, M. Robinius, D. Stolten, Impact of different time series aggregation methods on optimal energy system design, Renew. Energy 117 (2018) 474–487.

[22] S. Buchholz, M. Gamst, D. Pisinger, A comparative study of time aggregation techniques in relation to power capacity expansion modeling, Top 27 (2019) 353–405.

[23] O. Raventós, J. Bartels, Evaluation of temporal complexity reduction techniques applied to storage expansion planning in power system models, Energies 13 (4) (2020) 988.

[24] A. Akhavein, M.F. Firuzabad, R. Billinton, D. Farokhzad, Review of reduction techniques in the determination of composite system adequacy equivalents, Electr. Power Syst. Res. 80 (12) (2010) 1385–1393, https://doi.org/10.1016/j.epsr.2010.06.002.

[25] M.M. Frysztacki, G. Recht, T. Brown, A comparison of clustering methods for the spatial reduction of renewable electricity optimisation models of Europe, Energy Inform. 5 (1) (2022) 1–28.

[26] M. Macmillan, K. Eurek, W. Cole, M.D. Bazilian, Solving a large energy system optimization model using an open-source solver, Energy Strat. Rev. 38 (100755) (2021).

[27] T. Sharma, J. Glynn, E. Panos, P. Deane, M. Gargiulo, F. Rogan, B.Ó. Gallachóir, High performance computing for energy system optimization models: enhancing the energy policy tool kit, Energy Policy 128 (2019) 66–74, https://doi.org/10.1016/j.enpol.2018.12.055.

[28] G. Giannakidis, B.G. Maryse Labriet, G. Tosato, Informing energy and climate policies using energy systems models, Springer International Publishing, Switzerland. Doi 10 (1007) (2015) 978–983.

[29] Y. Chen, Z. Huang, S. Jin, L. Ang, Computing for power system operation and planning: then, now, and the future, iEnergy 1 (3) (2022) https://doi.org/10.23919/IEN.2022.0037.

[30] A.J. Conejo, E. Castillo, R. Minguez, R. Garcia-Bertrand, Decomposition Techniques in Mathematical Programming: Engineering and Science Applications., Springer Science & Business Media, 2006.

[31] A. Flores-Quiroz, K. Strunz, A distributed computing framework for multi-stage stochastic planning of renewable power systems with energy storage as flexibility option, Appl. Energy 291 (116736) (2021) https://doi.org/10.1016/j.apenergy.2021.116736.

[32] L. Gong, C. Wang, C. Zhang, F. Yong, High-performance computing based fully parallel security-constrained unit commitment with dispatchable transmission network, IEEE Trans. Power Syst. 34 (2) (2019) https://doi.org/10.1109/TPWRS.2018.2876025.

[33] L. Gong, Y. Peng, C. Zhang, F. Yong, Fully parallel optimization of coordinated electricity and natural gas systems on high-performance computing, IEEE Trans. Smart Grid 14 (5) (2023) https://doi.org/10.1109/TSG.2023.3235247.

[34] L. Göke, F. Schmidt, M. Kendziorski, Stabilized Benders decomposition for energy planning under climate uncertainty, Eur. J. Oper. Res. 316 (1) (2024) 183–199.

[35] F. Pecci, J.D. Jenkins, Regularized Benders decomposition for high performance capacity expansion models, in: IEEE Transactions on Power Systems, 2025.

[36] A. Papavasiliou, S.S. Oren, B. Rountree, Applying high performance computing to transmission-constrained stochastic unit commitment for renewable energy integration, IEEE Trans. Power Syst. 30 (3) (2014) 1109–1120.

[37] D. Ávila, A. Papavasiliou, M. Junca, L. Exizidis, Applying high-performance computing to the European resource adequacy Assessment, IEEE Trans. Power Syst. 39 (2) (2024) 3785–3797, https://doi.org/10.1109/TPWRS.2023.3304717.

[38] M. Lubin, C.G. Petra, M. Anitescu, V. Zavala, Scalable stochastic optimization of complex energy systems, in: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, 2011, pp. 1–64.

[39] C.G. Petra, O. Schenk, M. Anitescu, Real-time stochastic optimization of complex energy systems on high-performance computers, Comput. Sci. Eng. 16 (5) (2014).

[40] J. Wang, N.-Y. Chiang, C.G. Petra, An asynchronous distributed-memory optimization solver for two-stage stochastic programming problems, in: 2021 20th International Symposium on Parallel and Distributed Computing (ISPDC), 2021, pp. 33–40, https://doi.org/10.1109/ISPDC52870.2021.9521613.

[41] S. Shin, M. Anitescu, F. Pacaud, Accelerating optimal power flow with GPUS: SIMD abstraction of nonlinear programs and condensed-space interior-point methods, Electr. Power Syst. Res. 236 (110651) (2024). ISSN 03787796, https://doi.org/10.1016/j.epsr.2024.110651.

[42] K. Świrydowicz, N. Koukpaizan, T. Ribizel, F. Göbel, S. Abhyankar, H. Anzt, S. Peleš, GPU-resident sparse direct linear solvers for alternating current optimal power flow analysis, Int. J. Electr. Power Energy Syst. 155 (109517) (2024). ISSN 01420615, https://doi.org/10.1016/j.ijepes.2023.109517.

[43] J. Dong, J. Cao, Y. Lu, Y. Zhang, L. Jiulong, X. Chongshan, D. Zheng, S. Han, Transmission expansion planning: a deep learning approach, Sustain. Energy Grids Netw. 41 (101585) (2025) https://doi.org/10.1016/j.segan.2024.101585.

[44] D. Rehfeldt, H. Hobbie, D. Schönheit, T. Koch, D. Möst, A. Gleixner, A massively parallel interior-point solver for LPs with generalized arrowhead structure, and applications to energy system models, Eur. J. Oper. Res. 296 (1) (2022).

[45] E. Panos, A. Hassan, Accelerating the performance of large-scale times models in the modelling of sustainable development goals, In M. Labriet; K. Espegren; G. Giannakidis, B.Ó. Gallachóir (Eds.), Aligning the Energy Transition with the Sustainable Development Goals, vol. 101, Springer Nature Switzerland and Imprint Springer, Cham, 2024. ISBN pp. 67–95, 9783031588976 , https://doi.org/10.1007/978-3-031-58897-6_4 of *Lecture Notes in Energy*.

[46] M. Wetzel, E.S.A. Ruiz, F. Witte, J. Schmugge, S. Sasanpour, M. Yeligeti, F. Miorelli, J. Buschmann, K.-K. Cao, N. Wulff, H. Gardian, A. Rubbert, B. Fuchs, Y. Scholz, H.C. Gils, REMix: a GAMS-based framework for optimizing energy system models, J. Open Source Softw. 9 (99) (2024) https://doi.org/10.21105/joss.06330.

[47] H.C. Gils, H. Gardian, J. Schmugge, Interaction of Hydrogen infrastructures with other sector coupling options towards a zero-emission energy system in Germany, Renew. Energy 180 (2021) 140–156, https://doi.org/10.1016/j.renene.2021.08.016.

[48] M. Wetzel, H.C. Gils, V. Bertsch, Green energy carriers and energy sovereignty in a climate neutral European energy system, Renew. Energy 210 (2023) 591–603.

[49] C.K. Nils, PIPS-IPM++ – a massively parallel interior-point method, in: Operations Research 2021, International Conference of the Swiss, German and Austrian Operations Research Societies (SVOR/ASRO, GOR EV, ÖGOR), University of Bern, Switzerland, August 31–September 3, 2021.

[50] K.-K. Cao, M. Wetzel, N.-C. Kempke, T. Koch, Pushing computational boundaries: solving integrated investment planning problems for large-scale energy systems with PIPS-IPM++, in: Operations Research 2021, 2021. URL https://elib.dlr.de/143744/.

[51] T. Khaniyev, S. Elhedhli, F.S. Erenay, Structure detection in mixed-integer programs, Inf. J. Comput. 30 (3) (2018) https://doi.org/10.1287/ijoc.2017.0797.

[52] K.-K. Cao, J. Metzdorf, S. Birbalta, Incorporating power transmission bottlenecks into aggregated energy system models, Sustainability 10 (6) (2018) 1916,.

[53] M. Bröchin, B. Pickering, T. Tröndle, S. Pfenninger, Harder, better, faster, stronger: understanding and improving the tractability of large energy system models, Energy. Sustain. Soc. 14 (1) (2024) https://doi.org/10.1186/s13705-024-00458-z.

[54] O. Schenk, K. Gärtner, Solving unsymmetric sparse systems of linear equations with PARDISO, Futur. Gener. Comput. Syst. 20 (3) (2004). ISSN 0167739X, https://doi.org/10.1016/j.future.2003.07.011.

[55] S.D. Iain, MA57 – a code for the solution of sparse symmetric definite and indefinite systems, ACM Trans. Math. Softw. 30 (2) (2004) 118–144.

[56] F.U. Leuthold, H. Weigt, C. von Hirschhausen, A large-scale spatial optimization model of the European electricity market, Netw. Spat. Econ. 12 (1) (2012). ISSN 1566-113X, https://doi.org/10.1007/s11067-010-9148-1.

[57] R. Kannan, H. Turton, A long-term electricity dispatch model with the TIMES framework, Environ. Model. Assess. 18 (3) (2013). ISSN 1420-2026, https://doi.org/10.1007/s10666-012-9346-y.

[58] T. Brown, J. Hörsch, D. Schlachtberger, PyPSA: Python for power system analysis, J. Open Res. Softw. 6 (1) (2018b) https://doi.org/10.5334/jors.188.

[59] S. Hilpert, C. Kaldemeyer, U. Krien, S. Günther, C. Wingenbach, G. Plessmann, The Open Energy Modelling Framework (oemof) - a new approach to facilitate open science in energy system modelling, Energy Strat. Rev. 22 (2018) 16–25. ISSN 2211467X, https://doi.org/10.1016/j.esr.2018.07.001.

[60] N. Helistö, J. Kiviluoma, J. Ikäheimo, T. Rasku, E. Rinne, C. O'Dwyer, L. Ran, D. Flynn, Backbone—an adaptable energy systems modelling framework, Energies 12 (17) (2019) https://doi.org/10.3390/en12173388.

[61] F. Wiese, R. Bramstoft, H. Koduvere, A.P. Alonso, O. Balyk, J.G. Kirkerud, Å.G. Tveten, T.F. Bolkesjø, M. Münster, H. Ravn, Balmorel open source energy system model, Energy Strat. Rev. 20 (2018) 26–34. ISSN 2211467X, https://doi.org/10.1016/j.esr.2018.01.003.

[62] L. Göke, Anymod.jl: a Julia package for creating energy system models, SoftwareX 16 (100871) (2021). ISSN 23527110, https://doi.org/10.1016/j.softx.2021.100871.

[63] M. Howells, H. Rogner, N. Strachan, C. Heaps, H. Huntington, S. Kypreos, A. Hughes, S. Silveira, J. DeCarolis, M. Bazillian, A. Roehrl, OSeMOSYS: the open source energy modeling system, Energy Policy 39 (10) (2011). ISSN 03014215, https://doi.org/10.1016/j.enpol.2011.06.033.

[64] K. Löffler, K. Hainsch, T. Burandt, P.-Y. Oei, C. Kemfert, C. von Hirschhausen, Designing a model for the global energy system – GENeSYS-MOD: an application of the open-source energy modeling system (OSeMOSYS), Energies 10 (10) (2017) 1468, https://doi.org/10.3390/en10101468.

[65] D. Huppmann, M. Gidden, O. Fricko, P. Kolp, C. Orthofer, M. Pimmer, N. Kushin, A. Vinca, A. Mastrucci, K. Riahi, V. Krey, The MESSAGE Integrated assessment model and the IX modeling platform (IXMP): an open framework for Integrated and cross-cutting analysis of energy, climate, the environment, and sustainable development, Environ. Model. Softw. 112 (2019) 143–156. ISSN 13648152, https://doi.org/10.1016/j.envsoft.2018.11.012.

[66] L. Baumstark, N. Bauer, F. Benke, C. Bertram, B. Stephen, C.C. Gong, J.P. Dietrich, A. Dirnaichner, A. Giannousakis, J. Hilaire, D. Klein, J. Koch, M. Leimbach, A. Levesque, S. Madeddu, A. Malik, A. Merfort, L. Merfort, A. Odenweller, M. Pehl, R.C. Pietzcker, F. Piontek, S. Rauner, R. Rodrigues, M. Rottoli, F. Schreyer, A. Schultes, B. Soergel, Dominika Soergel, Jessica Strefler, Falko Ueckerdt, Elmar Kriegler, and Gunnar Luderer. REMIND2.1: transformation and innovation dynamics of the energy-economic system within climate and sustainability limits, Geosci. Model Dev. 14 (10) (2021) https://doi.org/10.5194/gmd-14-6571-2021.

[67] T.H. Ruggles, E. Virgüez, N. Reich, J. Dowling, H. Bloomfield, E.G.A. Antonini, S.J. Davis, N.S. Lewis, K. Caldeira, Planning reliable wind- and solar-based electricity systems, Adv. Appl. Energy 15 (100185) (2024) https://doi.org/10.1016/j.adapen.2024.100185.