European Journal of Operational Research 316 (2024) 183–199

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Stochastics and Statistics

Stabilized Benders decomposition for energy planning under climate uncertainty

Leonard Göke [a,b,c,*], Felix Schmidt [c], Mario Kendziorski [b,c]

[a] Energy and Process Systems Engineering, Department of Mechanical and Process Engineering, ETH Zurich, Tannenstrasse 3, Zurich 8092, Switzerland
[b] Workgroup for Infrastructure Policy (WIP), Technische Universität Berlin, Straße des 17. Juni 135, 10623 Berlin, Germany
[c] Energy, Transportation, Environment Department, German Institute for Economic Research (DIW Berlin), Mohrenstraße 58, 10117 Berlin, Germany

ARTICLE INFO

Keywords:
OR in energy
Large scale optimization
Benders decomposition
Stabilization
Climate uncertainty

ABSTRACT

This paper applies Benders decomposition to two-stage stochastic problems for energy planning under climate uncertainty, a key problem for the design of renewable energy systems. To improve performance, we adapt various refinements for Benders decomposition to the problem's characteristics—a simple continuous master-problem, and few but large sub-problems. The primary focus is stabilization, specifically comparing established bundle methods to a quadratic trust-region approach for continuous problems.

An extensive computational comparison shows that all stabilization methods can significantly reduce computation time. However, the quadratic trust-region and the linear box-step method are the most robust and straightforward to implement. When parallelized, the introduced algorithm outperforms the vanilla version of Benders decomposition by a factor of 100. In contrast to off-the-shelf solvers, computation time remains constant when the number of scenarios increases.

In conclusion, the algorithm enables robust planning of renewable energy systems with a large number of climatic years. Beyond climate uncertainty, it can make an extensive range of other analyses in energy planning computationally tractable, for instance, endogenous learning and modeling to generate alternatives.

1. Introduction

The global energy system must undergo a major transformation and replace 86% of primary energy from fossil fuels as of 2019 with renewable resources to achieve the objectives of the Paris Climate Agreement and limit global warming, (Ritchie & Roser, 2020). At the heart of this transformation is renewable electricity from wind and photovoltaic (PV) for two reasons: First, its technical potential exceeds other renewables and even demand projections (Creutzig et al., 2017). Estimates for the global potential of PV alone range from 1585 to 50,580 EJ—at least triple the primary consumption in 2019. Second, the use of renewable electricity is not limited to the power sector but can also be deployed for electric heating, mobility, and the creation of synthetic fuels to decarbonize the heat, transport, and industry sectors.

1.1. System planning under climate uncertainty

Techno-economic planning models are critical tools for analyzing the transformation towards renewable energy systems. Models decide on the expansion and operation of technologies by representing the flow and conversion of energy in future systems. Commonly, models are linear optimization problems that minimize the costs to satisfy an exogenous demand considering different boundary conditions, for instance, emission limits. Typical applications include developing long-term scenarios for the energy system, assessing technologies in a system context, and analyzing energy policy (Göke, Weibezahn, & von Hirschhausen, 2023).

The fluctuating nature of PV and wind generation is challenging for system planning. In conventional energy systems dominated by dispatchable thermal plants, reliable planning only requires a small number of representative time-steps. However, renewable systems require much higher temporal resolution to capture fluctuations of PV and wind. Additionally, planning must consider energy storage as a key option to balance fluctuations (Levin et al., 2023). Both characteristics significantly increase problem size and complexity, quickly rendering the underlying linear optimization problem computationally intractable (Göke & Kendziorski, 2021). As a result, the literature proposes different techniques to reduce model size while representing

* Corresponding author at: Energy and Process Systems Engineering, Department of Mechanical and Process Engineering, ETH Zurich, Tannenstrasse 3, Zurich 8092, Switzerland.
E-mail address: lgoeke@ethz.ch (L. Göke).

https://doi.org/10.1016/j.ejor.2024.01.016
Received 18 August 2022; Accepted 11 January 2024
Available online 15 January 2024

renewables accurately, for instance, iteratively adjusting the representative time-periods or limiting high resolution to selected parts of the system (Göke, 2021b; Teichgraeber, Küpper, & Brandt, 2021).

Nevertheless, all methods for tractable planning of renewable energy systems typically only consider one representative year of climate conditions, but research shows that renewable supply and energy demand vary substantially across years (Pfenninger, 2017; Pfenninger & Staffell, 2016). Based on a sample of 40 historical years, Bloomfield, Brayshaw, Shaffrey, Coker, and Thornton (2016) investigate effects on the British power system and conclude robust planning should consider at least ten of these years. Ruhnau and Qvist (2022) study the effect of inter-annual variability on storage requirements for a stylized fully renewable German power system. Compared to a representative year, storage requirements double when considering 35 years of climate data. Ohlendorf and Schill (2020) specifically analyze the frequency of low-wind power events in Germany threatening system adequacy, again concluding that planning models should consider more climatic years.

Overall, climate uncertainty and risk-aware planning are essential for the security of supply in renewable energy systems. Existing heuristic approaches to consider climate uncertainty are limited: Repeatedly solving the same planning problem for different climate conditions can indicate the extent of variability but not provide a generally reliable solution (Grochowicz, van Greevenbroek, Benth, & Zeyringer, 2023). Solving a single problem with representative time-steps from a multi-year sample of climate conditions cannot capture the entire range of climate conditions (Hilbers, Brayshaw, & Gandy, 2019). Accordingly, recent publications call for new planning methods that deploy stochastic optimization to consider the uncertainty of climate conditions, especially as climate change further increases uncertainty (Craig et al., 2022; Doss-Gollin, Amonkar, Schmeltzer, & Cohan, 2023; Plaga & Bertsch, 2023).

Considering climate uncertainty in system planning creates a two-stage stochastic problem: The first stage decides on capacity expansion with uncertainty regarding climate conditions. The second stage reveals the climate conditions and decides on the operation of capacities, e.g., the production of a thermal power plant. Stochastic programming uses independent scenarios with distinct probabilities to represent different climate conditions in the second stage.

So far, two-stage energy planning is limited to a few scenarios representing a short representative period of a week at maximum (Backe, Skar, del Granado, Turgut, & Tomasgard, 2022). This approach limits problem size and keeps the model tractable, but it is unsuitable for planning renewable systems. Here, scenarios must be long and cover at least one year of climate conditions to model seasonal storage. In addition, scenarios must cover various climate conditions to capture fluctuations and enable reliable planning. The resulting two-stage problem with multiple one-year scenarios is a simple but large linear problem that off-the-shelf solvers can hardly solve.

1.2. Benders decomposition for two-stage stochastic problems

Benders decomposition (BD), first introduced in Benders (1962), is a decomposition technique to solve optimization problems. Van Slyke and Wets (1969) first applied a variation of BD, termed the L-shaped method, to solve two-stage stochastic problems and potentially improve their computational tractability. In energy planning under climate uncertainty, BD decomposes the problem into a master-problem (MP) addressing technology expansion and several mutually independent sub-problems (SPs) for each scenario. Afterwards, the MP and SPs are solved repeatedly to generate constraints, so-called Benders cuts, that are added to the MP until the algorithm converges (Conejo, Castillo, Minguez, & Garcia-Bertrand, 2006).

In existing applications of BD in energy planning models, each SP corresponds to a short time-period, in some models as short as an hour (Brandenberg & Stursberg, 2021; Jacobson, Pecci, Sepulveda, Xu, & Jenkins, 2023; Lohmann & Rebennack, 2017; Skar, Doorman, & Tomasgard, 2014). As already discussed, such decomposition is not applicable when modeling renewable systems dependent on energy storage. In this case, each SP must cover at least one year to account for seasonal storage. Combined with high temporal detail, this results in a simple MP and several large SPs. Furthermore, MP and SPs can contain discrete and continuous variables. However, in the MP, at least renewable expansion is continuous, considering the size of a single wind turbine or PV panel is small from a macro perspective.

For many problems, the original BD converges slowly and is not competitive to off-the-shelf solvers for the deterministic equivalent. However, extensive research proposes different enhancements to improve the original BD. In a review, Rahmaniani, Crainic, Gendreau, and Rei (2017) group them into the following four most important groups, with three of them being relevant to this paper:

- **Solution procedure**: The majority of enhancements in this area reduce the computation time of the MP, like removing non-binding cuts or not solving to optimality in early iterations. In our case, the more relevant SPs can deploy distributed parallel computing. Furthermore, not solving SPs to optimality can produce inexact but valid cuts (de Oliviera & Solodov, 2020; Zakeri, Philpott, & Ryan, 2000). Alternatively, only selected SPs are solved in each iteration to reduce the computational load (van Ackooij & Frangioni, 2018).
- **Solution generation**: Again, most enhancements regarding solution generation aim to accelerate the MP, for instance, by solving a linear relaxation instead of a mixed-integer problem. Other adjustments to the MP improve the convergence rate, rendering them sensible even if the MP is small compared to the SPs. These include:
  - *Multi-cut reformulations* that use separate cuts for each SP instead of a single aggregated cut, improving convergence but making the MP harder. Multi-cut reformulations generally outperform aggregated cuts for energy planning problems since the MP is easy (Jacobson et al., 2023).
  - *Problem-specific heuristics* that provide an initial feasible solution for BD to obtain an upper bound on the objective value or exclude infeasible solutions at an early stage. To achieve the latter, additional constraints, known as valid inequalities, which are guaranteed to hold in the optimum, are added to the MP, narrowing down its solution space (Cordeau, Pasin, & Solomon, 2006). For instance, Wang, Wang, Liu, and Ruiz (2013) add the second-stage constraints of one scenario to the MP, improving convergence.
  - *Regularization* restricting the MP to solutions close to a reference point, usually the current best solution, to avoid heavy oscillation (Ruszczyński, 2003). These methods are not limited to BD or stochastic programming but have their origins in general convex optimization. For instance, Linderoth and Wright (2003) apply the l_∞-norm to limit the maximum difference across all variables between a new and the current best solution of the MP; a method first introduced in a general optimization context in Marsten, Hogan, and Blankenship (1975). Level and proximal bundle methods originate from nonsmooth convex optimization and add a quadratic term to the objective function for stabilization (Lemaréchal, Strodiot, & Bihain, 1981; Ruszczyński, 1986). A series of publications applies explicitly these methods to mitigate the oscillation of BD (e.g. van Ackooij & de Oliveira, 2015; Zverovich, Fábián, Eldon, & Mitra, 2012).
- **Cut generation**: Magnanti and Wong (1981) note that degenerate SPs, e.g., SPs with multiple optimal dual solutions, can generate a set of different cuts, all affecting convergence differently. To

select strong cuts that improve convergence, they propose to solve a modified version of the dual SP after the original SP. As a result, the method is most efficient if the MP is difficult and the SPs are small.

In conclusion, many established enhancements aim at problems with a difficult MP and small SPs, typically combinatorial problems with discrete complicating variables. Rahmaniani et al. (2017) correspondingly state *"it has often been reported that more than 90% of the total execution [...] time is spent on solving the MP"*. Consequently, their applicability to the two-stage planning problem investigated in this paper with a simple MP but large SPs is limited. In contrast to typical applications of BD, it is not combinatorial complexity but the size of the individual SPs that makes our problem computationally challenging.

1.3. Contribution

In this paper, we apply BD to solve a two-stage stochastic problem with discrete scenarios used for planning renewable macro-energy systems under climate uncertainty. The existing literature neither addresses this problem specifically nor generally covers BD for continuous linear problems (LPs) that are challenging due to the size of the individual subproblems.

The main contribution of the paper is to adapt and apply existing refinements of BD to the given problem. Overall, our contribution is rather practical than theoretical and has a focus on efficiently solving the problem at hand. The examined refinements can be divided into two groups.

1. First, common improvements for BD including inexact cuts, valid inequalities, and parallelization. Many other state-of-the-art refinements are not examined since they do not suit the specific structure of the problem.
2. Second, (quadratic) stabilization, or regularization, methods that originate from nonsmooth convex optimization and generally address the instability of cutting plane methods. A particular emphasis is on comparing established methods with a quadratic trust-region approach, which has not been widely used for BD so far.

The structure for the remainder of the paper is as follows: The next Section 2 generally introduces the investigated planning problem and a standard implementation of the Benders algorithm. Section 3 focuses on different stabilization methods and corresponding parameter strategies. Afterwards, Section 4 discusses additional refinements. Section 5 introduces a specific planning problem for an extensive computational comparison of algorithm setups in Section 6. The final section concludes and suggests further developments.

2. Benders algorithm

The two-stage energy planning problem decides on capacity expansion in the first stage and operation in the second. Different scenarios represent uncertainty in the operational stage, for instance, corresponding to different climatic years. The following subsections first introduce the closed formulation of the problem, followed by the basic BD implementation used to solve it. The notation uses lowercase letters for parameters, uppercase letters for variables, and Greek letters for anything specific to BD.

2.1. Problem formulation

Eqs. (1a) to (1j) provide the closed formulation of the two-stage stochastic energy planning problem. For a set of technologies $i \in I$, the problem decides on expanding capacities $Exp_{y,i}$ over a set of chronological but not necessarily consecutive years $y \in Y$. All variables of the problem are continuous and non-negative.

According to (1d), the capacity $Capa_{y,i}$ available in each year depends on expansion in the current and previous years provided by Y_y^{exp}. In Eqs. (1e) to (1h) these capacities constrain the operation of technologies at each time-step $t \in T$, in each scenario $s \in S$, and within each year. For generation technologies I^{ge}, the capacity constraint limiting the generation $Gen_{y,s,i,t}$ includes a scenario-dependent capacity factor $cf_{s,i,t}$ reflecting the available share of capacity. For storage technologies, I^{st}, two different capacities constrain three different operational variables: Charged energy $St^{in}_{y,s,i,t}$ and discharged energy $St^{out}_{y,s,i,t}$ are both constrained by the power capacity $Capa^{st}_{y,i}$; the current storage level $St^{size}_{y,s,i,t}$ is constrained by the energy capacity $Capa^{size}_{y,i}$. Net supply from generation and storage technologies must match the exogenous demand $dem_{y,s,t}$ in each year, time-step, and scenario, as expressed in the energy balance in Eq. (1i). The balance includes a loss-of-load variable $Lss_{y,s,t}$ to avoid an infeasible problem. The storage balance in Eq. (1j) computes the storage level at time-step t based on the storage level in the previous time-step $t-1$ plus charged and minus discharged energy. The storage constraint is circular for each year, meaning the last time-step in a year is previous to the first.

The objective function of the problem in Eq. (1a) minimizes total system costs comprised of expansion costs U and the sum of operational costs $V_{y,s}$ across all years y and scenarios s weighted according to the scenario probabilities p_s. Expansion costs defined in Eq. (1b) depend on the expansion variable and the specific expansion costs $c^{inv}_{y,i}$ of each technology. Operational costs defined in Eq. (1c) depend on the costs associated with loss-of-load c^{lss} and variable costs of generation technologies $c^{var}_{y,s,i}$, for instance, fuel costs.

$$\min U \quad + \sum_{y\in Y, s\in S} p_s \cdot V_{y,s} \tag{1a}$$

$$\text{s.t. } U \quad = \sum_{y\in Y, i\in I} c^{inv}_{y,i} \cdot Exp_{y,i} \tag{1b}$$

$$V_{y,s} \quad = \sum_{t\in T} Lss_{y,s,t} \cdot c^{lss} + \sum_{i\in I} Gen_{y,s,i,t} \cdot c^{var}_{y,s,i} \qquad \forall y \in Y, s \in S \tag{1c}$$

$$Capa_{y,i} \quad = \sum_{y'\in Y^{exp}_y} Exp_{y',i} \qquad \forall y \in Y, i \in I \tag{1d}$$

$$Gen_{y,s,i,t} \quad \le cf_{s,i,t} \cdot Capa^{ge}_{y,i} \qquad \forall y \in Y, s \in S, i \in I^{ge}, t \in T \tag{1e}$$

$$St^{out}_{y,s,i,t} \quad \le Capa^{st}_{y,i} \qquad \forall y \in Y, s \in S, i \in I^{st}, t \in T \tag{1f}$$

$$St^{in}_{y,s,i,t} \quad \le Capa^{st}_{y,i} \qquad \forall y \in Y, s \in S, i \in I^{st}, t \in T \tag{1g}$$

$$St^{size}_{y,s,i,t} \quad \le Capa^{size}_{y,i} \qquad \forall y \in Y, s \in S, i \in I^{st}, t \in T \tag{1h}$$

$$dem_{y,s,t} \quad = Lss_{y,s,t} + \sum_{i\in I^{ge}} Gen_{y,s,i,t} - \sum_{i\in I^{st}} St^{in}_{y,s,i,t} + St^{out}_{y,s,i,t} \qquad \forall y \in Y, s \in S, t \in T \tag{1i}$$

$$St^{size}_{y,s,i,t} \quad = St^{size}_{y,s,i,t-1} + St^{in}_{y,s,i,t} - St^{out}_{y,s,i,t} \qquad \forall y \in Y, s \in S, i \in I^{st}, t \in T \tag{1j}$$

The formulation in Eqs. (1a) to (1j) introduces the general problem structure and all details pivotal for BD but is still stylized. The model introduced in Section 5 and used for benchmarking includes additional elements. First, the model includes multiple regions of expansion and operation. The model can build transmission capacity between regions in the first stage to exchange energy in the second. Next, distinct energy carriers, like electricity and hydrogen, are converted into one another by respective technologies, like hydrogen turbines or electrolyzers. Finally, there are storage losses and additional constraints, imposing upper limits on capacities or restricting operation, for instance, a yearly emission limit. For a comprehensive formulation of the planning problem, see Göke (2021b).

2.2. Benders decomposition

Fig. 1 is an illustrative depiction of the planning problem based on the block structure of the constraint matrix. The expansion problem

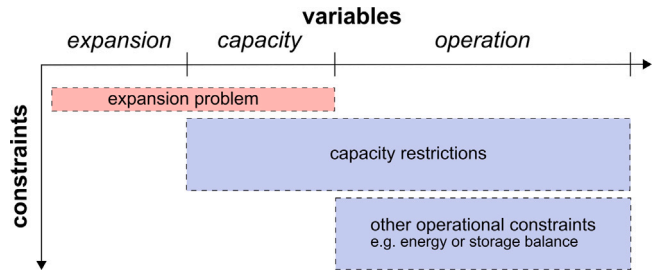

Fig. 1. Structure of closed two-stage problem.

in the first stage includes expansion and capacity variables connected by the constraint in (1d) and the cost definition in Eq. (1b). The second stage includes the remaining constraints, namely the energy and storage balance, the definition of variable costs, and the capacity restrictions. The latter link capacity with operational variables and connects the first stage and second stage of the problem.

Even for a single year and scenario, the number of constraints and variables of the expansion problem in the first stage is small compared to the operation in the second stage. Increasing the number of years extends both stages, but increasing the number of stochastic scenarios extends the operational problem only. Since the expansion problem is small, the total size almost linearly depends on the number of stochastic scenarios, and the problem quickly becomes intractable if their number increases.

For BD, the first stage with expansion corresponds to the MP, while several independent SPs cover operation in the second stage. Consider the temporal problem structure outlined in Fig. 2 for demonstration. The first stage models two steps of capacity expansion with a 10-year interval. Multi-year intervals are a common modeling strategy for representing transformation pathways, aiming to reduce computational complexity. The second stage includes two operational scenarios for each expansion step. These scenarios represent different patterns of renewable supply and energy demand. In this example, these patterns use historical data for the climatic years 2000′ and 2008′. These two years are selected at random here for illustrative purposes. For details on the methods for selecting a sample of climatic years, see Section 9.1 in Appendix. In addition, it is conceivable to have different operational scenarios for each expansion step, for instance, to account for changes in patterns caused by climate change.

According to the problem formulation in the previous section, operation is modeled for each year and scenario, resulting in four distinct operational problems, as shown in Fig. 2. Each operational problem can be solved independently for a given set of capacities if no complicating constraint links the variables from different SPs. An example of such a constraint is an emission limit not enforced for each SP separately but jointly across different years and scenarios, like a carbon budget.

Fig. 3 provides the structure of the decomposed problem, analogously to Fig. 1. It shows how the approach splits the operational problem into four independent SPs, each consisting of different operational variables but sharing capacities with the MP and other SPs for the same year. Splitting the second stage operation into several SPs is beneficial because it enables distributed computing and prevents complexity in the second stage to scale with the number of scenarios. In addition, it increases the number of cuts added to the MP in each iteration, improving convergence, as we will elaborate when introducing the BD algorithm next.

Algorithm 1 presents the standard version of the BD algorithm applied in this paper. k is an iteration counter; ζ_{low} and ζ_{up} are the objective function's current lower and upper bound, respectively. We use a multi-cut version and, at each iteration, add separate constraints for each sub-problem to the MP instead of a single aggregated cut. The approach improves convergence but increases the complexity of the

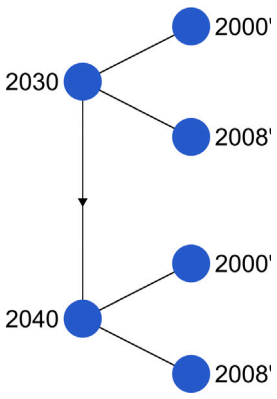

Fig. 2. Graph of years and scenarios for exemplary problem.

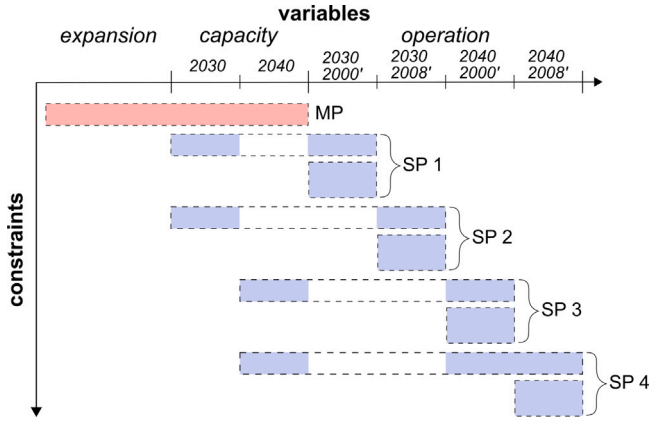

Fig. 3. Structure of decomposed problem.

MP—a favorable trade-off for linear planning problems since the MP is simple (Jacobson et al., 2023).

After initializing all variables, the algorithm starts iterating with a convergence check (Step 1), which tests whether the optimality gap is within a pre-defined tolerance. Step 2 solves the original MP, as described by Eqs. (2a) to (2c). Since the MP only includes the constraints of the expansion problem, its objective function approximates the actual costs of each SP $V_{y,s}$ using the estimator $\alpha_{y,s}$. After solving, the algorithm obtains the resulting capacity vector $Capa^{(k)}$ and expansion costs $U^{(k)}$ for the current iteration and sets the lower bound ζ_{low} to the current objective of the MP. In the first iteration, $\alpha_{y,s}$ is still unconstrained, and the trivial optimum for the MP is zero, meaning no capacity expansion at all.

$$\min \quad U \quad + \sum_{y\in Y, s\in S} p_s \cdot \alpha_{y,s} \tag{2a}$$

$$\text{s.t.} \quad U \quad = \sum_{y\in Y} c^{inv}_{y,i} \cdot Exp_{y,i} \tag{2b}$$

$$Capa_{y,i} = \sum_{y'\in Y^{exp}_{y}} Exp_{y',i} \qquad \forall y \in Y, i \in I \tag{2c}$$

In Step 3.1, the algorithm solves each SP with capacities fixed to the previously computed $Capa^{(k)}$, as described by Eqs. (3a) to (3i). We will discuss the parallelization mentioned in the algorithm in Section 4.3. Afterwards, the algorithm gets the actual costs of each SP $V^{(k)}_{y,s}$ and the dual values $\lambda^{(k)}_{y,s,i}$ of the constraint fixing capacities in Eq. (3i). In our

Algorithm 1: Multi-cut benders algorithm

Step 0 (Initialization)
chose convergence tolerance ϵ and deletion threshold η
set $k = 1$, $\zeta_{up} \leftarrow \infty$ and $\zeta_{low} \leftarrow 0$
Step 1 (Convergence test):
if $1 - \frac{\zeta_{low}}{\zeta_{up}} < \epsilon$ **then** *stop* **end if**
Step 2 (Obtain new iterate)
solve **MP** get $Capa^{(k)}$ and $U^{(k)}$
set $\zeta_{low} \leftarrow obj(\mathbf{MP})$
do in parallel
  for $y \in Y$, $s \in S$ **do**
    Step 3.1 (Solve sub-problems)
    solve $\mathbf{SP}_{y,s}$ with $Capa^{(k)}$
    get $V^{(k)}_{y,s}$ and $\lambda^{(k)}_{y,s}$
    Step 3.2 (Add cuts to MP)
    add $\Omega_{k,y,s}$ to **MP**
    set $\delta_{k,y,s} \leftarrow k$
  end for
end
Step 4 (Bundle management)
for $l \in \{1, \dots, k-1\}, y \in Y, s \in S$ **do**
  if $\Omega_{l,y,s}$ *is binding* **then**
    set $\delta_{l,y,s} \leftarrow k$
  else if $k - \delta_{l,y,s} > \eta$ **then**
    delete $\Omega_{l,y,s}$ from **MP**
end for
Step 5 (Descent test)
if $\zeta_{up} > U^{(k)} + \sum_{y\in Y, s\in S} p_s \cdot V^{(k)}_{y,s}$ **then**
  set $\zeta_{up} \leftarrow U^{(k)} + \sum_{y\in Y, s\in S} p_s \cdot V^{(k)}_{y,s}$
end if
set $k \leftarrow k + 1$
return to *Step 1*

case, SPs are always feasible since the loss-of-load $Lss_{y,s,t}$ serves as a slack variable for unmet demand.

$$\min V^{(k)}_{y,s} \tag{3a}$$

$$\text{s.t. } V^{(k)}_{y,s} = \sum_{t\in T} Lss_{y,s,t} \cdot c^{lss} + \sum_{i\in I, t\in T} Gen_{y,s,i,t} \cdot c^{var}_{y,s,i} \quad \forall y \in Y, s \in S \tag{3b}$$

$$Gen_{y,s,i,t} \le cf_{s,i,t} \cdot Capa^{ge}_{y,i} \quad \forall y \in Y, s \in S, i \in I^{ge}, t \in T \tag{3c}$$

$$St^{out}_{y,s,i,t} \le Capa^{st}_{y,i} \quad \forall y \in Y, s \in S, i \in I^{st}, t \in T \tag{3d}$$

$$St^{in}_{y,s,i,t} \le Capa^{st}_{y,i} \quad \forall y \in Y, s \in S, i \in I^{st}, t \in T \tag{3e}$$

$$St^{size}_{y,s,i,t} \le Capa^{size}_{y,i} \quad \forall y \in Y, s \in S, i \in I^{st}, t \in T \tag{3f}$$

$$dem_{y,s,t} = Lss_{y,s,t} + \sum_{i\in I^{se}} Gen_{y,s,i,t} - \sum_{i\in I^{st}} St^{in}_{y,s,i,t} + St^{out}_{y,s,i,t} \quad \forall y \in Y, s \in S, t \in T \tag{3g}$$

$$St^{size}_{y,s,i,t} = St^{size}_{y,s,i,t-1} + St^{in}_{y,s,i,t} - St^{out}_{y,s,i,t} \quad \forall y \in Y, s \in S, i \in I^{st}, t \in T \tag{3h}$$

$$Capa_{y,i} = Capa^{(k)}_{y,i} \;:\; \lambda^{(k)}_{y,s,i} \quad \forall i \in I \tag{3i}$$

Step 3.2 computes the Benders cuts $\Omega_{k,y,s}$ defined by Eq. (4) and adds them to the MP. Each cut improves the estimator $\alpha_{y,s}$ based on the exact result of the SP $V^{(k)}_{y,s}$ and its subgradient $\lambda^{(k)}_{y,s,i}$ at the point $Capa^{(k)}_{y,i}$. In other words, by adding hyperplanes restricting the estimator, BD performs a piece-wise linear approximation of the unknown but convex function that assigns capacities to SP-costs (Conejo et al., 2006). In the literature, cuts of this form are termed optimality cuts as opposed to feasibility cuts that the algorithm adds when the SP is infeasible, which cannot occur in our case. Section 4 discusses the bundle management in Step 4 already included in the algorithm here.

$$\Omega_{k,y,s} : \alpha_{y,s} \ge V^{(k)}_{y,s} + \sum_{i\in I} \lambda^{(k)}_{y,s,i} \cdot (Capa_{y,i} - Capa^{(k)}_{y,i}) \tag{4}$$

Once all SPs are solved, summing expansion costs U and accurate operational costs $V^{(k)}_{y,s}$ in Step 5 gives the total costs at $Capa^{(k)}$. If these undercut the current ζ_{up}, $Capa^{(k)}$ is a new best solution, and we adjust ζ_{up} accordingly. Finally, the iteration counter increases by one, and the next iteration starts if the convergence test in Step 1 fails.

3. Stabilization

Stabilization is the pivotal refinement of the developed BD algorithm. The stabilization methods discussed below are not restricted to stochastic programming but derive from a much broader field of literature concerned with cutting plane methods. Cutting plane methods iteratively minimize convex functions, which may be non-differentiable, by approximating the function through outer linearizations (Kelley, 1960). Consequently, BD is a cutting plane method for two-stage problems.

In the next section, we describe how the BD algorithm introduced in the previous section can incorporate stabilization. Afterwards, we introduce three groups of methods for stabilization and strategies to parameterize them.

3.1. Stabilized Benders algorithm

We begin by representing the BD algorithm above in a more general form to connect it to the literature on stabilization in non-smooth convex optimization.

Let $\phi_{y,s}(Capa)$ be the optimal objective value of the SP for investment year y and scenario s as a function of complicating variable choice $Capa$. This function and its weighted average are polyhedral (and thus convex and non-smooth), which allows constructing a cutting plane model (Shapiro, Dentcheva, & Ruszczyński, 2009, Prop. 2.2 & 2.3).

The cutting plane approximation of $\phi_{y,s}(Capa)$ at iteration k corresponds to the minimum of the estimator $\alpha_{y,s}$ restricted by the cuts $\Omega_{k',y,s}$, as shown in Eq. (5). Each cut, representing a supporting hyperplane of $\phi_{y,s}$, is defined by Eq. (4). This formulation is equivalent to the piece-wise maximum of all linearizations across the domain of $Capa$, as the convex optimization literature usually defines the cutting plane model.

$$\hat{\phi}^{(k)}_{y,s}(Capa) = \min_{k>k'} \{\alpha_{y,s} \mid \Omega_{k',y,s}\} \tag{5}$$

The sum of approximations over all SPs $\hat{\phi}^{(k)}(Capa)$ can accordingly be written as Eq. (6).

$$\hat{\phi}^{(k)}(Capa) = \sum_{y\in Y, s\in S} p_s \hat{\phi}^{(k)}_{y,s}(Capa) \tag{6}$$

By sending a candidate solution $Capa^{(k)}$ to an *oracle*, which corresponds to solving the SPs in our case, we obtain $\sum_{y,s} p_s \phi_{y,s}(Capa^{(k)})$ and subgradients $\lambda^{(k)}_{y,s,i}$, which are in the subdifferentials of the respective $\phi_{y,s}(Capa^{(k)})$. Subgradients generalize the concept of gradients to non-differentiable functions. g is a subgradient of $f(x)$ if $f(y) \ge f(x) + \langle g, y - x\rangle \; \forall y$. The subdifferential of $f(x)$ is the set of all subgradients at x. The subdifferential is a singleton if the function is differentiable at x. On this basis, we improve the cutting plane model in Eq. (6) by adding new cuts and then solve the MP, $\min U(Capa) + \hat{\phi}^{(k)}(Capa)$, to obtain a new candidate solution $Capa^{(k)}$.

Although the proof of convergence for $\min \hat{\phi}(Capa) \to \min \phi(Capa)$ is simple (Kelley, 1960), cutting plane algorithms suffer from two computational issues. First, the algorithm is inherently unstable since the capacities $Capa$ in two consecutive iterates can be extreme points of the feasible space, arbitrarily far apart. In energy planning, this problem is particularly pronounced because the feasible region is hardly restricted and completely continuous. Even enforcing the optimum as a starting

value may not improve the solution time since the algorithm is initially unable to evaluate the quality of the solution. Second, the size of the model approximating $\phi(Capa)$, in the case of BD the number of cuts, can become prohibitively large as the number of iterations k increases and slow the algorithm (Frangioni, 2020). We emphasize that these issues are mutually reinforcing: Oscillation increases the number of iterations, which in turn slows down the convergence of the approximating model $\hat{\phi}^{(k)}(Capa)$.

The term "bundle methods" encompasses methods to address these two issues by managing the "bundle" of information accumulated throughout the iterations (Frangioni, 2002). In the following, our exclusive focus will be on stabilization methods that center each iteration around a reference point to prevent oscillation (Bonnans, Gilbert, Lemaréchal, & Sagastizábal, 2006). Since these methods improve convergence at the expense of adding complexity to the MP, they are well-suited to the examined problem with a simple MP but costly oracles, viz., large individual SPs. Removing non-binding cuts after a certain number of iterations addresses the second issue, as further discussed in Section 4. Additionally, stabilization reduces the overall number of iterations, thereby mitigating problems associated with excessive growth of the bundle as well.

All stabilization methods in the literature on (non-smooth) convex optimization build on the cutting plane method and aim to mitigate oscillation by keeping the next iterate $Capa^{(k)}$ relatively close to a *stability center*, a reference point in the (feasible) space of complicating variables. Commonly, this point is part of the sequence of previous iterates $\{Capa^{(j)}\}_{j=1,\dots,k-1}$. In theory, various measures can quantify the distance between the stability center and the next iterate, but the literature typically relies on the Euclidean norm. For the proximal bundle method, discussed below, the use of the Euclidean distance is justified by constructing a "poor man's Hessian" approximating valuable second-order information otherwise absent in the polyhedral setting. Frangioni (2020).

In the problem described in Section 2.1, the capacity vector $Capa$ is implicitly defined by the selected expansion path (cf. Eq. (1d)). In practical applications, many elements of the capacity vector are fixed constants. Therefore, defining the stability components below in terms of the lower-dimensional vector of expansion decisions Exp is computationally advantageous.

An update of the stability center to the current iterate is called a *serious step*, and an iteration that keeps the current stability center is called a *null step*. Nevertheless, a null step is still beneficial since it improves the cutting plane model locally. The literature offers various methods to distinguish between null and serious steps. For example, a *serious step* can be a step that improves the objective by a specified value compared to the current stability center. In an alternative definition, the ratio between the actual improvement determined by $\phi(Capa)$ and the improvement estimated by $\hat{\phi}(Capa)$ must exceed a certain threshold. In our case, stabilization achieves the best results when the algorithm makes a *serious step* in each iteration that improves the current best solution, even marginally.

Algorithm 2 represents our general implementation of stabilization with BD. The following sections will extend this general form with specific stabilization methods and corresponding parameter strategies.

Step 0.2 of the algorithm produces an initial feasible solution. In this study, we solve a deterministic instance of the model with the most probable scenario. Section 6 goes more in-depth on testing different initialization strategies. Step 2 obtains the next iterate by solving the stabilized MP, but the algorithm still solves the original MP in each iteration to obtain a lower bound. The variable q counts the number of serious steps and is an input to the Parameter strategies discussed below. Again, to be concise, we show a parallelized version of the algorithm here but discuss the parallelization in Section 4.3.

As discussed above, we did not include a minimum descent target. However, replacing the condition of Step 5 with the equation below can easily implement this feature:

$$U^{(k)} + \sum_{y,s} p_s V^{(k)}_{y,s} < \zeta_{up} - \omega_k$$

Algorithm 2: General parallel stabilized Benders algorithm

Step 0.1 (Parameter Initialization)
Set convergence tolerance $\epsilon > 0$, deletion threshold η, $\zeta_{low} \leftarrow 0$, $\zeta_{up} \leftarrow \infty$, set dynamic parameter for chosen bundle method $\in \{\tau_1, \gamma_1, \ell_1\}$, define stability center $Exp^{up} \leftarrow \mathbf{0}$, init. serious step counter $q = 0$, set $k = 1$
Step 0.2 (Stability center initialization)
if *initialize stability center true* **then**
  solve **CP** for $S' := \{s \in S | p_s = max(p_s)\}$; obtain $Exp^{(0)}$
  do in parallel
    $\forall \mathbf{y}, \mathbf{s}$: solve $\mathbf{SP}_{y,s}$ with $Capa^{(0)}$; obtain $V^{(0)}_{y,s}, \lambda^{(0)}_{y,s,i}$, compute $\Omega_{0,y,s}$ add to stabilized **MP** and **MP**; $\delta_{0,y,s} = 0$
  end
  $Exp^{up} \leftarrow Exp^{(0)}$
end if
Step 1 (Convergence test)
if $1 - \frac{\zeta_{low}}{\zeta_{up}} < \epsilon$ **then** *stop* **end if**
Step 2 (Obtain new iterate)
Solve stabilized **MP** and obtain $Capa^{(k)}$, $Exp^{(k)}$ and $U^{(k)}$
do in parallel
  Step 3.1 (Oracle call: solve subproblems)
  $\forall \mathbf{y}, \mathbf{s}$: solve $\mathbf{SP}_{y,s}$ with $Capa^{(k)}$; obtain $V^{(k)}_{y,s}, \lambda^{(k)}_{y,s,i}$, compute $\Omega_{0,y,s}$ add to stabilized **MP** and **MP**; $\delta_{k,y,s} = k$
  Step 3.2 (Obtain lower bound)
  solve **MP**, set $\zeta_{low} \leftarrow obj(\mathbf{MP})$ and obtain $U^{(k)}_{unstab}$
end
Step 4 (Bundle management)
for $y, s, l \in \{0, \dots, k-1\}$ **do**
  if $\Omega_{l,y,s}$ *is binding* **then**
    $\delta_{l,y,s} = k$
  else if $k - \delta_{l,y,s} > \eta$ **then**
    delete $\Omega_{l,y,s}$ from **MP** and stabilized **MP**
end for
Step 5 (Descent test)
if $U^{(k)} + \sum_{y,s} p_s V^{(k)}_{y,s} < \zeta_{up}$ **then**
  $\zeta_{up} \leftarrow U^{(k)} + \sum_{y,s} p_s V^{(k)}_{y,s}$
  $Exp^{up} \leftarrow Exp^{(k)}$
  $q = q + 1$
else
  $q = 0$
end if
Step 6 (Dynamic parameter adjustment):
Follow steps of adjustment algorithms below
$k \leftarrow k + 1$ **Return to *Step 1***

where ω_k gives the descent target in iteration k. We refer to Frangioni (2020) for an introductory discussion of a commonly used Armijo-type rule.

3.2. Bundle methods

There are three relevant bundle methods for the stabilization of cutting plane algorithms and each method features at least one dynamic parameter that governs the restrictiveness of the stabilization. While all presented methods are theoretically equivalent in the sense that there exists a combination of respective parameter choices that results in the exact next iterate from each bundle method (see Bonnans et al., 2006, Thm. 10.7 for a proof), their practical performance hinges critically on the definition of a parameter strategy, also referred to as t-strategy (Frangioni, 2002). Finding good t-strategies is not trivial, and selecting a poor one may even produce inferior results compared to an unstabilized cutting plane algorithm. For instance, an overstabilized MP, will not benefit from the global nature of the cutting plane model and may take more iterations than an unstabilized version (cf. Frangioni, 2020, Fig. 3.2). Strategies suggested in the literature are usually

comprised of a set of rules that determines the parameter values for the next iteration dependent on a set of variables describing the algorithm's progress. For example, such a rule could prescribe that the dynamic parameter doubles after l null steps and halves after m serious steps. In this simple strategy l and m are *hyperparameters* subject to calibration.

In this section, we describe the bundle methods and corresponding parameter strategies. Parameter strategies depend on the method and replace Step 6 in the previously outlined algorithm. Finding good parameter strategies is complex and can be highly problem-specific (Frangioni, 2020). We highlight the hyperparameters used in each strategy before testing specific values in Section 6.

3.2.1. Proximal-bundle methods

The first bundle method is called *proximal bundle method* (PBM) (Lemaréchal, 1977). This variant adds a penalty term to the MP's objective function to prevent moving away from the stability center. Two-stage stochastic problems widely use this method, for example in Oliveira, Sagastizábal, and Scheimberg (2011) or Zverovich et al. (2012). The stabilized MP is a quadratic program and is given by:

$$\min_{\{Capa,Exp\}} U(Capa) + \hat{\phi}^{(k)}(Capa) + \frac{1}{2\tau_k}\|Exp - Exp^{up}\|_2^2 \quad s.t. \quad Capa_{y,i} = \sum_{y' \in Y_y} Exp_{y',i} \;\forall y, i \tag{7}$$

However, PBM can go beyond the l_2-norm. Pessoa, Sadykov, Uchoa, and Vanderbeck (2018) use a generalized linear PBM approach using a composite penalty term consisting of three terms. This approach does not penalize small deviations determined by the l_∞-norm, but penalizes deviations outside this confidence interval based on the l_1-norm. As the penalty parameter goes to infinity, this approach turns into the Boxstep method we will introduce in Section 3.2.3. Balancing rigor and practicality for the problem at hand, we limited our computational study to the standard PBM method.

Parameter strategies. The dynamic parameter τ_k in Eq. (7) in PBM scales the penalty in the objective function with larger values decreasing it; and smaller values increasing it.

Following de Oliveira and Solodov (2016), we use two ruled-based parameter strategies first proposed by Kiwiel (1990) and Lemaréchal and Sagastizábal (1997). We implement the strategy by inserting the steps of Algorithm 3 into Step 6 of Algorithm 2. Both strategies decrease the weight in response to a serious step and increase it in response to a null step (Step 6.3). However, the methods compute the auxiliary parameter τ_k^{aux} (Step 6.2) differently. For both methods, the auxiliary parameter mimics the Hessian used in a Newton-type algorithm on a Moreau envelope of the true function ϕ. Of course, this function is unknown and we can only approximate the optimal descent. For details, we refer to Frangioni (2020) and Hiriart-Urruty and Lemaréchal (1993), Chapter XV.

In PBM-1, the auxiliary parameter uses the ratio between the achieved descent and the descent predicted by the model (Kiwiel, 1990). PBM-2 derives directly from a quasi-Newton formula and requires some safeguards to ensure that the numerator of the fraction term remains non-negative (Lemaréchal & Sagastizábal, 1997). Both methods require the following hyperparameters: the starting value τ_1, the minimum parameter value τ_{min} and the scaling factor a.

3.2.2. Level-bundle methods

The level bundle method (LBM) is the opposite of PBM and the trust-region methods discussed below (Frangioni, 2020). Instead of maximizing the predicted descent of the model $(U^{up}+\phi(Capa^{up}))-(U^{(k)}+\hat{\phi}^{(k)}(Capa^{(k)}))$ subject to a constraint or a penalty on $\|Capa^{(k)}-Capa^{up}\|$, LBM determines a minimum descent target and minimizes the deviation from the stability center needed to reach it (Lemaréchal, Nemirovskii, & Nesterov, 1995). The stabilized MP for LBM is given by:

$$\min_{\{Capa,Exp\}} \frac{1}{2}\|Exp - Exp^{up}\|_2^2 \quad s.t. \quad U(Capa) + \hat{\phi}^{(k)}(Capa) \le \ell_k, \quad Capa_{y,i}^{up} = \sum_{y' \in Y_y} Exp_{y',i} \;\forall y, i \tag{8}$$

Algorithm 3: Dynamic adjustment of PBM parameter τ_k

Step 6.1 (Initialization):
initialize hyperparameters a, τ_{min}, τ_1
Step 6.2 (Define auxiliary parameter):
$\tau_k^{aux} \leftarrow 2\tau_k\left(1 + \frac{U(Capa^{up})-U(Capa^{(k)})+\phi(Capa^{up})-\phi(Capa^{(k)})}{U(Capa^{up})-U(Capa^{(k)})+\phi(Capa^{up})-\hat{\phi}^{(k)}(Capa^{(k)})}\right)$ **if** method is *PBM-1*

$\tau_k^{aux} \leftarrow \tau_k\left(1 + \frac{\sum_{y,s} p_s\left[\sum_i (\lambda_{y,s,i}^{(k)} - \lambda_{y,s,i}^{(k-1)})\cdot(Capa_i^{(k)} - Capa_i^{(k-1)})\right]}{\sum_{y,s} p_s\left[\sum_i (\lambda_{y,s,i}^{(k)} - \lambda_{y,s,i}^{(k-1)})^2\right]}\right)$ **if** method is *PBM-2*
Step 6.3 (Update dynamic parameter based on step type)
if *If* $q = 0$ **then**
  $\tau_{k+1} \leftarrow \min\{\tau_k, \max\{\tau_k^{aux}, \tau_k/a, \tau_{min}\}$
else
  if $q>5$ **then**
    $\tau_k^{aux} \leftarrow a\tau_k^{aux}$
  end if
  $\tau_{k+1} \leftarrow \min\{\tau_k^{aux}, 10\tau_k\}$
end if

The level parameter ℓ_k sets the maximum MP objective for the next iterate. It determines level sets of the cutting plane algorithm, i.e., decisions for capacity expansion for which the algorithm estimates function values below ℓ_k.

The level set is empty, and the problem is infeasible if the objective of the MP is strictly above the prescribed level for all feasible values of Exp. This information is still valuable since ℓ_k is a lower bound to the overall problem in this case. Therefore, one way to update the lower bound is to set it equal to ℓ_k if an iteration resulted in an empty level set (van Ackooij & de Oliveira, 2015). Alternatively, one can update the lower bound by solving the original and stabilized MP in each iteration. One potential advantage of LBM over PBM is that the dynamic parameter ℓ_k is in the same order of magnitude as the objective function, which facilitates setting it to appropriate values.

As an extension, de Oliveira and Solodov (2016) propose a doubly stabilized bundle method that combines PBM and LBM. This method automatically chooses between proximal and level iterates. The latter occurs when the level set constraint is binding, while proximal iterations find reasonable solutions within a level set instead. The method has the advantage that the strategy to set τ_k can factor in the dual variable of the level constraint, but due to the method's complexity, we did not include it in our subsequent benchmark.

Parameter strategies. We devise two different parameter strategies for LBM. The first strategy, LBM-1, is straightforward and uses a fixed weight, β, to compute a convex combination of the current upper and lower bounds of the problem. We make this adjustment in every iteration, regardless of whether a serious or a null step preceded the adjustment. Algorithm 4 formalizes this simple mechanism based on Frangioni (2020). As discussed above, we compute the lower bound by solving the unstabilized MP in every iteration.

Algorithm 4: Dynamic adjustment of level parameter ℓ_k for LBM-1

Step 6.1 (Initialization):
initialize hyperparameter β
Step 6.2 (Define level parameter):
$\ell_{k+1} = \beta\zeta_{low} + (1-\beta)\zeta_{up}$

The second strategy, LBM-2, originates from Brännlund, Kiwiel, and Lindberg (1995). Two features of this strategy are more refined: First, it only sets the level to a convex combination of the upper and lower bounds if the previous iteration performed a serious step

($q > 0$). Second, we only increase the level bound in a null step if the level constraint has been binding as indicated by a non-negative dual variable of the constraint ρ_k. These refinements ensure that level descent parameter υ_k^{ℓ} is kept constant for a short sequence of null steps before it is eventually adjusted (cf. de Oliveira & Solodov, 2016 for additional reasoning on the approach). The hyperparameters for LBM-2 are β and μ_{max}.

Algorithm 5: Dynamic adjustment of level parameter ℓ_k for LBM-2

Step 6.1 (Initialization):
initialize hyperparameters: weight parameter β and maximum dual threshold μ_{max}
Step 6.2 (Obtain dual of level set constraint):
Let ρ_k be the current dual solution belonging to constraint $U(Capa) + \hat{\phi}(Capa) \leq \ell_k : \rho_k$
Set $\mu_k = 1 + \rho_k$
Step 6.3 (Adjust level descent):
if $q>0$ **then**
  $\upsilon_{k+1}^{\ell} = \min\{\upsilon_k^{\ell}, (1-\beta)\cdot(\zeta^{up} - \zeta_{low})\}$
else
  if $\mu_k > \mu_{max}$ **then**
    $\upsilon_{k+1}^{\ell} = \beta \cdot \upsilon_k^{\ell}$
  end if
end if
Step 6.4 (Set level):
$\ell_{k+1} = \zeta^{up} - \upsilon_{k+1}^{\ell}$

3.2.3. Trust-region bundle methods

The final bundle method stabilizes the algorithm using a *quadratic trust-region* (QTR). The next iterate is constrained to be in a Euclidean hypersphere around the current stability center defined by a certain radius. The quadratically constrained stabilized MP is given by:

$$\min_{\{Capa,Exp\}} U(Capa) + \hat{\phi}^{(k)}(Capa) \quad s.t. \quad \|Exp - Exp^{up}\|_2^2 \leq \gamma_k, \quad Capa_{y,i}^{up} = \sum_{y' \in Y_y} Exp_{y',i} \; \forall y, i \tag{9}$$

Note that PBM is the Lagrangian relaxation of the QTR (Hiriart-Urruty & Lemaréchal, 1993), highlighting the theoretical equivalence of the methods.

The existing literature does not use QTRs, but commonly applies a linear alternative using the l_∞-norm named Boxstep method (Marsten et al., 1975). Although it keeps the stabilized MP linear, Frangioni and Gorgone (2014) show it is inferior to other bundle methods.

Parameter strategies. The dynamic parameter for the trust-region methods, QTR and Boxstep, is γ_k constraining deviations from the current stability center measured with the l_2-norm and the l_∞-norm, respectively.

For QTR, we describe the corresponding steps of the strategy in Algorithm 6. Rather than directly adjusting the radius itself, we introduce a scaling factor ψ_k, that defines the radius γ_k relative to the l_1-norm of the current stability center. As a result, the radius changes in response either when the dynamic parameter changes or when a serious step changes the stability center.

We illustrate the stabilization based on four consecutive iterations for a stylized example in Fig. 4. It includes three unbounded expansion variables, restricted by a spherical trust-region defined by the constraint of Eq. (9).

The current stability center $Exp^{\mu p}$ is indicated in orange, and the solution at the current iteration $Exp^{(k)}$ in yellow or red, depending on whether it improves the current best or not. The minimizer of $U(Capa)+\phi(Capa)$ is shown in green. In the first iteration, the distance between the origin and the initial stability center determines the sphere's radius. The first iteration in (a) results in an objective above the current best,

Algorithm 6: Dynamic adjustment of radius γ_k

Step 6.1 (Initialization):
initialize hyperparameters $\psi_1, \psi_{min}, \kappa \in (0,1], \varepsilon > 0$
Step 6.2 (Reduction test):
if $\left|1 - \frac{U_{unstab}^{(k)}}{U^{(k)}}\right| < \varepsilon$ **then**
  $\psi_{k+1} \leftarrow \min\{\psi_{min}, \psi_k \cdot \kappa\}$
else
  $\psi_{k+1} \leftarrow \psi_k$
end if
Step 6.3 (Update radius)
$\gamma_{k+1} \leftarrow (\psi_{k+1})^2 \cdot \|Exp^{up}\|_1^2$

but the trust-region is binding, as determined in Step 6.2, and remains unchanged. As the cutting plane model improves locally, the second iteration (b) results in a serious step. Re-centering the trust-region for the third iteration leads to a radius adjustment due to the new l_1-norm of the stability center. The third iteration in (c) results in a null step, but the trust-region is no longer binding, and the factor ψ_k is reduced by factor κ. The fourth iteration with a smaller trust-region in (d) does not improve the solution.

Since a small radius will result in numerical issues, we impose a minimum on the dynamic parameter ψ_k, ψ_{min}. The hyperparameters include the starting radius scaling factor ψ_1, the minimum scaling factor ψ_{min}, and the reduction factor κ. In addition, there is a threshold ε to check if the trust-region is non-binding since the original and stabilized MP will rarely have the same objective, even if the trust-region is non-binding due to numerical inaccuracies.

Adding a rule for increasing the radius again is also conceivable because a small starting radius scaling factor ψ_1 can result in slow convergence. However, this implies additional hyperparameters, and in practice, it is much easier to correct a small starting factor if the first iterations show slow progress.

For the Boxstep method, we test a range of different box sizes, which we also denote by ψ. We define the box size relative to the values for the stability center, so the current values impact the absolute boundaries imposed by the trust-region, resulting in a noncubical box (Marsten et al., 1975). In order to avoid problems with values of zero, we also impose a minimum upper limit.

4. Additional refinements

In addition to stabilization, there are various methods to improve the poor convergence of the basic BD algorithm. Most refinements, however, aim at problems with a complex combinatorial MP rather than large SPs, like the two-stage planning problem examined in this paper.

One refinement already included in Algorithm 1 and 2, is deleting non-binding cuts after a predefined number of iterations. The literature on bundle methods refers to such strategies as β-strategies (Frangioni, 2002). In the algorithms, $\delta_{k,y,s}$ tracks in which iteration a cut was created or binding the last time. Step 4 loops over all previous cuts, updating $\delta_{k,y,s}$ and deleting cuts that were not created or binding within the last η iterations.

Furthermore, the outlined algorithms already feature several practical refinements. All capacity variables have an upper limit that is far beyond any reasonable value to bound the solution space of the MP. We scale all equations according to the methodology outlined in Göke (2021a) to improve numerical properties. If scaling does not suffice, small values of $Capa_{y,i}^{(k)}$ in the SPs are rounded off to zero. The same applies to $\lambda_{y,s,i}^{(k)}$ when adding cuts to the MP. For storage, we enforce an upper and lower bound on the ratio between storage and energy capacity that reflects technical restrictions and confines the solution space.

The following sections discuss additional refinements to the stabilized BD in Algorithm 2, namely, inexact oracles, valid inequalities, and parallelization.

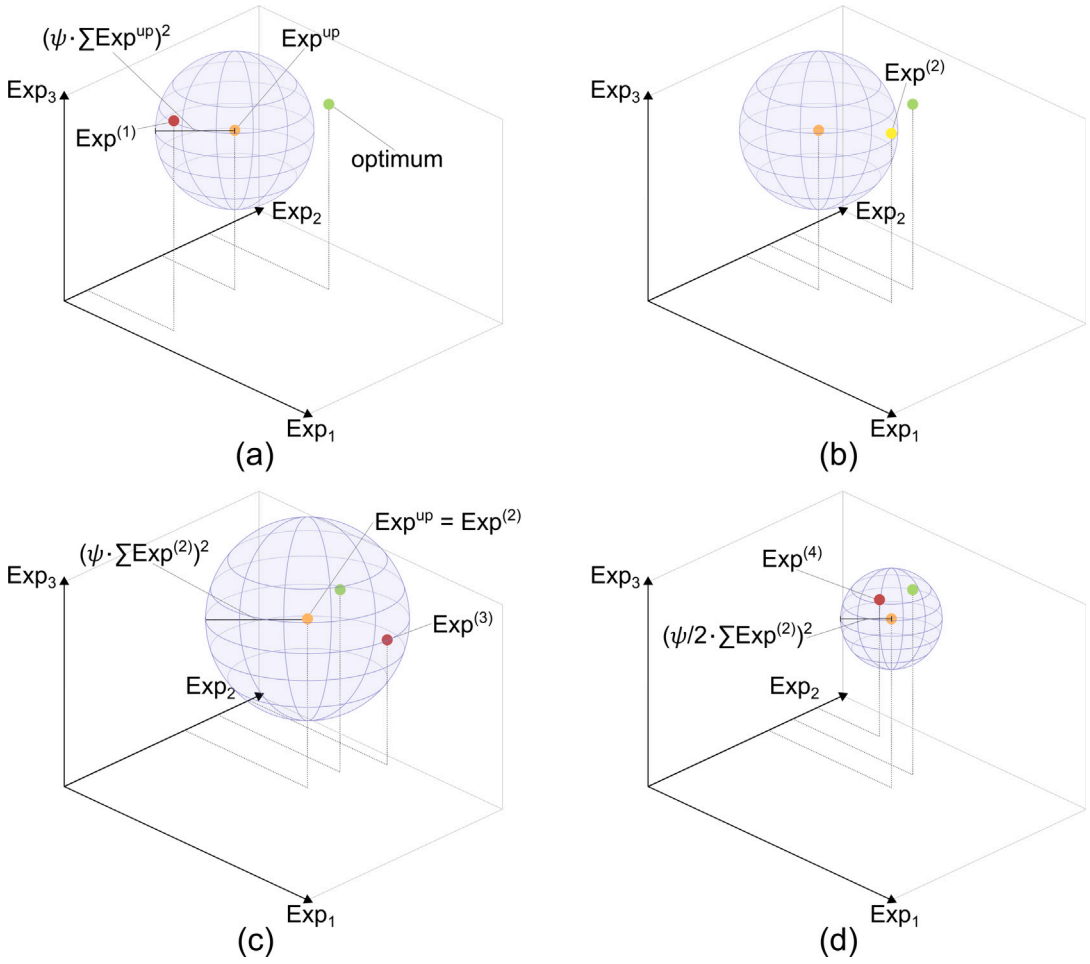

Fig. 4. Dynamic trust-region adjustment.

4.1. Inexact oracles

A large body of literature addresses bundle methods with oracles that return inexact information (de Oliveira, Sagastizábal, & Lemaréchal, 2014; de Oliviera & Solodov, 2020; van Ackooij & de Oliveira, 2015). Although it is generally possible that oracles cannot obtain exact information, in our application, the oracles deliberately only return inexact information $\{\phi_{y,s}(Capa), \lambda_{y,s,i}\}$ to reduce the computational time of solving the SPs.

In practice, we implement this feature by deactivating the crossover step of the Barrier algorithm. As a result, the SPs will return a feasible interior point instead of the optimal basic solution. In addition, we test an asymptotically exact oracle strategy by decreasing the Barrier convergence tolerance as the optimality gap closes, eventually reaching the desired optimality level (van Ackooij & de Oliveira, 2015; Zakeri et al., 2000). This strategy faces a trade-off: On the one hand, increasing the tolerance reduces the computation time of the SPs, but on the other hand, it result in less accurate cuts and a potential increase of iterations. Against this background, Fig. 5 presents three different interpolation methods for this process. In all cases, the algorithm starts at a tolerance of $1e^{-2}$ and decreases to $1e^{-8}$, but the reduction follows an exponential, linear, or logarithmic curve. We can easily implement this approach in Algorithm 2 without further changes.

Beyond the implemented approaches, on-demand oracle accuracy could improve the algorithm further (de Oliveira & Sagastizábal, 2014). With this method, interaction with the solver enables an assessment if the next iterate from Step 2 of Algorithm 2 will result in a serious step. However, practical implementations in the context of two-stage

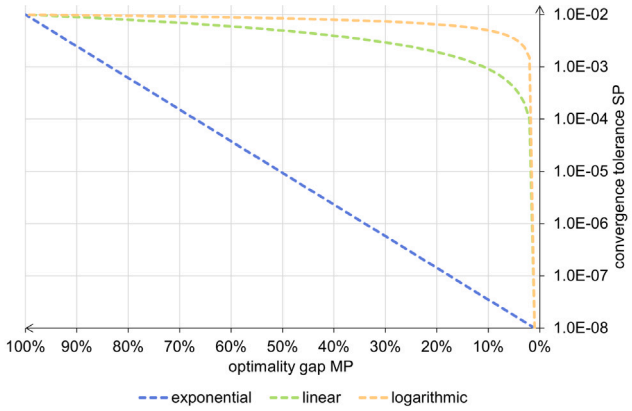

Fig. 5. Interpolation methods for convergence tolerance of SPs.

stochastic problems focus on cases where the use of multi-cuts significantly impact the MP's computation time. In our problem, this is not the case. The number of scenarios is small compared to the number of constraints in the MP, suggesting that multi-cuts will not deteriorate the performance of the MP (Fábián, 2013; Wolf, Fábián, Koberstein, & Suhl, 2014). Furthermore, the computational results show that the computation time of the MP is insignificant.

4.2. Valid inequalities

Valid inequalities (VI) are constraints added to the MP that are redundant in the closed formulation of the problem but effectively reduce the solution space of the MP (Cordeau et al., 2006). Thus, valid inequalities aim to improve the convergence of BD at the expense of increasing the size and solution time of the MP. Often, valid inequalities are derived from the SPs avoiding their infeasibility.

In the examined two-stage problem, the MP does not include an energy balance and will frequently underinvest in generation capacity, particularly in early iterations. To address this, Wang et al. (2013) add one scenario's operational constraints, including the energy balance, to the MP. In our case, this approach is not viable due to the size of the individual scenarios and corresponding SPs. Instead, we add aggregated operational constraints in Eqs. (10a) and (10b) to the MP for each scenario.

$$\sum_{t\in T} dem_{y,s,t} \leq \sum_{i\in I^{ge}} Gen_{y,s} \qquad \forall y \in Y, s \in S \tag{10a}$$

$$Gen_{y,s,i} \leq \sum_{t\in T} cf_{s,i,t} \cdot Capa^{ge}_{y,i} \quad \forall y \in Y, s \in S, i \in I^{ge} \tag{10b}$$

The first Eq. (10a) enforces a yearly energy balance, setting a lower bound for generation that is binding when hourly generation patterns precisely match demand without relying on storage methods that incur energy losses. The second Eq. (10b) establishes a connection between the lower limit on generation and the capacity variables. As discussed in Section 2.1, the problem formulation in the paper is still stylized and omits the multiple regions in the case study connected by transmission infrastructure. Therefore, the applied valid inequalities are aggregated by region, assuming lossless and unrestricted transmission as a lower bound.

As highlighted in Section 2.1, the problem formulation in the paper remains stylized and does not account for the multiple regions connected by transmission infrastructure in the case study. Consequently, the applied valid inequalities also aggregate demand and generation by region, assuming lossless and unrestricted transmission as a lower bound.

4.3. Parallelization

As shown in Algorithm 1 and 2, we parallelize solving the SPs. As a result, we can also solve the original MP in every iteration when solving the SPs to obtain a lower bound for the algorithm at no computational costs since the MP solves much faster than any of the SPs. This computation of the lower bounds is common for LBM methods with simple MPs relative to the SPs (de Oliveira & Solodov, 2016). It also enables us to implement the parameter strategy for the quadratic trust region.

The efficiency of parallelization is the observed speed-up divided by the number of distributed computation nodes, which corresponds to the number of SPs in our case. In the ideal case, parallelization has no overhead, and all SPs require the same time to converge, resulting in an efficiency of one.

5. Case study

This section introduces the case-study used to test and benchmark the algorithms introduced previously. The ambition of the case-study is not to achieve highly accurate system planning but to provide a plausible problem covering all critical elements of renewable energy systems. The public repository linked in the Supplementary material section provides all files to reproduce the case-study.

The applied planning model includes two expansion years, 2030 and 2040, focusing on the power sector. The planned system is not subject to any emission constraints in 2030 but must fully decarbonize by 2040 to cover a diverse range of system setups. Spatially, the model covers four distinct regions: France, Belgium, the Netherlands, and Germany.

Fig. 6 introduces all considered technologies, depicted as gray circles, and their interaction with energy carriers, depicted as colored squares. Exogenous demand in the model is limited to the carrier electricity modeled at an hourly resolution for the entire year. Hydrogen uses a daily resolution; oil and gas a yearly resolution.

In the graph, entering edges of technologies refer to their input carriers; outgoing edges relate to outputs. For example, the generation technology electrolysis uses electricity as an input to generate hydrogen. Storage technologies, like pumped storage, have an entering and an outgoing edge to represent charging and discharging. Beyond pumped storage, the model includes batteries for short-term and hydrogen tanks for long-term energy storage. Including long-term storage is critical since it is key for balancing seasonal fluctuations in decarbonized systems.

Wind and PV capacity are subject to an upper limit that restricts expansion. Pumped storage, run-of-river, and hydro reservoirs have capacities fixed to today's levels. Furthermore, hydro reservoirs are modeled as storage with an exogenous inflow instead of flexible charging. Beyond the listed technologies, the model also decides on the expansion and operation of transmission to exchange electricity and hydrogen between regions.

6. Results

The first two sections compare the different stabilization methods and additional refinements. Subsequently, we benchmark selected configurations to an off-the-shelf solver for the deterministic equivalent. This benchmark solves the case study while varying the number of stochastic scenarios from 2 to 16. For BD, the MP and each SP run on independent computing cluster nodes, each with four cores and 16 GB of memory. The number of nodes amounts to one for the MP plus twice the scenario number for the SPs since the case study covers two expansion years, 2030 and 2040.

The technical specifications of the cores can differ between runs and nodes, but we re-ran unexpected performance outliers to ensure the robustness of the results. For solving problems, each node deploys the Barrier implementation of Gurobi 10.1 without crossover and the *NumericFocus* parameter set to zero. For BD, we delete unused cuts after 20 iterations and choose a convergence tolerance ϵ of 0.2%. The algorithm sometimes fails to converge at smaller tolerances due to rounding. All deployed variations of BD are implemented in the version of the AnyMOD.jl modeling framework linked in the Supplementary material.

Generally, total run-time depends on the number of iterations and the average time to solve the SPs since the MP is small. Appendix 9.2 provides detailed results on the number of iterations, average time-per-subproblem, and total computation time for all following sections.

6.1. Stabilization

Table 1 gives an overview of the tested configurations for the bundle methods and strategies introduced in Section 3. For LBM-2, PBM-1, and PBM-2, we adopted the configurations proposed in de Oliveira and Solodov (2016). For all other methods, we chose the configurations ourselves. The initial benchmark covers all permutations in the table below. Accordingly, we tested five configurations for LBM-1, 15 for LBM-2, and 27 for PBM-1 and PBM-2 each. For box-step the test covers 4 configurations; for QTR 7. Since this totals 85, initially, each configuration is only tested once using a model with four different scenarios. These tests do not use any refinements other than the respective stabilization.

Fig. 7 compares the algorithm's run time when deploying the different configurations. Colored symbols indicate the median and minimum

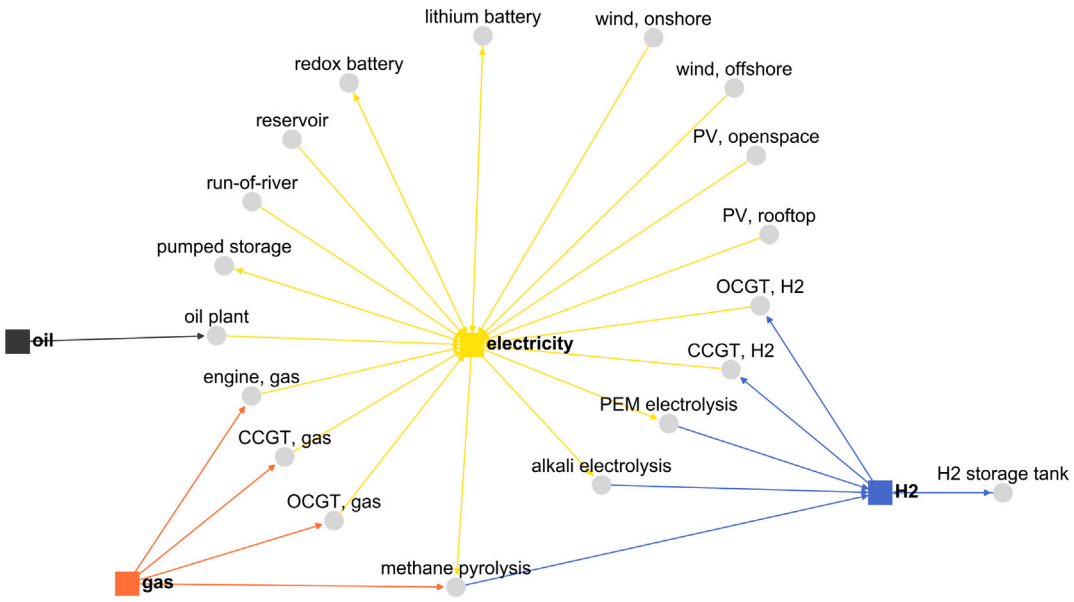

Fig. 6. Graph of model elements.

Table 1
Tested configurations for different stabilization methods and parameter strategies.

Method	Parameter strategy	Configurations
Level bundle method	LBM-1	$\beta \in \{0.125, 0.25, 0.375, 0.5, 0.625\}$
	LBM-2	$\beta \in \{0.2, 0.5, 0.7\}, \mu_{max} \in \{1, 5, 10, 50, 100\}$
Proximal bundle method	PBM-1	$a \in \{2, 4, 5\}, \tau_1 \in \{1, 5, 10\}$
	PBM-2	$\tau_{min} \in \{10^{-6}, 10^{-5}, 10^{-3}\}$
Box-step		$\psi \in \{2.5\%, 5\%, 7.5\%, 10\%\}$ upper limit at least 0.5 GW
Quadratic trust-region		$\psi_1 \in \{1, 0.5, 0.1, 0.05, 0.01, 0.005, 0.001\}$ $\psi_{min} = 1 \cdot 10^{-6}, \kappa = 0.5, \varepsilon = 5 \cdot 10^{-4}$

performance for each group of strategies. Without any stabilization, the computation time of BD amounts to 4'600 min, so all methods can substantially accelerate the algorithm, but effects greatly vary by method and configuration. As expected, stabilization improves performance by substantially reducing the number of iterations. For instance, the QTR with $\psi_1 = 0.01$ reduces the number of iterations to 30 from 660 without stabilization. The adverse impact on the MP is negligible since it accounts for less than 0.4% of the total run-time in both cases. Stabilization even decreases the average time per MP since quick convergence reduces the number of cuts added to the MP that continuously increase its size. The average time for the SPs fluctuates substantially across configurations but without any clear trend.

Overall, LBM-2 and PBM-2 are less performative than the other methods. For these methods and PBM-1, it is also challenging to identify a relation between the configuration and performance. For instance, in the case of PBM-1 and PBM-2, the first and second best configurations of both parameter strategies do not share any values. In addition, the spread of computation times is significant. For LBM-1, performance generally improves for smaller values of β that tighten the stabilization but the overall spread of results is small; results for BOX are similar. QTR also gives similar results until the initial radius ψ_1 reaches a threshold of 0.005. At this point, the radius is too tight, constraining the MP too much and slowing convergence.

In the following, we reduce the configurations to the two best for each method according to Table 2.

In the previous tests, we solve the model for the most probable scenarios with a daily instead of an hourly resolution to obtain an initial solution for the stabilization. In principle, stabilization benefits from a reasonable initial solution, but there is no strict relationship between

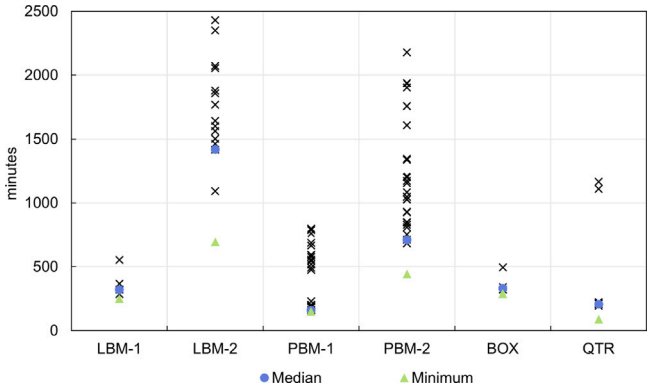

Fig. 7. Comparison of configurations for stabilization methods.

the quality of the initial solution and convergence time. Since the algorithm cannot identify an optimal solution in the beginning, poorer starting solutions can even result in a shorter computing time. In addition, obtaining better initial solutions will also increase pre-processing time.

Fig. 8 compares initialization with the most probable scenarios at a daily resolution on the right, to no initialization on the left, and initialization with the most probable scenarios but an hourly resolution in the middle. We run the tests for the selected subset of configurations, varying the number of scenarios from 2 to 8. Again, the results show a large spread; in some instances, initialization significantly impacts

Table 2
Two best configurations for each stabilization method and parameter strategies.

Method	Parameter strategy	Configurations	
Level bundle method	LBM-1	$\lambda = 0.125$	$\lambda = 0.25$
	LBM-1	$\lambda = 0.5, \mu_{max} = 100$	$\lambda = 0.7, \mu_{max} = 100$
Proximal bundle method	PBM-1	$a = 4.0, \tau_1 = 1.0, \tau_{min} = 10^{-6}$	$a = 5.0, \tau_1 = 10.0, \tau_{min} = 10^{-5}$
	PBM-2	$a = 5.0, \tau_1 = 5.0, \tau_{min} = 10^{-5}$	$a = 4.0, \tau_1 = 1.0, \tau_{min} = 10^{-3}$
Box-step		$\psi = 7.5\%$	$\psi = 10\%$
Quadratic trust-region		$\psi_1 = 0.05$	$\psi_1 = 0.01$

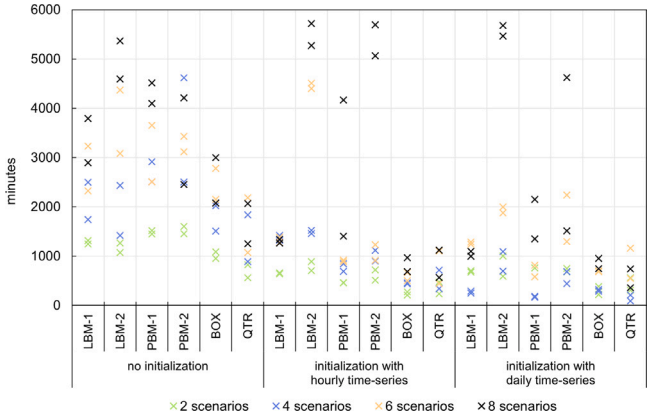

Fig. 8. Comparison of initialization strategies.

a specific configuration's performance. Such interactions are expected since the starting point is an important determinant for the stabilized iteration. However, predicting the effect initialization will have on a specific configuration is impossible. On average, stabilization with the most probable scenario and a daily resolution performs best. Therefore, we use this configuration in the following.

6.2. Additional refinements

In the next step, we analyze the impact other refinements have on the algorithm's performance.

First, Fig. 9 shows the impact the inexact-cut strategies introduced in Section 4.1 have on performance. As described, the four compared cases only vary the strategy for the convergence tolerance of the SPs. Since all skip the crossover phase of the Barrier algorithm that moves from an interior point to a basic solution, they all use inexact cuts. Without this feature, computation times are unreasonably slow, preventing an efficient benchmark.

The results show no clear benefit of inexact-cut strategies on performance. For specific scenarios and configurations, inexact cuts do reduce computation time. For instance, with six and eight scenarios, the logarithmic strategies achieve a median reduction of computation time by 33% and never significantly increase computation time. However, no clear trend is observed for the other scenario setups and inexact-cut strategies. For some configurations, non-constant strategies can even significantly increase computation time. What is particularly surprising is that in some cases, for example, in all tests with four scenarios, the average time per subproblem increases. In theory, inexact cuts reduce SP time but may increase the number of iterations. Due to the inconclusive results, we continue our benchmark using constant convergence tolerances.

Next, we compare the impact of valid inequalities in the MP as introduced in 4.2 on performance. Again, Fig. 10 compares performance for a different number of scenarios and configurations of stabilization. Overall, valid inequalities increase the spread of solution times without a robust overall reduction. Similar to stabilization, the increase of computation time for the MP is negligible, but we do not observe a consistent reduction of iterations. Even for specific stabilization methods, there is no identifiable trend. For instance, with four scenarios and QTR with a radius $\psi_1 = 0.05$, valid inequalities more than double computation time, but at six scenarios, reduce time by 16%. Therefore, we will not deploy valid inequalities in the subsequent benchmarks.

Finally, Fig. 11 shows the benefit of parallelization. As discussed in Section 4.3, parallelization does not strictly change the algorithm itself but distributes the SPs across different nodes to solve them in parallel. If the overhead is negligible, the refinement will strictly decrease computation time but requires additional computational resources. So, parallelization aims to reduce the average time to solve the SPs and not the number of iterations.

Across all tests, the efficiency of parallelization varies between 43% and 68% with an average of 57%. The overhead is negligible, and different solve times of the SPs account for almost the entire efficiency loss. To a certain extent, differences in the underlying years can explain the fluctuations of solve times, but mostly, they are random. As the number of scenarios increases, there is no added risk of outliers delaying the algorithm; efficiency stays constant, and the speed-up increases with the number of nodes.

6.3. Benchmark with the deterministic equivalent

In the final section, we perform a concluding evaluation of the refined Benders algorithm and compare it to solving the deterministic equivalent with an off-the-shelf solver. Such comparisons are always at risk of cherry-picking by comparing the best algorithm setup determined in an extensive series of tests against a single test with an off-the-shelf solver.

In the first step, we select four out of ten configurations for stabilization, as indicated by the blue shade in Table 2. This selection builds on the extensive testing with two, four, six, and eight scenarios in the previous sections. Apart from that, all four reference cases use daily initializations, constant convergence tolerances of the SPs, no valid inequalities, and parallelization.

Overall, the comparison of performance in Fig. 12 shows a massive speed-up of BD around a factor of 100 thanks to stabilization and parallelization. For two scenarios, the cherry-picked setup takes 29.4 min, outperforming the deterministic equivalent by 77.3 min, but the more robust reference cases still require between 132.2 (BOX) and 339.0 min (LBM-1). With eight scenarios, the reference cases take between 38.2 min (PBM-1) and 134.2 min (LBM-1) on average, outperforming the deterministic equivalent that requires 119 min.

However, the comparison is still biased since we selected the reference cases based on their performance with two to eight scenarios. Therefore, Fig. 13 tests the reference case "out-of-sample" with up to 16 scenarios. Fig. 14(a) shows the corresponding computation times for the deterministic equivalent. Fig. 14(b) describes how the matrix size for the deterministic equivalent in terms of rows, columns, and non-zeroes depends on the number of scenarios. Note that although the number of rows, columns, and non-zeroes increases linearly with the number of scenarios, the total matrix size corresponding to the product of rows and columns grows exponentially.

Computation time for the deterministic equivalent increases exponentially, although substantial outliers occur for 14 and 16 scenarios. The graph shows a significant fluctuation of results for the selected

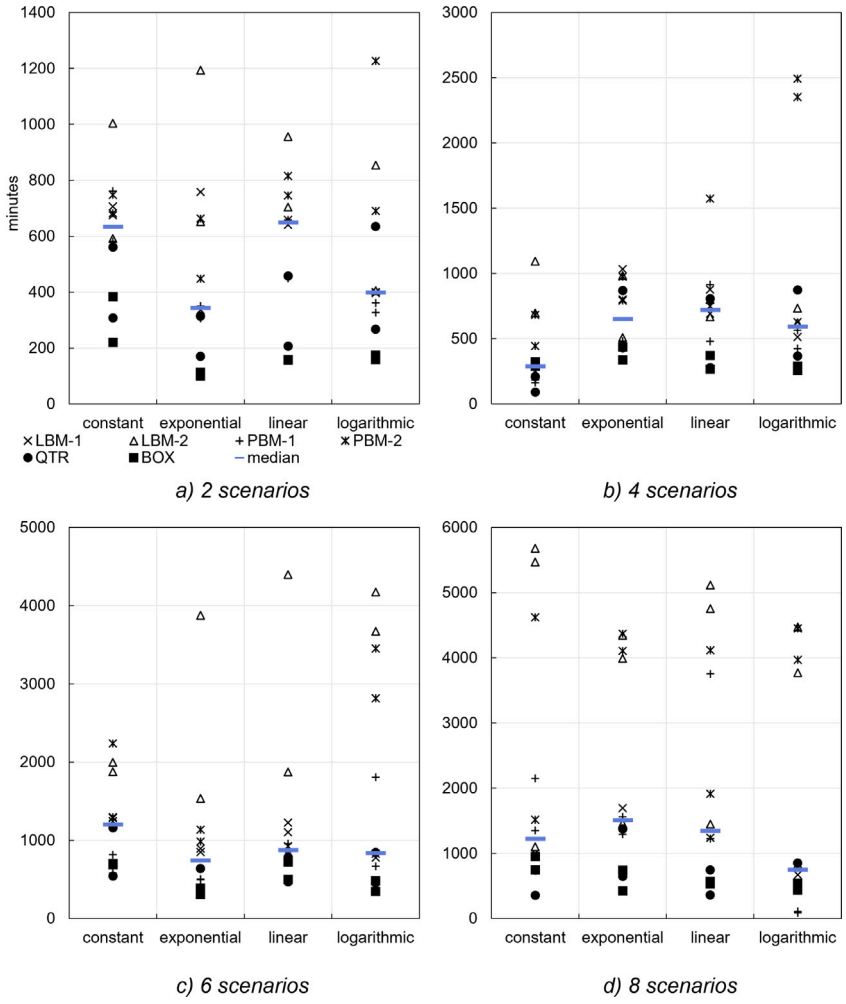

Fig. 9. Impact of inexact cuts.

proximal and level setups, and methods do not consistently outperform the deterministic equivalent at a certain threshold. For QTR and Boxstep, computation times barely fluctuate, and the selected setups reliably outperform the deterministic equivalent. QTR and Boxstep also showed little fluctuations when comparing different parameter configurations in Section 6.1, while the proximal method fluctuated substantially. There is no significant difference in solution quality between the refined BD algorithm and the deterministic equivalent. On average, objective values for the deterministic equivalent are 0.5‰ smaller.

This comparison has limitations since the deterministic equivalent and the parallelized algorithm utilize computational resources differently. In our case, the deterministic equivalent runs on a single node with 24 cores and 16 GB memory each, the most potent node available to us, while BD utilizes up to 33 nodes with 4 cores and 16 GB memory each. Beyond computational resources, performance can vary for other model setups beyond the range tested in this paper. Since the benchmark is not universally valid, the focus should not be on solution times but on scaling behavior.

7. Conclusion

This paper applied BD to two-stage stochastic problems for energy planning under climate uncertainty, a critical problem for designing renewable energy systems. To improve performance, we modify the standard algorithm and adapt various refinements to the problem's characteristics—a simple continuous MP, and few but large SPs. These refinements include inexact cuts, valid inequalities, and parallelization, but the focus is on stabilization methods to mitigate the oscillating behavior of BD. We test a broad range of bundle methods and configurations, including a quadratic trust-region, which is theoretically equivalent to other quadratic methods but novel for continuous problems.

In a quantitative case-study, all stabilization methods can significantly reduce computation time compared to the standard algorithm. However, the quadratic trust-region and the linear box-step method have practical advantages. Firstly, these methods have proven to be robust because their performance is less sensitive to the numeric configuration. As a result, they require less fine-tuning to a specific problem to achieve good results. Second, the practical implementation of these methods is more straightforward.

The developed algorithm can also benefit from inexact cuts and valid inequalities, but there are also adverse effects and no clear trend. On the other hand, parallelization consistently improves performance by a factor corresponding to the number of SPs divided by two, which translates to a parallelization efficiency of 50%. Overall, the refined algorithm accelerates the vanilla BD algorithm by a factor of 100. Thanks to parallelization, its computation time remains constant when the number of scenarios increases. Cherry-picked setups can already outperform off-the-shelves solvers for the deterministic equivalent for

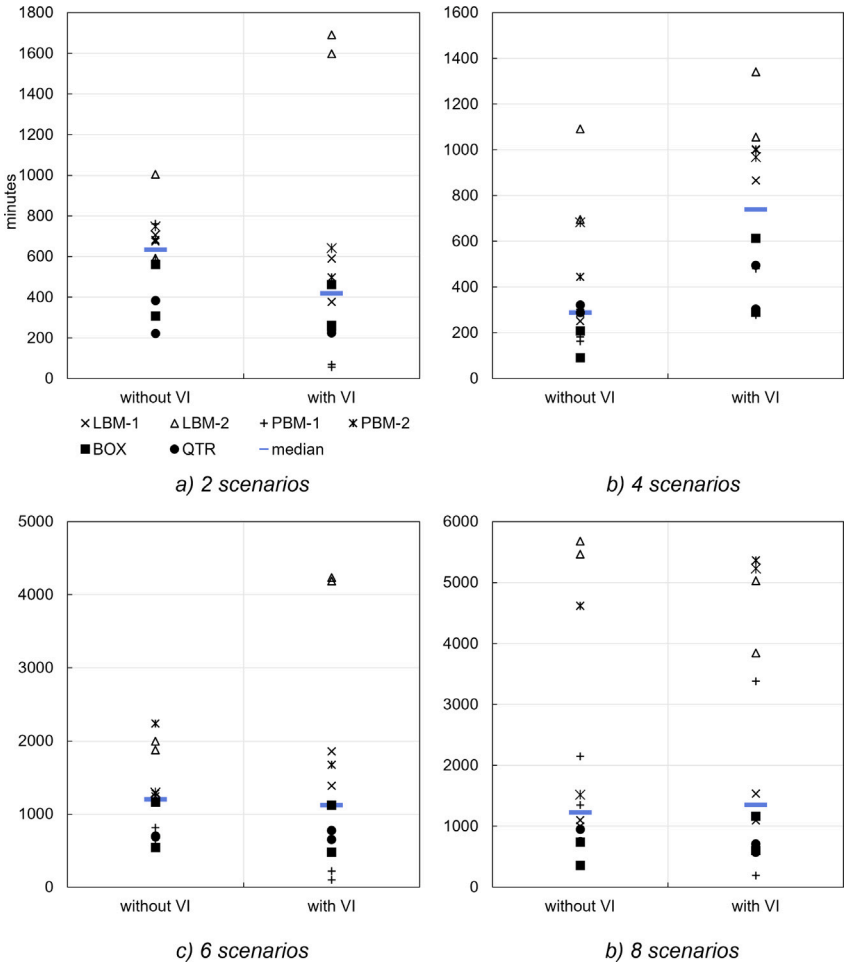

Fig. 10. Impact of valid inequalities.

two scenarios; more general algorithm setups start to achieve this around six scenarios, but note that performance can vary for other model setups beyond the range tested in this paper.

There are further conceivable improvements to the algorithm described in this paper. To list them, we again follow the categorization of enhancements introduced in Rahmaniani et al. (2017):

- **Solution procedure**: Considering most of the run-time is spent on the SPs, solving approximations instead of the original SPs can further increase performance. Not solving SPs to optimality in the current algorithm already exploits that valid cuts do not require exact solutions. Conceivable approximation methods include surrogate models based on machine learning, geometric interpolation, or using a reduced temporal resolution (Göke & Kendziorski, 2021; Mazzi, Grothey, McKinnon, & Sugishita, 2021; Neumann & Brown, 2021). However, approximations face two challenges: First, the high dimensionality of inputs to the SP corresponds to the number of complicating capacity variables. Second, using asymptotically inexact oracles also requires a revision of the stabilization method. The use of on-demand accuracy oracles appears promising. Introducing a solver component that allows to skip iterations and replace by them with approximations if they do no meet a predefined descent target is a natural extension of the current algorithm (van Ackooij & de Oliveira, 2015).
- **Solution generation**: Although heuristic methods are insufficient for adequate planning, they can greatly narrow the solution space of the MP to improve convergence. This approach also offers synergies with the selection of representative climatic years for the planning problem in the first place. For instance, solving the deterministic problem for specific climatic years, as performed in Section 9.1, is not only useful for clustering climatic years, but can also derive lower and upper bounds for capacity variables or system costs.
- **Cut generation**: The disproportional impact storage technologies have on the run-time that was discussed at the end of Section 6.1 suggests the algorithm would benefit from advanced cut generation. However, the size of the SPs still poses an obstacle to this approach. Therefore, refined approaches that only solve a modified but not the original SP are most promising (Papadakos, 2008b; Sherali & Lunday, 2011).

Apart from further enhancements, the outlined algorithm enables robust planning of renewable macro-energy systems wit a large number of climatic years, which was the original motivation for this work. How many climatic years to include and how to select these years for adequate system planning is an open question for future research. In addition, future research should extend the optimization under uncertainty to the operational stage in the SPs. Currently, uncertainty is limited to expansion in the MP, and operation in each scenario assumes perfect foresight, but stochastic dual dynamic programming could capture uncertainty at the operational stage as well (Papavasiliou, Mou, Cambier, & Scieur, 2018).

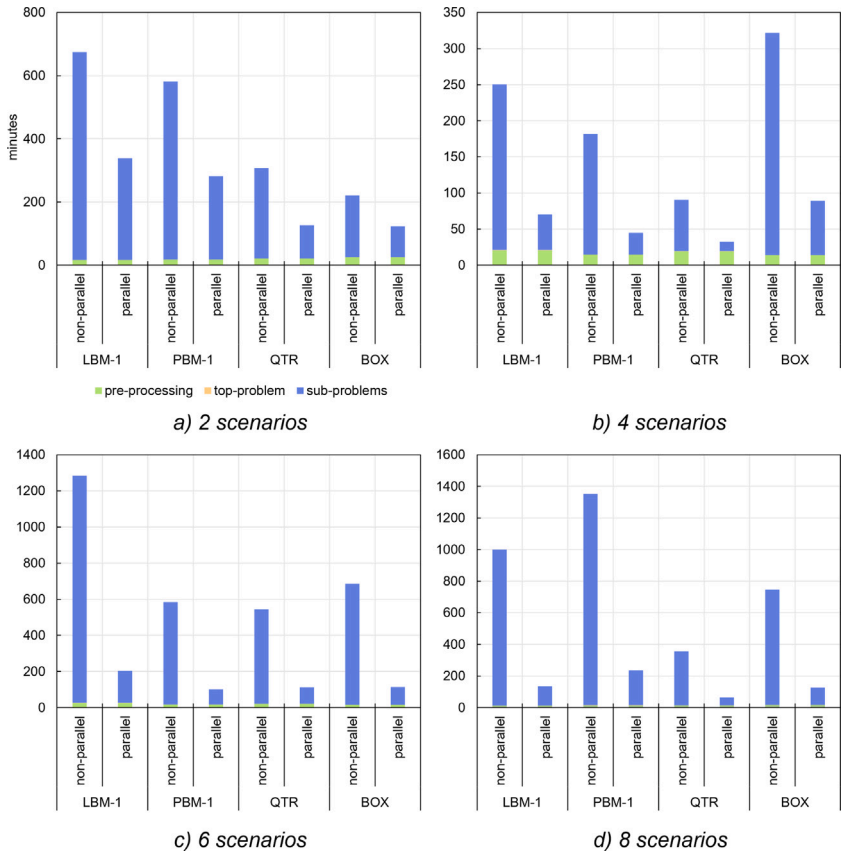

Fig. 11. Impact of parallelization.

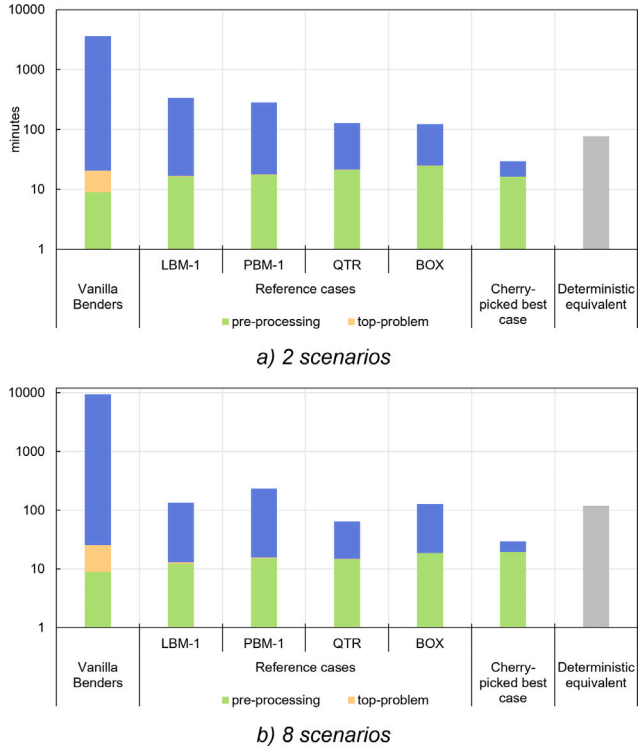

Fig. 12. Overall benchmark of refined algorithm.

Beyond climate uncertainty, the algorithm has the potential to make a broad range of other analyses in energy planning computationally tractable. In contrast to the deterministic equivalent, the algorithm scales well when adding complexity to the expansion problem since its computation time is negligible in the decomposed algorithm. For instance, the algorithm is suited for planning with endogenous learning, which adds mixed-integer variables to the expansion problem for a piecewise linear approximation of cost curves (Felling, Levers, & Fortenbacher, 2022; Zeyen, Victoria, & Brown, 2023). Similarly, the algorithm makes a stochastic representation of investment costs or discrete investment decisions in highly resolved models viable. Furthermore, it can efficiently compute near-optimal solutions due to its iterative nature and since oracles remain valid when changing the objective function.

Acknowledgments

The research leading to these results has received funding from the Federal Ministry for Economic Affairs and Climate Action, Germany via the project “MODEZEEN” (grant number FKZ 03EI1019D). A special thanks goes to all Julia developers. In addition, we would like to express our gratitude to the reviewers for their insightful feedback on earlier drafts of this paper.

Appendix A. Supplementary data

The data and execution scripts for the case-study are available here: https://github.com/leonardgoeke/EuSysMod/releases/tag/benderspaper.

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.ejor.2024.01.016.

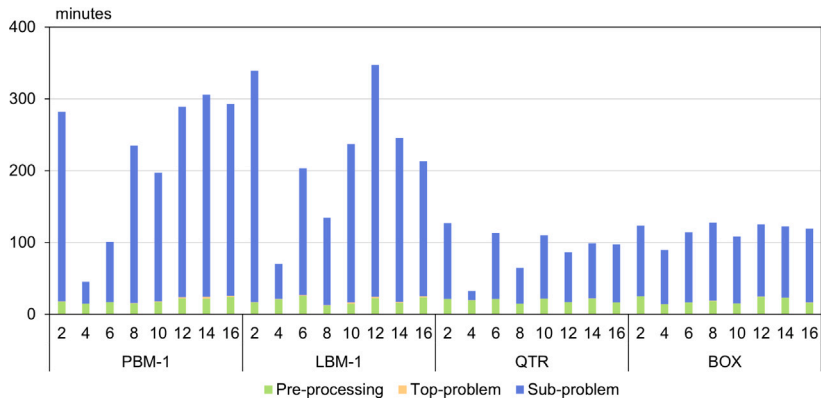

Fig. 13. Out-of-sample benchmark of reference cases.

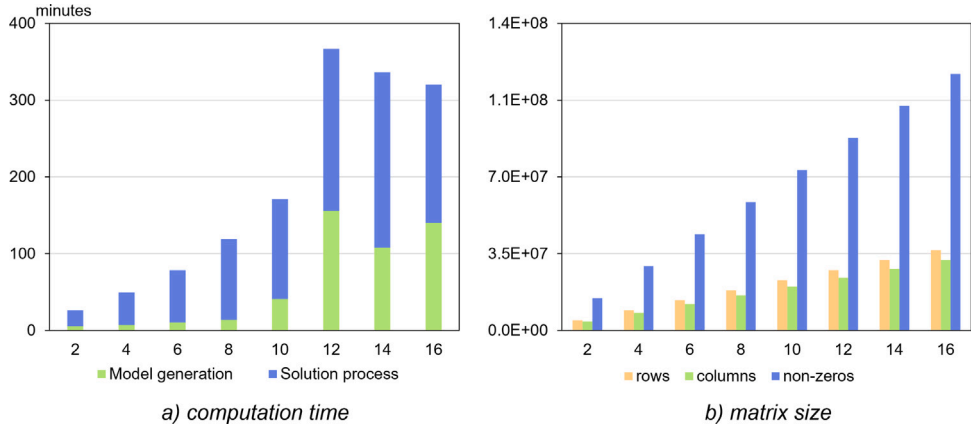

Fig. 14. Performance and size of the deterministic equivalent.

References

Backe, S., Skar, C., del Granado, P. C., Turgut, O., & Tomasgard, A. (2022). EMPIRE: An open-source model based on multi-horizon programming for energy transition analyses. *SoftwareX*, *17*, Article 100877. http://dx.doi.org/10.1016/j.softx.2021.100877.

Benders, J. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, *4*, 238–252.

Bloomfield, H. C., Brayshaw, D. J., Shaffrey, L. C., Coker, P. J., & Thornton, H. E. (2016). Quantifying the increasing sensitivity of power systems to climate variability. *Environmental Research Letters*, *11*(12), Article 124025. http://dx.doi.org/10.1088/1748-9326/11/12/124025.

Bonnans, J. F., Gilbert, J. C., Lemaréchal, C., & Sagastizábal, C. A. (2006). *Numerical optimization: Theoretical and practical aspects* (2nd ed.). (pp. 13–161). Springer-Verlag.

Brandenberg, R., & Stursberg, P. (2021). Refined cut selection for benders decomposition: applied to network capacity expansion problems. *Mathematical Methods of Operations Research*, *94*(3), 383–412. http://dx.doi.org/10.1007/s00186-021-00756-.

Brännlund, U., Kiwiel, K. C., & Lindberg, P. (1995). A descent proximal level bundle method for convex nondifferentiable optimization. *Operations Research Letters*, *17*(3), 121–126. http://dx.doi.org/10.1016/0167-6377(94)00056-C.

Conejo, A., Castillo, E., Minguez, R., & Garcia-Bertrand, R. (2006). *Decomposition techniques in mathematical programming: Engineering and science applications*. Springer Berlin Heidelberg.

Cordeau, J. F., Pasin, F., & Solomon, M. M. (2006). An integrated model for logistics network design. *Annals of Operations Research*, *144*, 59–82. http://dx.doi.org/10.1007/s10479-006-0001-3.

Craig, M. T., Wohland, J., Stoop, L. P., Kies, A., Pickering, B., Bloomfield, H. C., et al. (2022). Overcoming the disconnect between energy system and climate modeling. *Joule*, *6*, 1405–1417. http://dx.doi.org/10.1016/j.joule.2022.05.010.

Creutzig, F., Agoston, P., Goldschmidt, J. C., Luderer, G., Nemet, G. F., & Pietzcker, R. C. (2017). The underestimated potential of solar energy to mitigate climate change. *Nature Energy*, *2*, 17140.

de Oliveira, W., & Sagastizábal, C. (2014). Level bundle methods for oracles with on-demand accuracy. *Optimization Methods & Software*, *29*(6), 1180–1209. http://dx.doi.org/10.1080/10556788.2013.871282.

de Oliveira, W., Sagastizábal, C., & Lemaréchal, C. (2014). Convex proximal bundle methods in depth: a unified analysis for inexact oracles. *Mathematical Programming*, *148*(1), 241–277. http://dx.doi.org/10.1007/s10107-014-0809-6.

de Oliveira, W., & Solodov, M. (2016). A doubly stabilized bundle method for nonsmooth convex optimization. *Mathematical Programming*, *156*, 125–159. http://dx.doi.org/10.1007/s10107-015-0873-6.

de Oliviera, W., & Solodov, M. (2020). Bundle methods for inexact data. In A. M. Bagirov, M. Gaudioso, N. Karmitsa, M. M. Mäkelä, & S. Taheri (Eds.), *Numerical nonsmooth optimization: State of the art algorithms* (pp. 61–116). Springer International Publishing, http://dx.doi.org/10.1007/978-3-030-34910-3_3.

Doss-Gollin, J., Amonkar, Y., Schmeltzer, K., & Cohan, D. (2023). Improving the representation of climate risks in long-term electricity systems planning: a critical review. *Current Sustainable/Renewable Energy Reports*, http://dx.doi.org/10.1007/s40518-023-00224-3.

Fábián, C. I. (2013). *Computational aspects of risk-averse optimizationin two-stage stochastic models*. Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät II, Institut für Mathematik, http://dx.doi.org/10.18452/8433.

Felling, T., Levers, O., & Fortenbacher, P. (2022). Multi-horizon planning of multi-energy systems. *Electric Power Systems Research*, *212*, Article 108509. http://dx.doi.org/10.1016/j.epsr.2022.108509.

Frangioni, A. (2002). Generalized bundle methods. *SIAM Journal on Optimization*, *13*(1), 117–156. http://dx.doi.org/10.1137/S1052623498342186.

Frangioni, A. (2020). Standard bundle methods: Untrusted models and duality. In A. M. Bagirov, M. Gaudioso, N. Karmitsa, M. M. Mäkelä, & S. Taheri (Eds.), *Numerical nonsmooth optimization: State of the art algorithms* (pp. 61–116). Springer International Publishing, http://dx.doi.org/10.1007/978-3-030-34910-3_3.

Frangioni, A., & Gorgone, E. (2014). Bundle methods for sum-functions with "easy" components: applications to multicommodity network design. *Mathematical Programming*, *145*, 133–161. http://dx.doi.org/10.1007/s10107-013-0642-3.

Göke, L. (2021a). AnyMOD.jl: A julia package for creating energy system models. *SoftwareX*, *16*, Article 100871. http://dx.doi.org/10.1016/j.softx.2021.100871.

Göke, L. (2021b). A graph-based formulation for modeling macro-energy systems. *Applied Energy*, *301*, Article 117377. http://dx.doi.org/10.1016/j.apenergy.2021.117377.

Göke, L., & Kendziorski, M. (2021). The adequacy of time-series reduction for renewable energy systems. *Energy*, *238*, Article 121701. http://dx.doi.org/10.1016/j.energy.2021.121701.

Göke, L., Weibezahn, J., & von Hirschhausen, C. (2023). A collective blueprint, not a crystal ball: How expectations and participation shape long-term energy scenarios. *Energy Research & Social Science*, *97*, Article 102957. http://dx.doi.org/10.1016/j.erss.2023.102957.

Grochowicz, A., van Greevenbroek, K., Benth, F. E., & Zeyringer, M. (2023). Intersecting near-optimal spaces: European power systems with more resilience to weather variability. *Energy Economics*, *118*, Article 106496. http://dx.doi.org/10.1016/j.eneco.2022.106496.

Hilbers, A. P., Brayshaw, D. J., & Gandy, A. (2019). Importance subsampling: improving power system planning under climate-based uncertainty. *Applied Energy*, *251*, Article 113114. http://dx.doi.org/10.1016/j.apenergy.2019.04.110.

Hiriart-Urruty, J.-B., & Lemaréchal, C. (1993). In M. Artin, S. S. Chern, J. Coates, J. M. Fröhlich, H. Hironaka, F. Hirzebruch, & et al. (Eds.), *Grundlehren der mathematischen Wissenschaften: vol. 306*, *Convex analysis and minimization algorithms II: Advanced theory and bundle methods*. Berlin, Heidelberg: Springer Berlin Heidelberg, http://dx.doi.org/10.1007/978-3-662-06409-2.

Jacobson, A., Pecci, F., Sepulveda, N., Xu, Q., & Jenkins, J. (2023). A computationally efficient benders decomposition for energy systems planning problems with detailed operations and time-coupling constraints. *INFORMS Journal on Optimization*, http://dx.doi.org/10.1287/ijoo.2023.0005.

Kelley, J. E. (1960). The cutting-plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics*, *8*(4), 703–712. http://dx.doi.org/10.1137/0108053.

Kiwiel, K. C. (1990). Proximity control in bundle methods for convex nondifferentiable minimization. *Mathematical Programming*, *46*, 105–122. http://dx.doi.org/10.1007/BF01585731.

Lemaréchal, C. (1977). Nonsmooth optimization and descent methods.

Lemaréchal, C., Nemirovskii, A., & Nesterov, Y. (1995). New variants of bundle methods. *Mathematical Programming*, *69*, 111–147. http://dx.doi.org/10.1007/BF01585555.

Lemaréchal, C., & Sagastizábal, C. (1997). Variable metric bundle methods: From conceptual to implementable forms. *Mathematical Programming*, *76*(3), 393–410. http://dx.doi.org/10.1007/BF02614390.

Lemaréchal, C., Strodiot, J. J., & Bihain, A. (1981). On a bundle algorithm for nonsmooth optimization. *Nonlinear Programming*, *4*, 245–282. http://dx.doi.org/10.1016/B978-0-12-468662-5.50015-X.

Levin, T., Bistline, J., Sioshansi, R., Cole, W. J., Kwon, J., Burger, S. P., et al. (2023). Energy storage solutions to decarbonize electricity through enhanced capacity expansion modelling. *Nature Energy*, http://dx.doi.org/10.1038/s41560-023-01340-6.

Linderoth, J., & Wright, S. (2003). Decomposition algorithms for stochastic programming on a computational grid. *Computational Optimization and Applications*, *24*(2), 207–250. http://dx.doi.org/10.1023/A:1021858008222.

Lohmann, T., & Rebennack, S. (2017). Tailored benders decomposition for a long-term power expansion model with short-term demand response. *Management Science*, *63*(6), 2027–2048. http://dx.doi.org/10.1287/mnsc.2015.2420.

Magnanti, T. L., & Wong, R. T. (1981). Accelerating benders decomposition: Algorithmic enhancement and model selection criteria. *Operations Research*, *29*(3), 464–484. http://dx.doi.org/10.1287/opre.29.3.464.

Marsten, R. E., Hogan, W. W., & Blankenship, J. W. (1975). The boxstep method for large-scale optimization. *Operations Research*, *23*(3), 389–601. http://dx.doi.org/10.1287/opre.23.3.389.

Mazzi, N., Grothey, A., McKinnon, K., & Sugishita, N. (2021). Benders decomposition with adaptive oracles for large scale optimization. *Mathematical Programming Computation*, *13*(1), 683–703. http://dx.doi.org/10.1007/s12532-020-00197-0.

Neumann, F., & Brown, T. (2021). Broad ranges of investment configurations for renewable power systems, robust to cost uncertainty and near-optimality. http://dx.doi.org/10.48550/ARXIV.2111.14443.

Ohlendorf, N., & Schill, W.-P. (2020). Frequency and duration of low-wind-power events in Germany. *Environmental Research Letters*, *15*(8), Article 084045. http://dx.doi.org/10.1088/1748-9326/ab91e9.

Oliveira, W., Sagastizábal, C., & Scheimberg, S. (2011). Inexact bundle methods for two-stage stochastic programming. *SIAM Journal on Optimization*, *21*(2), 517–544. http://dx.doi.org/10.1137/100808289.

Papadakos, N. (2008b). Practical enhancements to the Magnanti–Wong method. *Operations Research Letters*, *36*(4), 444–449. http://dx.doi.org/10.1016/j.orl.2008.01.005.

Papavasiliou, A., Mou, Y., Cambier, L., & Scieur, D. (2018). Application of stochastic dual dynamic programming to the real-time dispatch of storage under renewable supply uncertainty. *IEEE Transactions on Sustainable Energy*, *9*(2), 547–558. http://dx.doi.org/10.1109/TSTE.2017.2748463.

Pessoa, A., Sadykov, R., Uchoa, E., & Vanderbeck, F. (2018). Automation and combination of linear-Programming based stabilization techniques in column generation. *INFORMS Journal on Computing*, *30*(2), 339–360. http://dx.doi.org/10.1287/ijoc.2017.0784.

Pfenninger, S. (2017). Dealing with multiple decades of hourly wind and PV time series in energy models: A comparison of methods to reduce time resolution and the planning implications of inter-annual variability. *Applied Energy*, *197*, 1–13. http://dx.doi.org/10.1016/j.apenergy.2017.03.051.

Pfenninger, S., & Staffell, I. (2016). Long-term patterns of European PV output using 30 years of validated hourly reanalysis and satellite data. *Energy*, *114*, 1251–1265. http://dx.doi.org/10.1016/j.energy.2016.08.060.

Plaga, L. S., & Bertsch, V. (2023). Methods for assessing climate uncertainty in energy system models — A systematic literature review. *Applied Energy*, *331*, Article 120384. http://dx.doi.org/10.1016/j.apenergy.2022.120384.

Rahmaniani, R., Crainic, T. G., Gendreau, M., & Rei, W. (2017). The Benders decomposition algorithm: A literature review. *European Journal of Operational Research*, *259*(3), 801–817. http://dx.doi.org/10.1016/j.ejor.2016.12.005.

Ritchie, H., & Roser, M. (2020). Energy. *Our World in Data*, URL https://ourworldindata.org/energy.

Ruhnau, O., & Qvist, S. (2022). Storage requirements in a 100% renewable electricity system: extreme events and inter-annual variability. *Environmental Research Letters*, *17*(4), Article 044018. http://dx.doi.org/10.1088/1748-9326/ac4dc8.

Ruszczyński, A. (1986). A regularized decomposition method for minimizing a sum of polyhedral functions. *Mathematical Programming*, *35*, 309–333. http://dx.doi.org/10.1007/BF01580883.

Ruszczyński, A. (2003). Decomposition methods. In *Handbooks in operations research and management science*: *vol. 10*, *Stochastic programming* (pp. 141–211). http://dx.doi.org/10.1016/S0927-0507(03)10003-5.

Shapiro, A., Dentcheva, D., & Ruszczyński, A. (2009). *Lectures on stochastic programming: Modeling and theory*. Society for Industrial and Applied Mathematics, http://dx.doi.org/10.1137/1.9780898718751.

Sherali, H., & Lunday, B. (2011). On generating maximal nondominated benders cuts. *Annals of Operations Research*, *210*, 57–72. http://dx.doi.org/10.1007/s10479-011-0883-6.

Skar, C., Doorman, G., & Tomasgard, A. (2014). Large-scale power system planning using enhanced benders decomposition. In *2014 Power systems computation conference* (pp. 1–7). http://dx.doi.org/10.1109/PSCC.2014.7038297.

Teichgraeber, H., Küpper, L. E., & Brandt, A. (2021). Designing reliable future energy systems by iteratively including extreme periods in time-series aggregation. *Applied Energy*, *304*, Article 117696. http://dx.doi.org/10.1016/j.apenergy.2021.117696.

van Ackooij, w., & de Oliveira, W. (2015). Level bundle methods for constrained convex optimization with various oracles. *Computational Optimization and Applications volume*, *57*, 555–597. http://dx.doi.org/10.1007/s10589-013-9610-3.

van Ackooij, W., & Frangioni, A. (2018). Incremental Bundle Methods using Upper Models. *SIAM Journal on Optimization*, *28*(1), 379–410. http://dx.doi.org/10.1137/16M1089897.

Van Slyke, R. M., & Wets, R. (1969). L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, *17*(4), 638–663. http://dx.doi.org/10.1137/0117061.

Wang, J., Wang, J., Liu, C., & Ruiz, J. P. (2013). Stochastic unit commitment with sub-hourly dispatch constraints. *Applied Energy*, *105*, 418–422. http://dx.doi.org/10.1016/j.apenergy.2013.01.008.

Wolf, C., Fábián, C. I., Koberstein, A., & Suhl, L. (2014). Applying oracles of on-demand accuracy in two-stage stochastic programming – A computational study. *European Journal of Operational Research*, *239*(2), 437–448. http://dx.doi.org/10.1016/j.ejor.2014.05.010.

Zakeri, G., Philpott, A. B., & Ryan, D. M. (2000). Inexact cuts in benders decomposition. *SIAM Journal on Optimization*, *10*(3), 643–657. http://dx.doi.org/10.1137/S1052623497318700.

Zeyen, E., Victoria, M., & Brown, T. (2023). Endogenous learning for green hydrogen in a sector-coupled energy model for Europe. *Nature Communications*, *14*, 3743. http://dx.doi.org/10.1038/s41467-023-39397-21.

Zverovich, V., Fábián, C. I., Eldon, E. F. D., & Mitra, G. (2012). A computational study of a solver system for processing two-stage stochastic LPs with enhanced benders decomposition. *Mathematical Programming Computation volume*, *4*, 211–238. http://dx.doi.org/10.1007/s12532-012-0038-z.