Optimization Letters (2022) 16:1373–1391
https://doi.org/10.1007/s11590-021-01826-w

ORIGINAL PAPER

Budget-cut: introduction to a budget based cutting-plane algorithm for capacity expansion models

Bismark Singh[1,2] · Oliver Rehberg[2] · Theresa Groß[3] · Maximilian Hoffmann[3] · Leander Kotzur[3] · Detlef Stolten[3,4]

Received: 10 March 2021 / Accepted: 3 November 2021 / Published online: 29 November 2021

Abstract
We present an algorithm to solve capacity extension problems that frequently occur in energy system optimization models. Such models describe a system where certain components can be installed to reduce future costs and achieve carbon reduction goals; however, the choice of these components requires the solution of a computationally expensive combinatorial problem. In our proposed algorithm, we solve a sequence of linear programs that serve to tighten a budget—the maximum amount we are willing to spend towards reducing overall costs. Our proposal finds application in the general setting where optional investment decisions provide an enhanced portfolio over the original setting that maintains feasibility. We present computational results on two model classes, and demonstrate computational savings up to 96% on certain instances.

1 Introduction

1.1 Background

Governments worldwide are pushing towards an increasing use of renewable energy technologies. In line with emission targets set by the European Commission—as part of the 2050 Low Carbon Economy roadmap to reduce emissions to 80% below the levels

✉ Bismark Singh
bismark.singh@fau.de

1 Department of Data Science, Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen 91058, Germany

2 Department of Mathematics, Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen 91058, Germany

3 Forschungszentrum Jülich GmbH, Institute of Techno-economic Systems Analysis (IEK-3), Wilhelm-Johnen-Str., Jülich 52428, Germany

4 Chair for Fuel Cells, c/o Institute of Techno-economic Systems Analysis (IEK-3) , Forschungszentrum Jülich GmbH, RWTH Aachen University, Wilhelm-Johnen-Str., Jülich 52428, Germany

in the year 1990 [6]—the German federal government plans to increase the percentage of energy derived from renewable sources to 80% by the year 2050 [3]. This rampant increase in the share of renewables requires important investment decisions that guide future energy policies, e.g., the expansion of existing energy capacity infrastructure to accommodate the needs and demands of future energy production and supply. Expanding existing capacity to include photovoltaics, hybrid generation systems, and storage devices are a few examples of such investment decisions that can potentially lead to lower greenhouse gas emissions [18].

Optimization models have a rich history for both operations of installed energy systems [1,33], as well as extending existing infrastructure; see, e.g., so-called capacity expansion models [2,25], models for integrating renewable sources with existing fossil fuels [29], and planning for expanding transmission networks [24]. Typically, mixed-integer programs (MIPs) are developed and employed to inform these decisions as well as optimize operations. Two examples of such energy system models are MARKAL [22] and TIMES [23]; see, also [20] and references within. The Framework for Integrated Energy System Assessment (FINE) is an open source python package that provides a framework for the modeling, optimization and analysis of energy systems [7,31]. The goal of such optimization models is to minimize the *total annual costs* for deploying and operating energy supply infrastructure, subject to the technical and operational constraints of a multi-commodity flow energy-system problem. FINE provides the functionalities to set up energy system models with multiple regions, commodities and time steps. Examples of commodities include electricity, hydrogen, and methane gas. Time steps can also be aggregated to reduce the complexity of the model [12]. In addition to existing infrastructural costs, costs are also incurred by building new components and increasing their capacities. The work in this article arises from a collaboration between mathematicians and energy-and-climate researchers at the Friedrich-Alexander-Universität Erlangen-Nürnberg and the Forschungszentrum Jülich, with the goal to improve the performance of the FINE package as well as other energy system models.

Against the backdrop of decreasing prices of renewable energy sources [13], rising CO_2 emission costs [5,6], a transforming energy demand and new options for energy storage and conversion [32], the consideration of novel technologies offers opportunities for further cost reduction of existing energy systems. We consider the problem of determining which components of an energy capacity infrastructure to install such that total annual costs are further reduced. Installing a new component incurs a fixed cost; however, overall costs are potentially reduced by allowing new components access to larger ranges of energy supplies thereby leading to a more efficient utilization of the entire system. Neumann and Brown consider a similar problem to expand transmission while minimizing total annual costs [27]. Such problems also find application in several other contexts within energy systems that seek to minimize total annual costs; see, e.g., [30] for a model that extends natural gas transmission networks. For an overview of transmission expansion planning problems, see, Mahdavi et al. [26].

1.2 Challenges

Such investment optimization problems are frequently formulated as MIPs that employ binary variables to inform "yes" or "no" decisions for utilizing new technologies; see, e.g., [9,20]. The choice of such discrete decision variables governing whether a new technology "is-built" leads to a significant increase in the computational effort to solve the MIP. Models covering multiple spatial regions impose further computational challenges as additional binary variables are required for each region. For an introduction to the challenges and model simplifications, see, e.g., [8,20]. However, as we mention in Sect. 1.1, integrating novel technologies into existing energy systems leads to potentially reduced overall costs and is also advantageous in keeping abreast with the dynamically changing energy sector. To this end, we provide a method that offers scope for reduced runtimes, both for systems with several existing technologies as well as new technologies to choose from.

This work is related to previous works on a so-called "district model" that includes six multi-family houses and households in each building [15], and also to a model with 16 transmission zones within Great Britain [28]. In previous improvements to the FINE package, Kannengießer et al. use a time-series aggregation method and present a temporally aggregated simplification of the model [15]. The authors use a multi-regional district model and fix the binary design decisions for all components. The corresponding optimization model is thus reduced to a linear program (LP); this model is solved easily, however the solutions are suboptimal. In contrast the approach we present is an iterative heuristic that is guaranteed to converge to the optimal solution.

2 Mathematical modeling

2.1 Notation

Indices/ Sets

$c \in \mathcal{C}$ Set of components $[c_1, ..., c_{|\mathcal{C}|}]$
$l \in \mathcal{L}$ Set of locations $[l_1, \ldots, l_{|\mathcal{L}|}]$

Parameters

$\text{CapMin}_{c,l}$	Minimum capacity for component c at location l [kW]
$\text{CapMin}_{c,(l_1,l_2)}$	Minimum capacity for component c on edge (l_1, l_2) [kW]
$\text{CapMax}_{c,l}$	Maximum capacity for component c at location l [kW]
$\text{CapMax}_{c,(l_1,l_2)}$	Maximum capacity for component c on edge (l_1, l_2) [kW]
$\text{TAC}^{cap}_{c,l}$	Total annual cost for installing one unit of capacity for component c at location location l [€/kW]
$\text{TAC}^{bin}_{c,l}$	Total annual cost for building component c at location l; independent of the size of the installed capacity [€]
$\text{TAC}^{op}_{c,l}$	Total annual cost for one unit of operation of component c at location l [€/kWh]
$\text{TAC}^{cap}_{c,(l_1,l_2)}$	Total annual cost for installing one unit of capacity for component c on edge (l_1, l_2) [€/kW]

$\text{TAC}^{bin}_{c,(l_1,l_2)}$	Total annual cost for building component c on edge (l_1, l_2); independent of the size of the installed capacity [€]
$\text{TAC}^{op}_{c,(l_1,l_2)}$	Total annual cost for one unit of operation for component c on edge (l_1, l_2) [€/kWh]

Decision Variables

$bin_{c,l}$	(IsBuilt-variable) 1 if component c is built at location l; else 0
$bin_{c,(l_1,l_2)}$	(IsBuilt-variable) 1 if component c is built on edge (l_1, l_2); else 0
$cap_{c,l}$	Installed unit capacity of component c at location l [kW]
$cap_{c,(l_1,l_2)}$	Installed unit capacity of component c on edge (l_1, l_2) [kW]
$op_{c,l}$	Used unit capacity of component c at location l [kWh]
$op_{c,(l_1,l_2)}$	Used unit capacity of component c on edge (l_1, l_2) [kWh]

In the above notation, bin denotes a *binary* decision variable, cap and op denote continuous decision variables for the *capacity* and *operation*, respectively. The parameters CapMin and CapMax denote bounds on the cap variable, while TAC denotes *total annual cost*. We provide details in Sect. 2.2.

2.2 Optimization model

The models build with the FINE package allow the inclusion of two types of components: (i) "optional" components that are modeled with additional investment costs independent of the installed unit size, and (ii) already existing components that are modeled with an installation cost contribution that is linearly dependent on the installed unit size. We have five choices for these optional and existing components that are relevant to the discussion in this work: Source, Sink, Storage, Conversion, and Transmission. The complete FINE package includes other components as well, and the user specifications decide the components that form part of the optimization model. We then have a graph where the nodes include combinations of the first four component types, while each Transmission component represents an edge connecting two nodes. Let the index $c \in \mathcal{C}$ denote the components, index $l \in L$ denote the locations of a node, and tuple (l_1, l_2) with $l_1, l_2 \in \mathcal{L}, l_1 \neq l_2$ denote the start and end locations for an edge, respectively.

Further, let $\mathcal{C}^N \subset \mathcal{C}$ denote the set of four components that form nodes, and $\mathcal{C}^E$ denote the set of components that form edges; i.e., $\mathcal{C}^N = \{\mathcal{C}^{\text{Source}}, \mathcal{C}^{\text{Sink}}, \mathcal{C}^{\text{Storage}}, \mathcal{C}^{\text{Conversion}}\}$ and $\mathcal{C}^E = \{\mathcal{C}^{\text{Transmission}}\}$. Here, the source components include nodes that provide commodities to the graph, while the sink components withdraw commodities from the graph. The storage components are nodes that connect the different time steps with each other by incorporating a so-called state of charge (SoC) variable; these components operate as a sink (that increases the SoC) or a source (that decreases the SoC) "state variable" (the SoC) at a given time step. The conversion components are nodes that connect multiple "commodity-subgraphs" to each other by converting one commodity to another; e.g., an electrolyzer consumes electricity and converts it into hydrogen. Finally, let $\mathcal{L}^c$ denote the location(s) of a component c. Although the decision variables and parameters within the FINE package include other indices (such as time and commodity) as well, we suppress these indices as they are not relevant to

the discussion in this article; see, the complete model description in the appendix, and further details in [31].

FINE includes binary variables to inform whether new optional components are built, and calls these binary variables as "IsBuilt-variables" [31]. A value of 1 denotes a component is built at a given location. If a component already exists, it does not have a IsBuilt-variable; we assume that there is at least one such component, else there is no configuration to start with. Building an optional component c at location l incurs a fixed total annual cost (TAC) of $\text{TAC}^{bin}_{c,l}$. For the sake of notation, we assume existing components also have associated binary variables that are always 1 with $\text{TAC}^{bin}_{c,l} = 0$. Further, the decision to build is governed by its corresponding capacity variable, $cap_{c,l}$. The value of a capacity variable corresponds to the scale of the installed component, e.g., for a photovoltaic Source component it corresponds to the area of solar panels installed [31]. If any capacity is installed, the corresponding IsBuilt-variable takes the value 1. All optional components have minimum capacity thresholds that are informed by data; we take this threshold as 0 if data is unspecified. In other words, $cap_{c,l}$ is a semi-continuous variable that is either 0 or lower bounded by the minimum capacity. Analogously, maximum capacity thresholds are available as well, and we take these as $+\infty$ if unspecified. Equations (1) and (2) summarize this discussion for the nodes and edges, respectively.

$$\forall c \in \mathcal{C}^E; l_1, l_2 \in \mathcal{L}^c, l_1 \neq l_2 :$$

$$\text{capMax}_{c,(l_1,l_2)} \cdot bin_{c,(l_1,l_2)} \geq cap_{c,(l_1,l_2)} \tag{1a}$$
$$\text{capMin}_{c,(l_1,l_2)} \cdot bin_{c,(l_1,l_2)} \leq cap_{c,(l_1,l_2)} \tag{1b}$$
$$cap_{c,(l_1,l_2)} \geq 0 \tag{1c}$$
$$bin_{c,(l_1,l_2)} \in \{0, 1\}. \tag{1d}$$

$$\forall c \in \mathcal{C}^N; l \in \mathcal{L}^c :$$

$$\text{capMax}_{c,l} \cdot bin_{c,l} \geq cap_{c,l} \tag{2a}$$
$$\text{capMin}_{c,l} \cdot bin_{c,l} \leq cap_{c,l} \tag{2b}$$
$$cap_{c,l} \geq 0 \tag{2c}$$
$$bin_{c,l} \in \{0, 1\}. \tag{2d}$$

If $bin_{c,(l_1,l_2)} = 1$, then Eqs. (1) enforce $\text{capMin}_{c,(l_1,l_2)} \leq cap_{c,(l_1,l_2)} \leq \text{capMax}_{c,(l_1,l_2)}$; else, $cap_{c,(l_1,l_2)} = 0$. Equations (2) are analogous to equations (1). Further, we have operational variables, that include time indices, corresponding to the components that we denote by $op_{c,l}$; these denote how much of the installed capacity is actually used.

Below is the optimization model we consider in this article.

$$z^* = \min \text{TAC}^{cap} \cdot cap + \text{TAC}^{bin} \cdot bin + \text{TAC}^{op} \cdot op \tag{3a}$$
$$\text{s.t. } (1), (2) \tag{3b}$$
$$\text{non-temporal bounding constraints} \tag{3c}$$
$$\text{temporal bounding constraints} \tag{3d}$$

component-linking constraints (3e)

transmission constraints (3f)

storage constraints. (3g)

The objective function in Eq. (3a) abbreviates the following quantity:

$$\sum_{c\in\mathcal{C}^N}\sum_{l\in\mathcal{L}^c}\left(\mathrm{TAC}^{cap}_{c,l}\cdot cap_{c,l}+\mathrm{TAC}^{bin}_{c,l}\cdot bin_{c,l}+\mathrm{TAC}^{op}_{c,l}\cdot op_{c,l}\right)+$$
$$\sum_{c\in\mathcal{C}^E}\sum_{\substack{(l_1,l_2)\in\mathcal{L}^c\times\mathcal{L}^c,\\ l_1\neq l_2}}$$
$$\left(\mathrm{TAC}^{cap}_{c,(l_1,l_2)}\cdot cap_{c,(l_1,l_2)}+\mathrm{TAC}^{bin}_{c,(l_1,l_2)}\cdot bin_{c,(l_1,l_2)}+\mathrm{TAC}^{op}_{c,(l_1,l_2)}\cdot op_{c,(l_1,l_2)}\right). \tag{4}$$

Here, cap, bin, and op denote three vectors corresponding to the capacity, IsBuilt, and operational variables, respectively. The coefficients — TAC^{cap}, TAC^{bin}, and TAC^{op} — denote vectors of appropriate size for the TAC corresponding to installing capacity, building new components, and operating these components, respectively. That is, the objective function includes cost contributions that scale with the size of the installed capacities of the components linearly (TAC^{cap}), cost contributions that are independent of the size of the installed capacity but occur if a component is built (TAC^{bin}), as well as cost contributions that are connected to the operation of the built components (TAC^{op}); see, the appendix for details. Then, the objective function seeks to minimize the sum of the TACs of the entire system, with both optional and existing components. The bounding constraints in Eq. (3c) and (3d) enforce further limits on the capacity and operation variables with and without time indices, respectively. The component-linking-constraints of Eq. (3e) include commodity balances, annual commodity inflow and outflow limits as well as shared potentials; they also serve to define limits via the capMax parameter.

The bidirectional and symmetric nature of the components along edges, as well as constraints that model the optimal power flow using the standard linearized DC formulation, is expressed via the transmission constraints of Eq. (3f). The storage constraints of Eq. (3g) express the charging and discharging status via the state of the charge for the different components. For a detailed description of these constraints, see [31]; we provide a complete model description in the appendix. Importantly, constraints (3c)–(3g) do not contain any binary variables.

3 A budget-cut algorithm

3.1 Background

MIP solvers typically rely on branch-and-bound strategies to identify the optimal solution. The presence of additional binary variables results in a larger search tree that

can create potential computational challenges. In this section, we propose an algorithm that prunes branches that lead to suboptimal solutions. To this end, we derive a sequence of cuts by solving a sequence of LPs. We use these cuts to determine a "budget" that provides a valid inequality for the original problem—that is now reduced in size. We formalize this concept in the Budget-Cut Algorithm, present it in Fig. 1, and explain it in more detail below. We first note that when constraints (1d) and (2d) are reduced to their continuous relaxation, model (3) is an LP. To this end, we solve two models where all the binary IsBuilt-variables are apriori determined. Consider the following optimization models:

$$\bar{z} = \min \ \mathrm{TAC}^{cap} \cdot cap + \mathrm{TAC}^{op} \cdot op \tag{5a}$$
$$\text{s.t.} \ \ (1a) - (1c), (2a) - (2c) \tag{5b}$$
$$(3c) - (3g) \tag{5c}$$
$$bin = 0, \tag{5d}$$

and

$$\underline{z} = \min \ \mathrm{TAC}^{cap} \cdot cap + \mathrm{TAC}^{op} \cdot op \tag{6a}$$
$$\text{s.t.} \ \ (1a), (1c), (2a), (2c) \tag{6b}$$
$$(3c) - (3g) \tag{6c}$$
$$bin = 1. \tag{6d}$$

We denote models (5) and (6) as `Existing` and `Extended` with optimal objective function values of $\bar{z}$ and $\underline{z}$, respectively. We assume `Existing` is feasible; in Sect. 5 we discuss the implications of this assumption. The `Existing` model determines a baseline where no optional components are built, and the only components used for the solution are those that that do not have costs independent of the size of the installed capacity; i.e., all the IsBuilt-variables are set to 0. Thus, the second term - $\mathrm{TAC}^{bin} \cdot bin$ - in the objective function of model (3) is 0 for all the IsBuilt variables. We note that the `Existing` model still includes the already existing components, and these are the only components we optimize over. The `Extended` model determines the other extreme where all the optional components are built; i.e., all the IsBuilt-variables are set to 1. The idea of the `Extended` model is subtle. Although all the IsBuilt-variables are set to 1, we ignore the cost to build them by not including the second term - $\mathrm{TAC}^{bin} \cdot bin$ - of the objective function of model (3). By ignoring the cost to build the components, the `Extended` model focuses on finding the most cost-effective components to use and install capacities on. Further, we do not include the lower thresholds on capacity for optional components in Eqs. (1c) and (2c). Since optional components are built to reduce total costs, the `Extended` model represents the best we can hope to achieve. Intuitively, the `Existing` model chooses an optimal solution from the existing capacity infrastructure, while the `Extended` model chooses the optimal solution from the existing and the optional capacity infrastructure. The following lemma relates the `Existing` and `Extended` models with the "true" model (3).

Algorithm 1 Budget-Cut Algorithm

Input: an instance of model (3); a_i ←objective function coefficient of element $i, \forall i \in I$; $TIME$
Output: z^*; optimality gap for best known feasible solution to model (3)

$iter \leftarrow 0$, search ← `True`
while time $\leq TIME$
  Solve model (5), get $\bar{z}$
  Solve model (6), get $\underline{z}$
  Update time to the cumulative wall-clock time
  $b \leftarrow \bar{z} - \underline{z}$
  if $b < min_{i \in I} a_i$
    $z^* \leftarrow \bar{z}$; Gap ← 0%; go to 23
  if $b > max_{i \in I} a_i$
    search ← `False`
  while search
    $J = \cup_{i:a_i > b}$; $I \leftarrow I \setminus J$
    if $J \neq \emptyset$
      $iter \leftarrow iter + 1$
      Solve model (6) with $bin_i \leftarrow 0, \forall i \in J$; update $\underline{z}$, Gap ← optimality gap
      $b \leftarrow \bar{z} - \underline{z}$
      if $b < min_{i \in I} a_i$
        $z^* \leftarrow \bar{z}$; go to 23
    else
      go to 22
  Update time to the cumulative wall-clock time
  Solve model (8); $z^* \leftarrow z^*$, Gap ← optimality gap
  Return: z^*, Gap

Lemma 1 $\underline{z} \leq z^* \leq \bar{z}$.

Proof The feasible region represented by constraints (5b)–(5d) is a subset of the feasible region represented by constraints (3b)–(3g). Further, the objective functions of models (3) and (5) are identical, since $\text{TAC}^{bin} \cdot bin$ is 0 for model (5). With a minimization objective, $z^* \leq \bar{z}$ follows. Next, we note that constraints (6b) and (6d) restrict the capacity variables between 0 and capMax; while, constraints (3b) enforce the capacity variables are either 0 or bounded between capMin and capMax. Since the other constraints of models (3) and (6) are identical, the feasible region of model (3) is smaller than that of model (6). Further, the objective function in Eq. (3a) is at least that in Eq. (6a). Thus, $\underline{z} \leq z^*$ follows. □

Let the triplets $[cap^*, bin^*, op^*]$, $[\overline{cap}, \overline{bin}, \overline{op}]$ and $[\underline{cap}, \underline{bin}, \underline{op}]$ denote the optimal solutions for models (3), (5) and (6), respectively. Here, $\overline{bin} = 0$ and $\underline{bin} = 1$. Further, we define $\bar{z} - \underline{z} = b \geq 0$. Next, we show that b determines a "budget" - the maximum cost we are willing to spend on building IsBuilt components. In other words, spending more than b exceeds the potential savings offered by the addition of optional components. We use the following corollary to Lemma 1 in the proof.

Corollary 1 $TAC^{cap} \cdot cap^* + TAC^{op} \cdot op^* \geq TAC^{cap} \cdot \underline{cap} + TAC^{op} \cdot \underline{op}$.

Proof From the proof of Lemma 1, it follows that the solution $(cap^*, op^*, 1)$ is feasible for model (6). □

Lemma 2 *$TAC^{bin} \cdot bin \leq b$ is a valid inequality for model (3).*

Proof From Lemma 1 and the definition of b we have, $\text{TAC}^{cap} \cdot cap^* + \text{TAC}^{op} \cdot op^* + \text{TAC}^{bin} \cdot bin^* - (\text{TAC}^{cap} \cdot \underline{cap} + \text{TAC}^{op} \cdot \underline{op}) \leq b$. Then, from Corollary 1, we have $\text{TAC}^{bin} bin^* \leq b$, for any optimal solution for model (5). □

For sake of completeness, we rewrite the valid inequality in its full form:

$$\sum_{c \in \mathcal{C}^N} \sum_{l \in \mathcal{L}^c} \text{TAC}^{bin}_{c,l} \cdot bin_{c,l} + \sum_{c \in \mathcal{C}^E} \sum_{\substack{(l_1,l_2) \in \mathcal{L}^c \times \mathcal{L}^c, \\ l_1 \neq l_2}} \text{TAC}^{bin}_{c,(l_1,l_2)} \cdot bin_{c,(l_1,l_2)} \leq b, \quad (7)$$

as well as model (3) with the valid inequality:

$$z^* = \min \text{TAC}^{cap} \cdot cap + \text{TAC}^{bin} \cdot bin + \text{TAC}^{op} \cdot op \quad (8a)$$

$$\text{s.t. } (3b) - (3g), \quad (8b)$$

$$(7). \quad (8c)$$

3.2 A budget-cut algorithm

Equation (7) provides a valid inequality for model (3) that takes the form of a typical 0–1 knapsack constraint. For ease of exposition within this section, we represent Eq. (7) as $\sum_{i \in I} a_i \cdot bin_i \leq b$. Although the weights of the knapsack items, a_i, are known, the benefit derived by the addition or removal of a single element requires the solution of a new instance of model (3). We could determine this benefit by solving an instance with a subset of IsBuilt-variables fixed to 1, and then compute the possible savings minus the construction costs. However, this requires a solution to $2^{|I|}$ problem instances. Instead, we use b to determine elements that are too expensive to construct, remove them apriori via Lemma 2, and re-solve model (3) with the valid inequality (i.e., model (8)). Small values of b provide tighter models. In this section, we seek to further reduce the size of the reduced model (8) by reducing b. We begin by distinguishing three cases.

1. Case 1: $b < \min_i a_i$ It follows from Lemma 2 that $bin_i = 0, \forall i \in I$; then, from Lemma 1 $\overline{z} = z^*$.
2. Case 2: $\min_i a_i \leq b \leq \max_i a_i$ We proceed by first fixing all binaries with $a_i > b$ to 0. Then, we recompute the budget; i.e., we solve `Extended` with these binaries set to 0. This guarantees the updated budget is no more than the previous budget, and we repeat this process.
3. Case 3: $\max_i a_i < b$ In this case, we cannot reduce the budget further. Further, if $\sum_{i \in I} a_i < b$ is true, Eq. (7) is redundant.

We reflect these three cases in the Budget-Cut Algorithm. The algorithm takes as input an instance of model (3), and a time limit, $TIME$. A key prerequisite of the algorithm is that model (5) is feasible, else the algorithm's step 3 fails. In Sect. 5, we provide a discussion on handling infeasible instances. The other assumption of the algorithm is that not all the components are optional. In the absence of this assumption, the initial configuration has no cost and thus there is nothing to do. The Budget-Cut

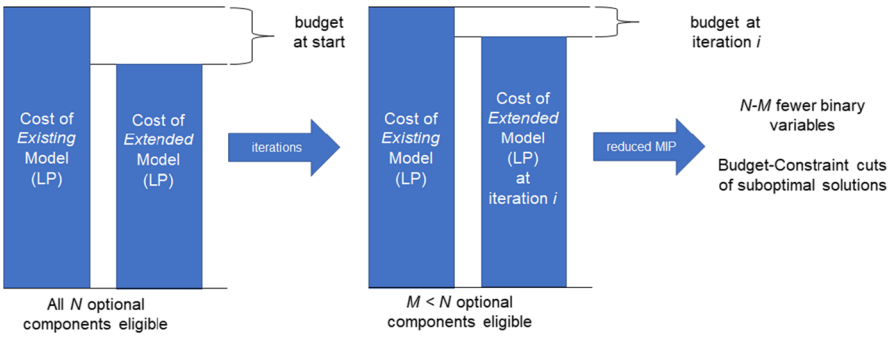

Fig. 1 Visualization of the budget calculation and update during the Budget-Cut Algorithm. Here, `Existing` and `Extended` are the restricted and relaxed versions of the original problem. Here N is the number of optional components, while M is an integer less than N. See, Sect. 3 for details

Algorithm solves two LPs in Steps 3 and 4, at most $|I|$ LPs in the loop around step 15, and finally a MIP in step 22. The algorithm outputs the optimal solution and objective function value for model (3), or the corresponding optimality gap and the best found feasible solution in the time limit. The Budget-Cut Algorithm provides at least three advantages compared to a naive solution method. First, from Lemma 2 we can directly proceed to Step 23; this happens if the algorithm recognizes that the budget does not allow building any optional elements. Second, if the algorithm does proceed to the while loop, we are guaranteed at least one element has a binary variable that is fixed to 0. This ensures a finite termination of the algorithm in at most $|I|$ iterations. Third, the solutions of the LPs serve as warmstarts for model (8). Figure 1 presents a visualization of the budget update. In the next section, we compare the computational performance of the algorithm with a naive solution method.

4 Computational results

4.1 Setup

To examine the computational performance of the Budget-Cut Algorithm, we conduct a number of computational experiments on different instances of model (3). The FINE package uses a time series aggregation method to reduce the size of the optimization model; i.e., it aggregates the complete considered time horizon of, e.g., 365 days into so-called "typical days". For details on the time series aggregation methods used within FINE, see [12]. We define a model instance with a time horizon of one year using typical days — 7, 14, 28, 56, and 112 — and weather years from 1995 to 2000. The weather year parameter does not affect the size of the optimization model, but determines which input data set of commodity power demands and supplies is used. However, models with a larger number of typical days result in larger optimization models.

All tests in this article are carried out with Pyomo 5.7.1 [11] using Gurobi 9.0.2 [10] on two machines: (i) an Intel Core i7 2.8 GHz processor with 16 GB of memory,

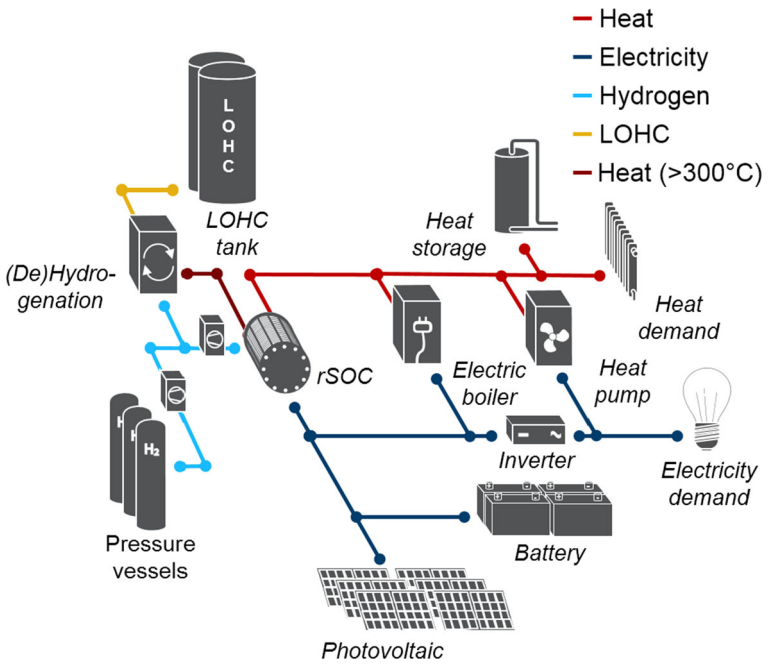

Fig. 2 Structure of the Self-sufficient building scenario of Sect. 4.1 from Figure 1 of [19]

and (ii) a node on the Jülich Research on Exascale Cluster Architectures (JURECA) supercomputer, with a cluster's batch partition 2x Intel Xeon E5-2680 v3 (Haswell) and a 2.5GHz processor and 128 GB of memory [14]. We refer to these two machines as Machine I and Machine II, respectively. We solve smaller models of up to 56 typical days on Machine I with the Gurobi `threads` parameter set to 3, and larger models with 112 typical days on Machine II with the `threads` parameter set to 32. We use $TIME = 900$ seconds and $TIME = 15,000$ seconds as time limit for our computational experiments on Machine I and Machine II, respectively.

We use the self-sufficient building scenario (SelfScen) of Kotzur et al. [19] for our experiments; see, also [16]. The SelfScen optimizes the cost of a single household building to construct and operate on its own, thereby being self-sufficient from an energy perspective. The available technologies include rooftop photovoltaic systems and batteries for short-term electricity storage, as well as reversible fuel cells and liquid organic hydrogen storage systems for long-term energy storage, to meet demand for power. There is also a demand for heat, this is fulfilled by a combination of electric boilers, heat pumps and heat storage. SelfScen includes the following commodities - heat, electricity, hydrogen, liquid organic hydrogen carrier (LOHC), and high temperature heat. Scenario instances include heat and electricity demand for the household, and the maximum power rate that can be generated by the photovoltaic units, for the weather years 1995-2000. To this end, the SelfScen chooses the technologies to install (i.e., bin), the capacities for each component (i.e., cap), and their operation (i.e., op).

Optional modeled components are the heat pump, the reversible fuel cells, and conversion technologies that are required for using the hydrogen storage components. See Fig. 2 for an illustration of the SelfScen, and [17] for the dataset associated with the SelfScen. In Online Appendix A.3 we provide additional computational experiments for another scenario that includes multiple buildings.

4.2 Computational experiments

In Table 1, we compare the computational results for the naive solution method and the Budget-Cut Algorithm. Within our time limit, the Budget-Cut Algorithm succeeds in finding the optimal solution in all the instances. The naive solution method, however, fails to find even a feasible solution for all instances with 56 typical days. For the instance with 112 typical days and the year 1995—that we solve on Machine II—the naive solution terminates with a large MIP gap of 45%. The value in the third column of Table 1 is the best known value of the objective function in model (3); the value reported by the algorithm is indeed the optimal in all instances. Next, we note that the naive solution method performs better for smaller instances up to 7 typical days. However, for the larger instances on Machine I the improvements are significant; for instances with 14, 28, and 56 typical days the average improvement is 61.2%, 88.0%, and 82.2%, respectively. For the largest instance with 112 typical days that we solve on Machine II, the runtimes are an order of magnitude lower except for the year 2000; the average improvement here is 81.4%.

All instances in Table 1 are solved without any iterations of the Budget-Cut Algorithm; i.e., the `while` loop in the Budget-Cut Algorithm is not entered. Then, the valid inequality in Eq. (7) is trivially true with $bin_i = 1, \forall i \in I$. In other words, no component is trivially excluded for having too high costs. However, even then the naive solution method is significantly slower as the algorithm benefits from warm-starts derived from the `Extended` model as well as additional cuts derived by the optimization solver with the addition of the trivially true valid inequality.

To this end, we modify the SelfScen to ensure $\sum_{i \in I} a_i \cdot bin_i > b$; thus, the algorithm enters the `while` loop. For details on how we modify the SelfScen, see Online Appendix A.2. We denote this modified scenario as ModSelfScen. Table 2 presents our computational results for the ModSelfScen, analogous to Table 1. The optimal objective function values for the ModSelfScen are larger than those of SelfScen due to the increased objective function coefficients of the ModSelfScen. All instances in Table 2 are solved to optimality within the time limit, thus we remove the two columns corresponding to "Gap" in Table 1. All instances require two iterations. Trends for the ModSelfScen mirror those of the SelfScen, however the percentage savings are smaller for the former. The average improvements for instances with 14, 28, 56, and 112 typical days are 25.6%, 53.2%, 61.2% and 68.4%, respectively.

In Online Appendix A.3 we provide an additional set of computational results on another class of scenarios. The results again follow the same trends as those we report above.

Table 1 Summary of computational experiments for the SelfScen scenario. All instances are solved in zero iterations for Algorithm 1

Instance		Objective function	Gap (%)		Time (seconds)		Improvement (%)
Typical days	Year	(€)	Naive	Algorithm 1	Naive	Algorithm 1	Algorithm 1
7	1995	4999.1			8.6	14.1	-64.4
	1996	4899.7			7.1	12.0	-70.2
	1997	5205.7			7.3	12.8	-76.7
	1998	5299.0			7.8	12.2	-56.8
	1999	5118.5			6.7	12.5	-86.7
	2000	4775.3			7.2	11.9	-65.1
14	1995	5092.7			45.0	17.4	61.4
	1996	5012.9			43.4	17.8	58.9
	1997	5131.1			40.7	17.2	57.8
	1998	5241.4			41.6	18.6	55.4
	1999	4660.6			58.4	18.1	69.0
	2000	5015.8			49.4	17.3	65.1
28	1995	4,986.5			535.0	44.7	91.7
	1996	5075.7			407.1	48.5	88.1
	1997	5106.6			291.5	42.0	85.6
	1998	5266.6			240.0	40.4	83.2
	1999	4724.1			573.6	45.2	92.1
	2000	5057.9			346.9	43.1	87.6

Table 1 continued

Instance		Objective function	Gap (%)		Time (seconds)		Improvement (%)
Typical days	Year	(€)	Naive	Algorithm 1	Naive	Algorithm 1	Algorithm 1
56	1995	4811.6	∞		**T**	152.8	83.0
	1996	5129.6	∞		**T**	131.8	85.4
	1997	5004.5	∞		**T**	151.4	83.2
	1998	5211.0	∞		**T**	146.7	83.7
	1999	4825.5	∞		**T**	194.0	78.4
	2000	4863.6	∞		**T**	185.5	79.4
112	1995	4816.9	45.0		**T**	554.2	96.3
	1996	5031.7			12,585.6	552.0	95.6
	1997	4879.4			13,385.1	495.1	96.3
	1998	5152.6			12,692.9	440.0	96.5
	1999	4818.8			10,657.8	606.0	94.3
	2000	4755.3			12,017.1	11,073.8	7.8

The "Gap" columns provide the MIP gap reported by Gurobi; a blank denotes a gap of 0%, while ∞ denotes no feasible solution is found. Entries marked with **T** in the "Time" column denote the maximum time limit is reached. The "Improvement" columns provide the percent improvement in run time for the algorithm compared to the naive solution method. All entries are rounded to the first decimal. For details, see Sect. 4

Table 2 Summary of computational experiments for the ModSelfScen scenario, analogous to that of Table 1

Instance		Objective function	Time (seconds)		Improvement (%)
Typical days	Year	(€)	Naive	Algorithm 1	the Budget-Cut Algorithm
7	1995	33,415.7	14.2	15.2	−7.2
	1996	35,618.9	10.1	14.3	−42.4
	1997	49,276.1	10.6	15.5	−47.0
	1998	21,413.3	12.2	15.6	−27.6
	1999	27,463.2	4.9	15.4	−214.6
	2000	16,437.5	5.5	14.7	−165.9
14	1995	33,425.0	35.3	25.9	26.5
	1996	23,349.1	36.3	25.3	30.2
	1997	33,807.9	36.3	24.8	31.6
	1998	23,552.9	30.5	28.2	7.7
	1999	21,286.8	36.3	26.6	26.7
	2000	32,146.4	38.9	26.9	30.7
28	1995	33,715.3	133.1	49.8	62.6
	1996	21,331.2	92.3	51.0	44.8
	1997	39,426.4	140.2	67.1	52.2
	1998	23,923.8	128.3	56.7	55.8
	1999	23,356.4	125.1	68.5	45.2
	2000	29,167.4	135.7	56.5	58.3

Table 2 continued

Instance		Objective function	Time (seconds)		Improvement (%)
Typical days	Year	(€)	Naive	Algorithm 1	the Budget-Cut Algorithm
56	1995	27,713.0	464.4	221.8	52.2
	1996	24,991.0	449.5	100.5	77.6
	1997	35,332.9	666.4	145.9	78.1
	1998	21,938.0	280.3	200.1	28.6
	1999	27,244.7	455.0	129.0	71.7
	2000	21,412.6	315.3	130.2	58.7
112	1995	29,236.5	4,900.9	469.8	90.4
	1996	24,366.8	5,663.8	328.5	94.2
	1997	36,473.1	4,269.1	683.0	84.0
	1998	23,663.0	1,748.0	1,659.8	5.0
	1999	28,201.6	2,716.4	742.9	72.6
	2000	21,403.8	1,438.1	516.7	64.1

All instances are solved in two iterations for Algorithm 1 with a 0% MIP gap. For details, see Sect. 4

5 Conclusions

5.1 Limitations

At least two limitations of our proposed algorithm offer work for future research. First, as we mention in Sect. 3, the algorithm fails when the `Existing` model is infeasible. However, checking the usability of Budget-Cut Algorithm for an optimization model requires the solution of a single LP; computationally this is a small effort. Future work could focus on determining feasible start solutions for Budget-Cut Algorithm, as well as solving the `Existing` and `Extended` problems in parallel. Next, given a feasible solution to initiate the algorithm, we can determine a valid upper bound for model (3). That being said, determining feasible solutions for a MIP is, in general, hard [4]. However, for certain classes of problems—such as the one we present—feasibility is often maintained when no investment decisions are made [21].

A second limitation of Budget-Cut Algorithm occurs when the budget is too large; i.e., $b > \sum a_i$. In this case, the algorithm skips directly to the final solution of the MIP without a valid inequality. The only benefits of our proposal in this case is the use of a feasible solution from `Extended` as a warmstart. However, even then, the computational benefits we demonstrate in Sect. 4 with the SelfScen instances are significant.

Finally, we mention that several practical problems include a known budget—the maximum possible expenditure for building new components. Then, the parameter b of Algorithm Budget-Cut Algorithm serves as an input.

5.2 Summary

To summarize, we present a simple-to-implement algorithm for reducing runtimes of a capacity extension problem. This problem is computationally demanding when exponentially many choices for installing new components exist. Intuitively, the general class of problems we study addresses the following concern: given a portfolio of potential investments with varying purchase and operation costs, choose the ones that minimize long-term horizon expenditures subject to a given budget. Such situations also find application in the general setting when investment decisions are optional and only provide an enhanced portfolio, thereby the original problem serves as a base-case. We relate this problem to a knapsack problem, and propose an algorithm that determines a valid inequality to cut off suboptimal branches of the branch-and-bound search tree. Our algorithm rests on determining upper and lower bounds by solving two extremes of problems — the first where no optional components are built, and the second where all optional components are built, respectively. By iteratively pruning off suboptimal solutions, we increase the lower bounds obtained by the second of these extremes.

Documentation and data for the FINE code is available under the MIT License at https://github.com/FZJ-IEK3-VSA/FINE. Data for the SelfScen is available under the MIT License at https://data.mendeley.com/datasets/zhwkrc6k93/1 [16].

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s11590-021-01826-w.

Acknowledgements We are grateful to Dane Lacey for the modification of the district scenario that we present in Online Appendix A.3. The authors acknowledge the financial support by the Federal Ministry for Economic Affairs and Energy of Germany in the project METIS (project number 03ET4064).

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Banos, R., Manzano-Agugliaro, F., Montoya, F., Gil, C., Alcayde, A., Gómez, J.: Optimization methods applied to renewable and sustainable energy: A review. Renew. Sustain. Energy Rev. **15**(4), 1753–1766 (2011). https://doi.org/10.1016/j.rser.2010.12.008
2. Billinton, R., Karki, R.: Capacity expansion of small isolated power systems using PV and wind energy. IEEE Trans. Power Syst. **16**(4), 892–897 (2001). https://doi.org/10.1109/59.962442
3. Bundesregierung.de: Das Energiekonzept 2050 (2010). https://www.bundesregierung.de/resource/blob/997532/778196/c6acc2c59597103d1ff9a437acf27bd/infografik-energie-textversion-data.pdf?download=1. Accessed 07 Dec 2020
4. Chinneck, J.W.: Feasibility and infeasibility in optimization: algorithms and computational methods, vol. 118. Springer Science & Business Media, Berlin (2007)
5. Ember: Daily EU ETS carbon market price (Euros). https://ember-climate.org/data/carbon-price-viewer. Accessed 26 Jan 2021
6. European Commission: The roadmap for transforming the EU into a competitive, low-carbon economy by 2050. Tech. rep., European Commission (2011). https://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=COM:2011:0112:FIN:EN:PDF. Accessed 19 Dec 2020
7. Forschungszentrum, J.: Welcome to FINE's documentation! https://vsa-fine.readthedocs.io/en/master/index.html. Accessed 21 Nov 2020
8. Gabrielli, P., Gazzani, M., Martelli, E., Mazzotti, M.: Optimal design of multi-energy systems with seasonal storage. Appl. Energy **219**, 408–424 (2018). https://doi.org/10.1016/j.apenergy.2017.07.142
9. Goderbauer, S., Bahl, B., Voll, P., Lübbecke, M.E., Bardow, A., Koster, A.M.: An adaptive discretization MINLP algorithm for optimal synthesis of decentralized energy supply systems. Comput. Chem. Eng. **95**, 38–48 (2016). https://doi.org/10.1016/j.compchemeng.2016.09.008
10. Gurobi Optimization LLC: Gurobi optimizer reference manual (2020). http://www.gurobi.com. Accessed 03 Dec 2020
11. Hart, W.E., Watson, J.P., Woodruff, D.L.: Pyomo: Modeling and solving mathematical programs in python. Math. Program. Comput. **3**(3), 219–260 (2011). https://doi.org/10.1007/s12532-011-0026-8
12. Hoffmann, M., Kotzur, L., Stolten, D., Robinius, M.: A review on time series aggregation methods for energy system models. Energies **13**, 641 (2020). https://doi.org/10.3390/en13030641

13. International Renewable Energy Agency (IRENA): Renewable power generation costs in 2019 (2020). https://www.irena.org/-/media/Files/IRENA/Agency/Publication/2020/Jun/IRENA_Power_Generation_Costs_2019.pdf. Accessed 29 Jan 2021
14. Jülich Supercomputing Centre: JURECA: Modular supercomputer at Jülich Supercomputing Centre. J. Large-scale Res. Facil. JLSRF (2018). https://doi.org/10.17815/jlsrf-4-121-1
15. Kannengießer, T., Hoffmann, M., Kotzur, L., Stenzel, P., Schuetz, F., Peters, K., Nykamp, S., Stolten, D., Robinius, M.: Reducing computational load for mixed integer linear programming: an example for a district and an island energy system. Energies (2019). https://doi.org/10.3390/en12142825
16. Knosala, K., Kotzur, L., Röben, F.T., Stenzel, P., Blum, L., Robinius, M., Stolten, D.: Hybrid hydrogen home storage for decentralized energy autonomy. Int. J. Hydrogen Energy **46**(42), 21748–21763 (2021). https://doi.org/10.1016/j.ijhydene.2021.04.036
17. Knosala, K., Kotzur, L., Röben, F.T., Stenzel, P., Blum, L., Robinius, M., Stolten, D.: Hybrid hydrogen home storage for decentralized energy autonomy. Mendeley Data available at https://data.mendeley.com/datasets/zhwkrc6k93/1
18. Kotzur, L.: Future grid load of the residential building sector. Ph.D. thesis, RWTH Aachen (2018)
19. Kotzur, L., Markewitz, P., Robinius, M., Stolten, D.: Kostenoptimale Versorgungssysteme für ein vollautarkes Einfamilienhaus. In: 10. Internationale Energiewirtschaftstagung, vol. 10, pp. 1–14 (2017)
20. Kotzur, L., Nolting, L., Hoffmann, M., Groß, T., Smolenko, A., Priesmann, J., Büsing, H., Beer, R., Kullmann, F., Singh, B., Praktiknjo, A., Stolten, D., Robinius, M.: A modeler's guide to handle complexity in energy system optimization (2020). arXiv:2009.07216. Accessed 03 Feb 2021
21. Lim, S.R., Suh, S., Kim, J.H., Park, H.S.: Urban water infrastructure optimization to reduce environmental impacts and costs. J. Environ. Manag. **91**(3), 630–637 (2010). https://doi.org/10.1016/j.jenvman.2009.09.026
22. Loulou, R., Goldstein, G., Noble, K., et al.: Documentation for the MARKAL family of models. Energy Technology Systems Analysis Programme pp. 65–73 (2004)
23. Loulou, R., Remme, U., Kanudia, A., Lehtila, A., Goldstein, G.: Documentation for the TIMES model part II. Energy Technology Systems Analysis Programme (2005)
24. Lumbreras, S., Ramos, A., Banez-Chicharro, F.: Optimal transmission network expansion planning in real-sized power systems with high renewable penetration. Electric Power Syst. Res. **149**, 76–88 (2017). https://doi.org/10.1016/j.epsr.2017.04.020
25. Luss, H.: Operations research and capacity expansion problems: a survey. Oper. Res. **30**(5), 907–947 (1982). https://doi.org/10.1287/opre.30.5.907
26. Mahdavi, M., Sabillon, C., Ajalli, M., Romero, R.: Transmission expansion planning: literature review and classification. IEEE Syst. J. **3**, 3129–3140 (2019). https://doi.org/10.1109/JSYST.2018.2871793
27. Neumann, F., Brown, T.: Heuristics for transmission expansion planning in low-carbon energy system models. In: 2019 16th International Conference on the European Energy Market (EEM), pp. 1–8 (2019). https://doi.org/10.1109/EEM.2019.8916411
28. Samsatli, S., Staffell, I., Samsatli, N.J.: Optimal design and operation of integrated wind-hydrogen-electricity networks for decarbonising the domestic transport sector in Great Britain. Int. J. Hydrogen Energy **41**(1), 447–475 (2016). https://doi.org/10.1016/j.ijhydene.2015.10.032
29. Singh, B., Morton, D.P., Santoso, S.: An adaptive model with joint chance constraints for a hybrid wind-conventional generator system. CMS **15**(3–4), 563–582 (2018). https://doi.org/10.1007/s10287-018-0309-x
30. Üster, H., Dilaveroğlu, Ş: Optimization for design and operation of natural gas transmission networks. Appl. Energy **133**, 56–69 (2014). https://doi.org/10.1016/j.apenergy.2014.06.042
31. Welder, L., Ryberg, D., Kotzur, L., Grube, T., Robinius, M., Stolten, D.: Spatio-temporal optimization of a future energy system for power-to-hydrogen applications in germany. Energy **158**, 1130–1149 (2018). https://doi.org/10.1016/j.energy.2018.05.059
32. Yao, L., Yang, B., Cui, H., Zhuang, J., Ye, J., Xue, J.: Challenges and progresses of energy storage technology and its application in power systems. Journal of Modern Power Systems and Clean Energy (2016). https://doi.org/10.1007/s40565-016-0248-x
33. Zhang, Y., Sahinidis, N.V.: Global optimization of mathematical programs with complementarity constraints and application to clean energy deployment. Optim. Lett. **10**(2), 325–340 (2016). https://doi.org/10.1007/s11590-015-0880-9