Progressive hedging for stochastic energy management systems

The mixed-integer linear case

Valentin Kaisermayer^{1,2} · Daniel Muschick¹ · Martin Horn^{1,2} · Markus Göllles^{1,2}

Received: 19 December 2019 / Accepted: 6 August 2020 / Published online: 29 August 2020
© The Author(s) 2020

Abstract

Energy systems have increased in complexity in the past years due to the ever-increasing integration of intermittent renewable energy sources such as solar thermal or wind power. Modern energy systems comprise different energy domains such as electrical power, heating and cooling which renders their control even more challenging. Employing supervisory controllers, so-called energy management systems (EMSs), can help to handle this complexity and to ensure the energy-efficient and cost-efficient operation of the energy system. One promising approach are

The research leading to these results was carried out in two research projects. One part of the research leading to these results received funding from the COMET program managed by the Austrian Research Promotion Agency under Grant Number 869341. The program is co-financed by the Republic of Austria and the Federal Provinces of Lower Austria, Styria and Vienna. The financial support by the Austrian Federal Ministry of Science, Research and Economy and the National Foundation for Research, Technology and Development is gratefully acknowledged. The other part of the research leading to these results received was part of the Austrian flagship project ThermoFLEX managed by the FFG under Grant Number 868852 and funded by the Austrian Climate and Energy Fund in the framework of the research initiative “Green Energy Lab” as part of the Austrian innovation campaign “Vorzugsregion Energie”.

✉ Valentin Kaisermayer
valentin.kaisermayer@best-research.eu; valentin.kaisermayer@tugraz.at

Daniel Muschick
daniel.muschick@best-research.eu

Martin Horn
martin.horn@tugraz.at

Markus Göllles
markus.goellles@best-research.eu; markus.goellles@tugraz.at

¹ BEST - Bioenergy and Sustainable Technologies GmbH, Inffeldgasse 21/B, 8010 Graz, Austria

² Institute of Automation and Control, Graz University of Technology, Inffeldgasse 21/B, 8010 Graz, Austria

optimization-based EMS, which can for example be modelled as stochastic mixed-integer linear programmes (SMILP). Depending on the problem size and control horizon, obtaining solutions for these in real-time is a difficult task. The progressive hedging (PH) algorithm is a practical way for splitting a large problem into smaller sub problems and solving them iteratively, thus possibly reducing the solving time considerably. The idea of the PH algorithm is to aggregate the solutions of subproblems, where artificial costs have been added. These added costs enforce that the aggregated solutions become non-anticipative and are updated in every iteration of the algorithm. The algorithm is relatively simple to implement in practice, re-using almost all of a possibly existing deterministic implementations and can be easily parallelized. Although it has no convergence guarantees in the mixed-integer linear case, it can nevertheless be used as a good heuristic for SMILPs. Recent theoretical results shown that for applying augmented Lagrangian functions in the context of mixed-integer programmes, any norm proofs to be a valid penalty function. This is not true for squared norms, like the squared L_2 -norm that is used in the classical progressive hedging algorithm. Building on these theoretical results, the use of the L_1 and L_∞ -norm in the PH algorithm is investigated in this paper. In order to incorporate these into the algorithm an adapted multiplier update step is proposed. Additionally a heuristic extension of the aggregation step and an adaptive penalty parameter update scheme from the literature is investigated. The advantages of the proposed modifications are demonstrated by means of illustrative examples, with the application to SMILP-based EMS in mind.

Keywords Energy management system · Stochastic programming · Progressive hedging · Mixed-integer linear programming

1 Introduction

The increased complexity of energy systems renders their operation a difficult task. An energy system may incorporate intermittent producers such as solar thermal, photovoltaics and wind power, and may span multiple energy domains such as electrical power, heating and cooling. Energy systems that span multiple energy domains are often referred to as *multi energy systems* (MES), see e.g. [18, 32]. In order to control such complex energy systems, a *supervisory control* approach is needed. These supervisory controllers are often referred to as *energy management systems* (EMS), see e.g. [29]. The EMS then relies on *low-level* controllers that actually control the producers and energy distribution in the MES.

There are numerous possibilities for implementing EMS for MES. The simplest approach is to use a rule-based system, see e.g. [29]. However, for complex energy systems that incorporate multiple producers and consumers on different energy domains, deriving rules may be a very complex task. It may be especially difficult to incorporate forecasts of the future demand, yield or price into such rule-based EMS.

A common approach to mitigate these problems is to use an optimization-based EMS, see e.g. [8]. In this case, mathematical models of the individual energy system components, together with forecasts of the future demand, yield and prices, are used in a mathematical programme to compute an optimal schedule for a given control horizon. This is typically done in a 1-day-ahead schedule, see e.g. [27].

A variant of this optimization-based EMS employs the *receding horizon methodology*. In this case an optimal control schedule for a control horizon of typically 24 h is computed, but only the first time instance of the computed control schedule is applied to the MES. After a specific time interval has passed, newly available measurement information is considered and an optimal control schedule for the next horizon is computed. The receding horizon methodology provides robustness to model errors, forecast uncertainties or price variations. This control approach is widely used in the industry and commonly referred to as *model predictive control* (MPC).

For optimization-based approaches to deliver satisfactory control schedules, a sufficiently descriptive mathematical model of the whole MES, an expressive cost function and reliable forecasts are needed. In order to keep the problem sufficiently simple to be solvable in real-time, both the mathematical model and the cost function are often assumed to be linear in the decision variables. The addition of boolean or integer variables allows the formulation of more descriptive models, but at the cost of increasing the problem's complexity. The resulting mathematical programme that has to be solved is therefore often a mixed-integer linear programme (MILP). Algorithms like *branch-and-bound* and *branch-and-cut* are efficient algorithms for solving such MILPs, see e.g. [1]. However, the computational complexity of a MILP depends on two main factors: the number of integer variables and the tightness of the problem, i.e. the difference between the MILP solution and the solution of its linear programme (LP) relaxation, see e.g. [30]. In the worst case the computational complexity grows exponentially with the number of integer variables, see e.g. [25]. Therefore, splitting the MILP into smaller subproblems can help to manage larger problems that otherwise would not be tractable.

Since a perfect forecast of the actual demand or yield is unrealistic, however, an MPC approach that only considers one possible scenario of these forecasts for the EMS (so-called *deterministic* MPC) may not handle this uncertainty sufficiently well. In this case, the consideration of multiple scenarios representing the numerical distribution of the uncertain variables in a *stochastic* MPC approach can yield to improved control schedules, see e.g. [27]. The stochastic MPC can be modelled as so-called recourse problem, which is a stochastic programme that models some kind of optimal decision-making process, that is done in stages. Each stage determines when new information is available and uncertainty is revealed, see e.g. [23]. The simplest case is a *two-stage* problem, where all decisions from the first stage have to be made *non-anticipative* of the actual realization of the uncertainty from the second stage. Decisions from the second stage are the *recourse decisions* for the actual realization of the uncertainty. In case of the stochastic MPC-based EMS, *stochastic programmes* (SPs) have to be solved in real-time. In order to obtain robust solutions, possibly very many scenarios, i.e. realizations of the uncertainty have to be considered. The disadvantage of this approach is that the problem complexity now also grows with the number of scenarios.

In order to utilize the stochastic MPC approach in real-time, it is necessary to solve the stochastic mixed-integer programmes within tight time constraints. Numerous decomposition algorithms that aim to reduce the problem size by decomposing the main problem into smaller subproblems exist in literature, see e.g. [37] for an overview. Some tailored decomposition algorithms exist for specialized versions of stochastic programmes. These algorithms perform very well for the right class of problem, however, are typically difficult to implement.

If *only* the first stage incorporates integer variables and the second stage *only* real variables (for two-stage problems), decomposition algorithms such as *Benders' decomposition* could be used, see e.g. [24, 37]. The algorithm divides the SP into a *master problem* and a set of *subproblems*, corresponding to each scenario. These subproblems can be solved in parallel. However, since the classical Benders' decomposition relies on *strong duality*, it cannot be used in the context of general mixed-integer stochastic programmes, see e.g. [16]. In [11] Benders' decomposition is adopted for problems with only integer second-stage variables. In [16] Benders' decomposition is used within a branch-and-bound framework enabling it to be applied to problems with mixed-integer subproblems. However, one has to be able to access the dual subproblems, which might be difficult for existing deterministic implementations.

In [10] the *dual ascent method*, together with a branch-and-bound framework is used to solve general mixed-integer stochastic programmes, the algorithm is often referred to as *dual decomposition*. However, [9] states that dual ascent method are often not robust and converge slowly. Hence, in [20] the integration of the *progressive hedging algorithm* with the dual decomposition algorithm is proposed.

The progressive hedging algorithm is based on the augmented Lagrangian method, which combines the *penalty method* and the *method of Lagrange multipliers* to solve programmes with equality constraints. The augmented Lagrangian method is stated to be more robust than the dual ascent method, see e.g. [9]. Unfortunately, the classical PH algorithm is not guaranteed to converge in the non-convex mixed-integer case, see e.g. [12, 28]. Instead, it might experience problems such as *cyclic behaviour*, see e.g. [39]. Some heuristic extensions to the algorithm have been proposed to cope with this problem, see e.g. [39]. In any case, the algorithm is nevertheless used as a *good* heuristic for SMILPs, see e.g. [12, 28, 39, 40]. Since the EMS is a supervisory controller which can depend on the underlying, low-level controllers to operate the producers as desired, it is not necessary to obtain solutions with very tight tolerances, and heuristic solutions may be acceptable.

The classical PH algorithm uses a quadratic penalty function, which renders the mathematical programme to be solved *non-linear*, i.e. with quadratic terms in the objective function. This is a major disadvantage of the PH algorithm, especially in the mixed-integer case, since the computational complexity increases further due to the quadratic terms, making mixed-integer quadratic programmes (MIQPs) generally harder to solve than MILPs. In [40] the quadratic terms in the objective function are approximated by a piecewise affine function, eliminating the need for a quadratic solver. In contrast to other methods, PH has the advantage of being relatively easy to implement. Certain parts of the algorithm can also be easily parallelized. Due to the way the PH algorithm works, most, if not all, previously existing implementations

for solving deterministic problems can be re-used, which is a huge practical advantage. In other words, PH can be simply added on top of existing implementations. In [17] it is shown that for solving mixed-integer problems with an augmented Lagrangian function, as is done in the PH algorithm, a *sharp* Lagrangian should be used. It is shown that using *any* norm as the penalty function results in a *sharp augmented Lagrangian*. This is *not* true if a *squared* norm is used as in the classical PH algorithm [17]. Hence, in this paper the following question is addressed:

Is it possible to apply the theoretical results from [17] to the classical progressive hedging algorithm, improving its mixed-integer performance, with respect to the convergence time and solution quality?

The results from [17] prove that augmented Lagrangian functions can be used in the context of mixed-integer programming, but not how this can be incorporated into algorithms based on the augmented Lagrangian method such as PH. Building on the theoretical results from [17], in this paper the use of the L_1 and L_∞ -norm as penalty functions instead of the squared L_2 -norm-based penalty function from the classical PH algorithm is investigated and it is shown how these can be incorporated into the PH algorithm. Additionally, small heuristic modifications to the algorithm are proposed that improve the convergence speed, hence increasing the real-time capability. In order to show the validity of the results, the proposed modifications were implemented and are compared on three examples of multi energy systems for real-time EMS.

The remainder of this paper is structured as follows. First, in Sect. 2, an introduction into the used mathematical modelling approach for MES is given. The resulting EMS control problem is then stated, first in its deterministic, then in its stochastic formulation. In Sect. 3 the classical PH algorithm is described and the proposed modifications to the algorithm are explained. In Sect. 4 these modifications are

Fig. 1 Example of a multi energy system that spans three energy domains: *heat* (orange), *electrical power* (blue) and *gas* (yellow), incorporating heat and power consumers, a thermal storage buffer, a combined heat and power plant, a gas boiler and a power-to-heat plant (colour figure online)

evaluated on the example of representative energy management problems. In the end, conclusions are drawn and an outlook is given.

2 Optimization-based energy management system

The purpose of an energy engagement system (EMS) is to supervise the operation of an multi energy system (MES). Optimization-based EMS require a mathematical model of the MES suitable for this purpose.

An MES consists of a collection of interconnected *prosumers* (producers and consumers). Such prosumers of a typical MES may resemble various forms of *energy storage units*, *energy producers*, *energy consumers*, *sources* and *sinks*.

For example, the configuration in Fig. 1 spans three domains: *heat*, *power* and *gas*, and incorporates three producers: a *gas boiler*, a *combined heat and power plant* (CHP) and a *power-to-heat plant*. The gas boiler and the CHP are supplied with gas from the public gas grid, which is represented by a *gas source*. Additionally, a connection to the public power grid is represented by both a *power grid source* for buying and a *power grid sink* for selling electrical energy. The configuration also features a thermal storage (*thermal buffer*). The demand is represented through separate *heat* and *power consumers*. The heating grid is modelled with a feed line for hot water and a return line for cold water.

All the connections between the prosumers are modelled by simple balance equations. Each prosumer has its own mathematical model that describes its (possibly dynamic) behaviour.

2.1 Prosumer models

Each prosumer model consists of a set of constraints and a contribution to the overall cost function. The constraints mainly govern the dynamic behaviour of the prosumer; additional constraints can be e.g. ramping constraints, constraints on the number of startups per day and minimal up and down time constraints. The costs may include, among others, feedstock costs for e.g. gas or power, costs for switching individual producers on or off, or costs for emissions like CO_2 and NO_x . Each prosumer model may use discrete variables for e.g. modelling the on and off switching of a producer. Different prosumers are connected via mass and energy balance preserving equations. The used convention is that inflows of mass or energy into a prosumer are negative and outflows are positive. Hence, the balance equations are a simple sums over the respective inputs and outputs. A representative collection of these prosumer models is described in the following sections. If not explicitly stated otherwise all variables are considered to be real-valued.

2.1.1 Thermal energy storage

The thermal energy storage tank is modelled as a simple integrator, with a first order loss term in order to model the heat losses due to imperfect insulation:

$$m_{h,t+1} = e^{-\frac{T_s}{\tau}} m_{h,t} + \tau \left(1 - e^{-\frac{T_s}{\tau}}\right) (\dot{m}_{h,in,t} - \dot{m}_{h,out,t}) \quad (1a)$$

$$\dot{m}_{h,in,t} - \dot{m}_{h,out,t} + \dot{m}_{c,in,t} - \dot{m}_{c,out,t} = 0 \quad (1b)$$

$$0 \leq m_{h,t} \leq m_{\max}. \quad (1c)$$

The mass of hot water $m_{h,t}$ in the storage depends on the in- and outflows of hot water $\dot{m}_{h,in,t}$ and $\dot{m}_{h,out,t}$. This model assumes that the thermal energy storage tank is perfectly stratified i.e. split into two layers of *hot* and *cold* water. The temperatures of the hot and cold layer are assumed to be constant. Two additional constraints ensure the mass-flow and mass balance, where $m_{\max}$ is the mass of water in the buffer. The *time constant* τ describes the heat loss of the buffer and T_s is the sample time of the model. The thermal energy storage model has no costs associated with it.

2.1.2 Combined heat and power (CHP) plant

The model of a combined heat and power (CHP) plant describes a typical cross-sectoral prosumer that can be operated by the EMS:

$$P_{el,t} = \eta_{el}(z_t) \cdot \text{HHV} \cdot \dot{m}_{\text{gas},t}(z_t) \cdot u_{\text{on/off},t} \quad (2a)$$

$$\dot{Q}_{th,t} = \eta_{th}(z_t) \cdot \text{HHV} \cdot \dot{m}_{\text{gas},t}(z_t) \cdot u_{\text{on/off},t} \quad (2b)$$

$$\dot{m}_{c,in,t} c_p T_c + \dot{m}_{h,out,t} c_p T_h = \dot{Q}_{th,t} \quad (2c)$$

$$\dot{m}_{c,in,t} + \dot{m}_{h,out,t} = 0, \quad (2d)$$

where $P_{el,t}$ and $\dot{Q}_{th,t}$ are the electrical and thermal power produced, respectively, and $\dot{m}_{\text{gas},t}$ is the mass flow rate of gas required. The term HHV denotes the higher heating value of the gas. The cold water inflow and hot water outflow have constant temperatures T_c and T_h , respectively, and c_p is the specific thermal heat capacity of water, which can be assumed as constant in the considered temperature range. The mass flows $\dot{m}_{h,out,t}$ and $\dot{m}_{c,in,t}$ are the mass flows of hot and cold water in and out of the plant. The *operating point* of the plant i.e. how much power is produced, is determined by the control signal $z_t \in (0, 1)$ and the binary commitment $u_{\text{on/off},t}$. Since $u_{\text{on/off},t}$ is a binary variable, any terms in (2) involving multiplication with it and another term can be conveniently modelled with a set of four constraints, see e.g. [4, 32]. Therefore, these multiplications are not in violation of the chosen MILP formalism.

The electrical η_{el} and thermal efficiencies, η_{th} and therefore the fuel consumption $\dot{m}_{\text{gas},t}$ are in general non-linear functions of the operating point. The modelling assumption is that the products of those functions in (2a) and (2b) can be approximated sufficiently well with piece-wise linear functions. Modelling such functions within a MILP

is well known, but requires additional variables that satisfy a *special ordered set (SOS) type 2* constraint, which in turn needs additional binary variables to model, see e.g. [3].

The model for a gas boiler would be the same, but without the electrical output equation (2a) and higher thermal efficiency η_{th} . Hence, the model can be seen as general model for any producer in the MES.

It is assumed that the output power of the CHP is controlled via the dispatch $u_{d,t}$, while the commitment $u_{on/off,t}$ determines if the CHP is on or off. The linking of the real-valued *dispatch* $u_{d,t} \in \mathbb{R}$ and the binary *commitment* $u_{on/off,t}$ with the control signal z_t is modelled with:

$$z_t = u_{d,t} \cdot u_{on/off,t} \quad (3)$$

Ramping constraints and minimal and maximal dispatch are modelled via:

$$u_{d,min} \leq u_{d,t} \leq u_{d,max} \quad (4a)$$

$$\Delta u_{d,RD} \leq u_{d,t} - u_{d,t-1} \leq \Delta u_{d,RU} \quad (4b)$$

where $\Delta u_{d,RU}$ and $\Delta u_{d,RD}$ are the ramp-up and ramp-down limits, and $u_{d,min}$ and $u_{d,max}$ are the minimal and maximal dispatch.

Finally, minimal *up* and *down* time constraints of the plant are imposed via:

$$\sum_{i=t-T_{MUT}+1}^t v_i \leq u_{on/off,t}, \quad \text{and} \quad \sum_{i=t-T_{MDT}+1}^t w_i \leq 1 - u_{on/off,t}, \quad (5)$$

with

$$v_t - w_t = u_{on/off,t} - u_{on/off,t-1}, \quad (6)$$

where $v_t \in \mathbb{B}$ and $w_t \in \mathbb{B}$ are the binary indicators for a *startup* and *shutdown* and T_{MUT} and T_{MDT} are the minimal up and down times, respectively. Constraint (6) is known as the *3-bin* formulation, see e.g. [31]. Due to service contracts the number of startups per day might also be restricted:

$$\sum_{i \in \mathcal{D}_j} v_i \leq n_{MS}, \quad \forall j \in \mathcal{N} \quad (7)$$

where $\mathcal{D}_j$ is the set of time instances t within each day, $\mathcal{N}$ is the set of days within the considered control horizon and n_{MS} is the maximal number of startups per day.

The gas consumed by the CHP causes *fuel costs*, but these are modelled centrally at the source, in this case the *gas source*. In real-world applications there often is a cost associated with each hour the plant is running, i.e. due to existing service contracts. This can be easily modelled as:

$$c_{op,t} = C_{op} u_{on/off,t}, \quad (8)$$

where C_{op} is the *operations cost* coefficient and the sum of the operations costs per time interval $c_{op,t}$ is minimized in the objective function of the optimization problem

that needs to be solved by the EMS. Additional costs may include *startup costs* $c_{\text{startup},t}$ or *ramping costs* $c_{\text{ramping},t}$, which are modelled as:

$$c_{\text{startup},t} = C_{\text{startup}} v_t \quad (9a)$$

$$c_{\text{ramping},t} = C_{\text{ramping}} \alpha_t \quad (9b)$$

$$\frac{u_{d,t-1} - u_{d,t}}{T_s} \leq \alpha_t \leq \frac{u_{d,t} - u_{d,t-1}}{T_s} \quad (9c)$$

where C_{startup} and C_{ramping} are the startup and ramping cost coefficients and α_t models the absolute value of the rate of change of the dispatch $u_{d,t}$. Again the sums of the individual costs terms in (9) are minimized in the objective function. This ensures that frequent startups and sudden changes in the power output of the plant are penalized.

2.1.3 Thermal solar collector

As an example of an uncontrolled energy producer a thermal solar collector is considered. For this a forecast of the thermal yield $\dot{Q}_{\text{yield},t} \geq 0$ for each time instance of the considered control horizon is needed. The thermal solar collector is modelled as

$$\dot{m}_{c,\text{in},t} c_p T_c + \dot{m}_{h,\text{out},t} c_p T_h \leq \dot{Q}_{\text{yield},t} \quad (10a)$$

$$\dot{m}_{c,\text{in},t} + \dot{m}_{h,\text{out},t} = 0. \quad (10b)$$

The modelling assumption is again, that the temperatures of the cold water inflow $\dot{m}_{c,\text{in},t}$ and hot water outflow $\dot{m}_{h,\text{out},t}$ are assumed to be fixed at T_c and T_h , respectively. The inequality in (10a) allows for *shedding* of the thermal yield of the solar collector and equation (10b) ensures the mass flow balance.

2.1.4 Heat and power consumption

In order to model the heat demand of e.g. a residential heating grid, a forecast of the heat demand $\dot{Q}_{\text{demand},t}$ is assumed to be known. The model of the *heat consumer* can be stated as

$$\dot{m}_{h,\text{in},t} c_p T_h + \dot{m}_{c,\text{out},t} c_p T_c + \dot{Q}_{\text{demand},t} = 0 \quad (11a)$$

$$\dot{m}_{h,\text{in},t} + \dot{m}_{c,\text{out},t} = 0, \quad (11b)$$

where $\dot{m}_{h,\text{in},t}$ and $\dot{m}_{c,\text{out},t}$ are the hot and cold mass flows of water in and out of the heat consumer. The temperatures T_h and T_c of the hot inflow and cold outflow are again assumed to be constant. Electrical power consumption is simply modelled as:

$$P_{\text{el},\text{in},t} + P_{\text{el},\text{demand},t} = 0, \quad (12)$$

where the forecasted electrical power demand $P_{\text{el},\text{demand},t}$ is assumed to be known and $P_{\text{el},\text{in},t}$ is the input of electrical power into the *power consumer*. Since only consumption is modelled, $\dot{Q}_{\text{demand},t} \geq 0$ and $P_{\text{el},\text{demand},t} \geq 0$ holds.

2.1.5 Sources and sinks

Any source or sink, e.g. a power grid connection or a connection to a public gas grid, simply provides or takes energy or mass flow at a given cost per unit. The cost may very well depend on time and can also be forecast.

The overall MES model is the union of all the individual prosumer models together with the mass and balance preserving equations due to the connections between them. It is a set of constraints that need to hold true at each time instance t from the current time until the end of the control horizon $t + N_c$.

2.2 Deterministic EMS control problem

Overall, the optimization problem for an EMS that acts as a supervisory controller for an MES can be cast into a mixed-integer linear programme (MILP) of the following form:

$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathcal{X}}{\operatorname{argmin}} \quad \mathbf{c}^\top \mathbf{x} \quad (13a)$$

$$\text{subject to} \quad \mathbf{A}\mathbf{x} = \mathbf{b}, \quad (13b)$$

with $\mathbf{x} \in \mathbb{R}^n \times \mathbb{Z}^m$ being the vector of optimization variables. The vector $\mathbf{c}$ represents the cost coefficients of the objective function. The matrix $\mathbf{A}$ together with the vector $\mathbf{b}$ represent the equality constraints (inequality constraints are omitted for a simpler notation). The vector of optimization variables $\mathbf{x}$ comprises all variables from the considered MES model, except for the forecast variables of e.g. heat demand or solar thermal yield. The set $\mathcal{X}$ determines which variables are real, integer or binary and also specifies finite lower and upper bounds on each of the variables. The solution to the mathematical programme is the optimal control schedule for the MES for the considered control horizon.

If all forecast variables of the MES are assumed to be known exactly, in turn all coefficients in (13) are known, hence (13) is referred to as a *deterministic control problem*. The solution to (13) can be obtained with standard off-the-shelf MILP solvers such as Gurobi [21] or CPLEX [22].

However, in practice not all forecasts are exact and thus not all coefficients in (13) are known exactly. If at least a numerical representation of the distribution of the uncertain variables is known, one can solve the resulting *stochastic control problem* by minimizing its expected costs. If the forecast uncertainty is large, this can lead to improved control schedules, see e.g. [27].

2.3 Stochastic EMS control problem

Stochastic programmes are typically used to model a kind of optimal decision-making process. In a so-called *recourse problem*, the decision making process is modelled in stages, see e.g. [23]. First, some decisions are made without any knowledge, hence *non-anticipative* of the actual realization of the uncertain variables; these decisions therefore need to lead to good results independent of the actual outcome of the uncertain variables. In a later step, after these decisions have been made, different possible realizations (*scenarios*) of the probability distribution are investigated, and for each realization so-called *recourse decisions* are made that lead to an optimal outcome in terms of the objective function value. This decision-making process could have many steps or *stages*; in each stage, only a subset of the probability distribution is revealed. While a multi-stage approach would allow for a more detailed modelling of the decision making process, the number of decision variables, and hence the computational cost, would increase exponentially with every stage. While the assumption of a two-stage approach, that all uncertainty is revealed at once, does not modelled the real-world decision-making process of an MPC correctly, in [27] the two-stage modelling approach is combined with the receding horizon methodology of an MPC-based EMS and demonstrated to perform well. Therefore, for simplicity only *two-stage* recourse problems are considered in this paper. However, the presented results are expected to also work for the multi-stage case.

The basic mathematical formulation of such a two-stage recourse problem is the following stochastic programme, see e.g. [23]:

$$\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x} \in \mathcal{X}} \quad \mathbf{c}^\top \mathbf{x} + \mathbb{E}_{\omega}[Q(\mathbf{x}, \omega)] \quad (14a)$$

$$\text{subject to} \quad \mathbf{A}\mathbf{x} = \mathbf{b}. \quad (14b)$$

Equation (14) is called the first-stage problem, with $\mathbf{x}$ being the vector of first-stage variables, or *here-and-now decision variables*, that have to be selected before the realizations of the uncertain variables ω are known. The function $Q(\mathbf{x}, \omega)$ is the second-stage cost function, which depends on both the first-stage variables and on the uncertain variables ω . Hence $Q(\mathbf{x}, \omega)$ is a *random variable* and can only be used in the first-stage cost function via statistical measures like its expected value $\mathbb{E}_{\omega}[\cdot]$ with respect to the vector of uncertain variables ω . The second-stage cost function $Q(\mathbf{x}, \omega)$ is an optimization problem in itself, see e.g. [23]:

$$Q(\mathbf{x}, \tilde{\omega}) = \operatorname{argmin}_{\mathbf{y} \in \mathcal{Y}} \quad \mathbf{q}(\tilde{\omega})^\top \mathbf{y} \quad (15a)$$

$$\text{subject to} \quad \mathbf{T}(\tilde{\omega})\mathbf{x} + \mathbf{W}(\tilde{\omega})\mathbf{y} = \mathbf{h}(\tilde{\omega}). \quad (15b)$$

The vector $\mathbf{y}$ represents the so-called *recourse variables*. The set $\mathcal{Y}$ determines which recourse variables $\mathbf{y}$ are real, integer or binary and also specifies finite lower and upper bounds on them. Note that the second-stage cost coefficients $\mathbf{q}$, the so-called *technology matrix* $\mathbf{T}$, the *recourse matrix* $\mathbf{W}$ and the right-hand side of the

second-stage constraints $\mathbf{h}$ can all depend on the specific realization $\tilde{\omega}$ of the uncertain variables ω .

It is assumed that the uncertain variables ω can be numerically represented by a *discrete distribution*, with $\mathcal{S}$ being the set of realizations obtained from this numerical distribution, and p_s the probability for each realization: $\sum_{s \in \mathcal{S}} p_s = 1$.

Then the expected value of the second-stage cost in (14) can be written as follows:

$$\mathbb{E}_{\omega}[Q(\mathbf{x}, \omega)] = \sum_{s \in \mathcal{S}} p_s Q(\mathbf{x}, \omega_s), \quad (16)$$

with ω_s being the realizations of the uncertain variable ω and p_s their respective probabilities. In order to be computationally tractable, only a *small* set of scenarios can be used in practice. In the next section the process of obtaining a small but representative set of scenarios, which is referred to as *scenario generation and reduction* will be elaborated on.

2.3.1 Scenario generation and reduction

In this paper the realizations of the uncertain variables, or *scenarios*, are obtained by the following process, see e.g. [34]. First, *historic point forecasts* of the uncertain variables are obtained by using special forecast algorithms for e.g. the heat demand or solar thermal yield, see. e.g. [33, 38]. Using these historic point forecasts and *historic measurements*, residuals can be computed for which a stochastic *residual model* is fitted. By adding realizations of the previously fitted stochastic residual model to a computed point forecast, a large number of scenarios is obtained. Because seasonal trends are already captured well by the point forecasts, a simple auto-regressive (AR) process is used for the stochastic residual model in this paper, see e.g. [15]. The obtained realizations are assumed to be equiprobable and are used as a basis for heuristic scenario reduction using the *k-means* or *k-medoids* clustering algorithms, see e.g. [14, 36]. With this process a set of scenarios for the future heat and power demand is obtained. The reduced scenario set $\mathcal{S}$ is then used in the stochastic programme (14)–(16).

2.3.2 First and second-stage variables

Before reformulating the control problem for the EMS as a stochastic programme, one has to decide which are the first-stage decisions and which are the recourse decisions. The main question is: When is insight into the actual realizations of the uncertain variables made available? In the case of MPC-based EMS new information is made available with new measurements after each decision. This, however, would result in a multi-stage problem. Following the receding horizon methodology, only the solution for the first time instance is really applied to the MES, and the decision for the next time step is actually based on a new problem which is solved in the next time step. Therefore it can reasonably be assumed that all the uncertainty after the first time step is revealed for the whole control horizon at once, see e.g. [27]. Hence, only the first time step of the control variables of each producer are considered first-stage variables, all other variables are considered to be second-stage variables.

Solving the stochastic programme (14), (15) and (16) can be quite difficult and, depending on the problem size, intractable. The problem size of a stochastic programme depends not only on the number of decision variables but also on the number of scenarios. The simplest option is to solve the *extensive form* (EF) of the stochastic programme. It is a standard way of writing a stochastic programme where the uncertainty is represented via a discrete scenario set $\mathcal{S}$.

2.3.3 Extensive form problem formulation

Assuming a discrete set of scenarios ω_s the *extensive form* (EF) of (14)–(16) can be stated as:

$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathcal{X}, \mathbf{y}_s \in \mathcal{Y}}{\operatorname{argmin}} \quad \mathbf{c}^\top \mathbf{x} + \sum_{s \in \mathcal{S}} p_s \mathbf{q}_s^\top \mathbf{y}_s \quad (17a)$$

$$\text{subject to} \quad \mathbf{A}\mathbf{x} = \mathbf{b} \quad (17b)$$

$$\mathbf{T}_s \mathbf{x} + \mathbf{W}_s \mathbf{y}_s = \mathbf{h}_s, \quad \forall s \in \mathcal{S}, \quad (17c)$$

where the following abbreviations for the second-stage cost coefficients $\mathbf{q}_s$ and the recourse variable $\mathbf{y}_s$ are used:

$$\mathbf{q}_s := \mathbf{q}(\omega_s), \quad \mathbf{y}_s := \mathbf{y}(\omega_s).$$

Similar abbreviations are used for the technology matrix $\mathbf{T}_s$, the recourse matrix $\mathbf{W}_s$ and the right-hand side of the second stage constraint $\mathbf{h}_s$:

$$\mathbf{T}_s := \mathbf{T}(\omega_s), \quad \mathbf{W}_s := \mathbf{W}(\omega_s), \quad \mathbf{h}_s := \mathbf{h}(\omega_s).$$

The resulting mathematical programme is also called the *deterministic equivalent* form, since it is essentially just a bigger version of (13) that incorporates all scenarios in one big mathematical programme.

For some problems the EF formulation can be solved using off-the-shelf solvers and hardware. However, for larger problems with many scenarios it may not be tractable under tight time constraints. If it were not for the first-stage variables $\mathbf{x}$, the problem (17) would be separable into each scenario *subproblem*. Decomposition methods exploit this fact and try to separate the stochastic programme by scenario.

With the MES model described in Sect. 2 and the choice of first-stage variables described in Sect. 2.3.2, the resulting EMS control problem is a stochastic programme, that incorporates integer variables in both the first *and* the second stage, however. An example of an algorithm able to cope with this, at least as a heuristic, is the *progressive hedging algorithm*, which is also often called *scenario aggregation algorithm*. In contrast to other methods, PH has the advantage of being relatively easy to implement. Additionally, due to the way the PH algorithm works, most, if not all, previously existing implementations for solving deterministic problems can be re-used, which is a huge practical advantage.

3 Solving stochastic programmes with progressive hedging

The progressive hedging (PH) algorithm is based on the *split-variable* formulation of (17), which can be compactly written as

$$\mathbf{x}^* = \underset{\mathbf{x}_s \in \mathcal{X}, \mathbf{y}_s \in \mathcal{Y}, \bar{\mathbf{x}}}{\operatorname{argmin}} \sum_{s \in \mathcal{S}} p_s (\mathbf{c}^\top \mathbf{x}_s + \mathbf{q}_s^\top \mathbf{y}_s) \quad (18a)$$

$$\text{subject to} \quad \mathbf{A}\mathbf{x}_s = \mathbf{b}, \quad \forall s \in \mathcal{S} \quad (18b)$$

$$\mathbf{T}_s \mathbf{x}_s + \mathbf{W}_s \mathbf{y}_s = \mathbf{h}_s, \quad \forall s \in \mathcal{S} \quad (18c)$$

$$\mathbf{x}_s - \bar{\mathbf{x}} = \mathbf{0}, \quad \forall s \in \mathcal{S}, \quad (18d)$$

where copies of the first-stage variables $\mathbf{x}_s$ are introduced for each scenario. The *non-anticipativity constraint* (18d) enforces the *non-anticipativity* of the solution: The vector $\bar{\mathbf{x}}$ is called the *non-anticipative* solution of the problem. The non-anticipativity constraint (18d) is the only difficulty that prohibits a *scenario-wise decomposition* of (18). The idea behind a scenario-wise decomposition is to relax the constraint (18d), enabling the scenario subproblems to be solved separately.

The progressive hedging algorithm is based on the idea of relaxing the non-anticipativity constraint, solving all scenario subproblems separately and then iteratively achieving consensus on the *non-anticipative* first-stage solution (although PH can also be used for multi-stage problems, the focus in this paper is on two-stage problems).

The core idea of PH can be motivated via the *augmented Lagrangian method*, see e.g. [5, 6, 26]. With the augmented Lagrangian method, constrained problems of the following form can be solved

$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathcal{X}}{\operatorname{argmin}} f(\mathbf{x}) \quad (19a)$$

$$\text{subject to} \quad \mathbf{g}(\mathbf{x}) = \mathbf{0}, \quad (19b)$$

where $\mathbf{g}(\mathbf{x})$ are m equality constraints. The idea is to relax the constraints $\mathbf{g}(\mathbf{x})$ with the use of a *penalty function* $\psi(\cdot)$, but still keep the constraints, resulting in the mathematical programme

$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathcal{X}}{\operatorname{argmin}} f(\mathbf{x}) + \rho \psi(\mathbf{g}(\mathbf{x})) \quad (20a)$$

$$\text{subject to} \quad \mathbf{g}(\mathbf{x}) = \mathbf{0}, \quad (20b)$$

where $\rho > 0$ is a positive penalty parameter and $\psi(\cdot)$ is a penalty function that fulfils

$$\psi(\mathbf{0}) = 0, \quad \psi(\mathbf{v}) > 0, \quad \forall \mathbf{v} \neq \mathbf{0}. \quad (21)$$

The augmented Lagrangian method applies the *method of Lagrange multipliers* on (20), resulting in the *augmented Lagrangian* function

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda^\top \mathbf{g}(\mathbf{x}) + \rho \psi(\mathbf{g}(\mathbf{x})), \quad (22)$$

where $\lambda \in \mathbb{R}^m$ is the vector of *Lagrange multipliers*, also often called *dual variables*, see e.g. [24]. The augmented Lagrangian method then sequentially solves the following programme, see e.g. [6]:

$$\mathbf{x}^{(k+1)} = \underset{\mathbf{x} \in \mathcal{X}}{\operatorname{argmin}} L(\mathbf{x}, \lambda^{(k)}) \quad (23)$$

thus obtaining a new solution $\mathbf{x}^{(k+1)}$ in each iteration k . The Lagrange multipliers $\lambda^{(k)}$ are also updated iteratively, see e.g. [9]. A derivation of this update rule is described next. Since $\mathbf{x}^{(k+1)}$ minimizes (23) the following holds:

$$\begin{aligned} \mathbf{0} &\in \partial f(\mathbf{x}^{(k+1)}) + \partial \mathbf{g}(\mathbf{x}^{(k+1)}) \lambda^{(k)} + \rho \partial \mathbf{g}(\mathbf{x}^{(k+1)}) \partial \psi(\mathbf{g}(\mathbf{x}^{(k+1)})) \\ &= \partial f(\mathbf{x}^{(k+1)}) + \partial \mathbf{g}(\mathbf{x}^{(k+1)}) (\lambda^{(k)} + \rho \partial \psi(\mathbf{g}(\mathbf{x}^{(k+1)}))), \end{aligned} \quad (24)$$

where $\partial \psi(\mathbf{v})$ is the derivative of the penalty function with respect to its arguments and not with respect to $\mathbf{x}$. If the functions in (24) and (25) are continuously differentiable, the inclusion $\in$ can be replaced by an equality and ∂ by ∇ , see e.g. [9]. Hence, if the Lagrange multipliers $\lambda^{(k)}$ are updated in the following way

$$\lambda^{(k+1)} = \lambda^{(k)} + \rho \nabla \psi(\mathbf{g}(\mathbf{x}^{(k+1)})), \quad (25)$$

one gets the stationary condition of the original problem (19):

$$\mathbf{0} \in \partial f(\mathbf{x}^{(k+1)}) + \partial \mathbf{g}(\mathbf{x}^{(k+1)}) \lambda^{(k+1)}. \quad (26)$$

The update step (25) is commonly referred to as the *subgradient scheme*, see e.g. [9]. The positive penalty parameter $\rho > 0$ in (23) and (25) may vary in each iteration, see e.g. [5, 6].

Applying the augmented Lagrangian method to the split variable formulation in (18), one obtains the Lagrangian function:

$$\begin{aligned} L(\mathbf{x}, \bar{\mathbf{x}}, \mathbf{y}, \lambda) &= \sum_{s \in \mathcal{S}} p_s \left(\mathbf{c}^\top \mathbf{x}_s + \mathbf{q}_s^\top \mathbf{y}_s + \lambda_s^\top (\mathbf{x}_s - \bar{\mathbf{x}}) \right. \\ &\quad \left. + \frac{\rho^{(k)}}{2} \|\mathbf{x}_s - \bar{\mathbf{x}}\|_2^2 \right), \end{aligned} \quad (27)$$

where λ_s is the vector of Lagrange multipliers of the non-anticipativity constraint and ρ is some positive penalty parameter. The last, quadratic term in (27) serves as the penalty function ψ , see e.g. [35].

It seems that this formulation is not really beneficial, since all the scenarios are now connected via $\bar{\mathbf{x}}$ in the cost function, but the trick is to solve the problem iteratively, see [35]. This is done by replacing $\bar{\mathbf{x}}$ with $\bar{\mathbf{x}}^{(k)}$ from the previous iteration. The update of the estimate of the non-anticipative solution $\bar{\mathbf{x}}^{(k+1)}$ is commonly done by calculating its expected value, see e.g. [35, 39]

$$\bar{\mathbf{x}}^{(k+1)} = \sum_{s \in \mathcal{S}} p_s \mathbf{x}_s^{(k+1)}. \quad (28)$$

The Lagrange multiplier update equation (25) in this case becomes

$$\lambda_s^{(k+1)} = \lambda_s^{(k)} + \rho^{(k)} (\mathbf{x}_s^{(k+1)} - \bar{\mathbf{x}}^{(k+1)}). \quad (29)$$

This results in the following augmented Lagrangian function:

$$L(\mathbf{x}, \mathbf{y}, \lambda) = \sum_{s \in \mathcal{S}} p_s \left(\mathbf{c}^\top \mathbf{x}_s + \mathbf{q}_s^\top \mathbf{y}_s + \lambda_s^\top (\mathbf{x}_s - \bar{\mathbf{x}}^{(k-1)}) + \frac{\rho^{(k)}}{2} \|\mathbf{x}_s - \bar{\mathbf{x}}^{(k)}\|_2^2 \right), \quad (30)$$

which is separable by scenario.

The algorithm has converged if all $\mathbf{x}_s^{(k)}$ are close enough to $\bar{\mathbf{x}}^{(k)}$, see. e.g. [39]. A common termination criterion is that the tolerance $\delta^{(k)}$

$$\delta^{(k)} = \sum_{s \in \mathcal{S}} p_s \|\mathbf{x}_s^{(k)} - \bar{\mathbf{x}}^{(k)}\|_2 \quad (31)$$

is smaller than a given positive threshold $\delta_0 > 0$.

The complete classical progressive hedging algorithm is stated in Algorithm 1.

Algorithm 1: Classical Progressive Hedging Algorithm

- 1 Set the iteration counter $k = 0$, the initial penalty parameter $\rho^{(0)} = \rho_0$
- 2 Solve the scenario subproblems $\forall s \in \mathcal{S}$:

$$\mathbf{x}_s^{(0)}, \mathbf{y}_s^{(0)} = \underset{(\mathbf{x}_s, \mathbf{y}_s) \in \mathcal{C}_s}{\operatorname{argmin}} \quad \mathbf{c}^\top \mathbf{x}_s + \mathbf{q}_s^\top \mathbf{y}_s \quad (32)$$

with

$$\mathcal{C}_s = \{\mathbf{x}_s, \mathbf{y}_s \mid \mathbf{A} \mathbf{x}_s = \mathbf{b}, \mathbf{T}_s \mathbf{x}_s + \mathbf{W}_s \mathbf{y}_s = \mathbf{h}_s, \mathbf{x}_s \in \mathcal{X}, \mathbf{y}_s \in \mathcal{Y}\} \quad (33)$$

- 3 Compute the initial solution $\bar{\mathbf{x}}^{(0)} = \sum_{s \in \mathcal{S}} p_s \mathbf{x}_s^{(0)}$ and the initial duals

$$\lambda_s^{(0)} = \rho^{(0)} (\mathbf{x}_s^{(0)} - \bar{\mathbf{x}}^{(0)}), \quad \forall s \in \mathcal{S}$$

4 **repeat**

- 5 Solve the augmented scenario subproblems for all $s \in \mathcal{S}$:

$$\mathbf{x}_s^{(k+1)}, \mathbf{y}_s^{(k+1)} = \underset{(\mathbf{x}_s, \mathbf{y}_s) \in \mathcal{C}_s}{\operatorname{argmin}} \quad \mathbf{c}^\top \mathbf{x}_s + \mathbf{q}_s^\top \mathbf{y}_s + \left(\lambda_s^{(k)} \right)^\top \mathbf{x}_s + \frac{\rho^{(k)}}{2} \|\mathbf{x}_s - \bar{\mathbf{x}}^{(k)}\|_2^2 \quad (34)$$

- 6 Update the solution $\bar{\mathbf{x}}^{(k+1)} = \sum_{s \in \mathcal{S}} p_s \mathbf{x}_s^{(k+1)}$

- 7 Update the duals $\lambda_s^{(k+1)} = \lambda_s^{(k)} + \rho^{(k)} (\mathbf{x}_s^{(k+1)} - \bar{\mathbf{x}}^{(k+1)})$, $\forall s \in \mathcal{S}$

- 8 $k = k + 1$

- 9 **until** $\sum_{s \in \mathcal{S}} p_s \|\mathbf{x}_s^{(k+1)} - \bar{\mathbf{x}}^{(k+1)}\|_2 < \delta_0$ **or** $k > k_{max}$;
-

Steps 2 and 5 of the algorithm can be solved in *parallel*, which can be exploited in implementations of the algorithm. The practical implementation of the PH algorithm is comparatively easy, since one essentially has to solve a deterministic problem for each scenario (Algorithm 1 step 2). Additionally, one has to solve modified versions of the deterministic problem for each scenario (Algorithm 1 step 5). However, these *augmented subproblems* are *non-linear*, incorporating quadratic terms in the objective function. Hence, in the mixed-integer case a solver for mixed-integer quadratic programmes (MIQP) is required for the classical PH algorithm implementation. However, the option to reuse parts or even all of the deterministic implementation makes the algorithm an attractive variant for solving stochastic programmes in practice. An additional benefit, that is not further investigated in this paper, is the fact that the algorithm can be very easily extended to multistage stochastic programmes, see e.g. [28].

If the problem is convex, the PH algorithm experiences *linear convergence*, approximately meaning that the tolerance parameter $\delta^{(k)}$ from (31) is multiplied by a positive constant smaller than one in each iteration, see e.g. [39]. Unfortunately, the classical progressive hedging algorithm (Algorithm 1) might not converge in the non-convex mixed-integer case, see e.g. [12, 28]. Instead, it might experience *cyclic behaviour*, see e.g. [39]. Some heuristic additions to the algorithm have been proposed to cope with this problem, see e.g. [39]. In any case, the algorithm is nevertheless considered as a *good* heuristic for stochastic MILP problems, see e.g. [12, 28, 39, 40].

3.1 PH for mixed-integer problems

Since in the mixed-integer case, the PH algorithm is just a heuristic, [19] proposes to assess the quality of the solutions by computing *lower bounds* z_{lb} of the objective function z^* of the stochastic programme, using the Lagrange multipliers $\lambda_s^{(k)}$ obtained by the PH algorithm in each iteration. However, the Lagrangian dual of (18):

$$\mathcal{D}(\lambda_s^{(k)}) = \min_{(\mathbf{x}_s, \mathbf{y}_s) \in \mathcal{C}_s} \mathbf{c}^\top \mathbf{x}_s + \mathbf{q}_s^\top \mathbf{y}_s + (\lambda_s^{(k)})^\top \mathbf{x}_s \quad (35)$$

where

$$\mathcal{C}_s = \{\mathbf{x}_s, \mathbf{y}_s \mid \mathbf{A}\mathbf{x}_s = \mathbf{b}, \mathbf{T}_s\mathbf{x}_s + \mathbf{W}_s\mathbf{y}_s = \mathbf{h}_s, \mathbf{x}_s \in \mathcal{X}, \mathbf{y}_s \in \mathcal{Y}\} \quad (36)$$

has to be solved for each subproblem and each iteration in order to calculate the lower bound $\mathcal{D}(\lambda)$:

$$\mathcal{D}(\lambda^{(k)}) = \sum_{s \in \mathcal{S}} p_s \mathcal{D}(\lambda_s^{(k)}). \quad (37)$$

This increases the computational burden substantially, since the complexity of (35)–(37) is comparable with the subproblems solved by PH. In order for (37) to be a lower bound of the problem, $\sum_{s \in \mathcal{S}} p_s \lambda_s = \mathbf{0}$ has to hold element-wise. However, if

any heuristic computation is involved in (28) or (29), this no longer holds and hence the lower bound can not be computed in the way presented by [19].

In [2] PH is used within a *branch-and-bound* framework, enabling it to be used for mixed-integer problems. In contrast to the use of branch-and-bound for solving the augmented subproblems, in this case the branching occurs on $\bar{x}$. At each node of the branch-and-bound tree, classical PH is used to solve the *nodal problem*, which itself is a mixed-integer SP. The performance of the algorithm is demonstrated first on a two-stage problem with purely binary first-stage variables, and second for a two-stage problem with mixed-integer first-stage variables. For the first example the algorithm achieves solutions with negligible optimality gaps in considerably less time than the classical PH algorithm. The example with mixed-integer first-stage variables proved to be challenging for the algorithm, since the usefulness of branch-and-bound is limited in this case. The authors employ a rounding heuristic for all integer first-stage variables in this case. The algorithm was able to reduce the optimality gap very fast but was very slow to converge thereafter, which is typical for branch-and-bound algorithms. Improving the underlying PH algorithm may enhance this convergence speed.

In the literature some heuristic measures have been proposed in order to manage the non-convex mixed-integer case. For a class of stochastic resource allocation problems, the idea of fixing *converged* integer variables, detecting and handling cyclic behaviour is described in [39]. A combination of PH and *tabu search* for mixed-integer $\{0,1\}$ stochastic problems is proposed in [28]. Heuristic measures for PH in the context of stochastic network design problems are described in [12]. As mentioned in Sect. 2.3, some decomposition algorithms for specialized versions of the two-stage problem exist, see e.g. [37]. Since the stochastic programme used in the considered optimization-based EMS is mixed-integer in both the first and second stage, these specialized decomposition algorithms cannot be used.

In [17] it is shown that for applying augmented Lagrangian functions in the context of mixed-integer programmes, *any norm* proofs to be a valid penalty function, see [17, Thm. 4]. This is not true for squared norms, like the squared L_2 -norm that is used in the classical progressive hedging algorithm. Building on the theoretical results from [17], in this paper the use of the L_1 and L_∞ -norm as penalty functions instead of the squared L_2 -norm-based penalty function from the classical PH algorithm is investigated.

3.2 Proposed modifications to the PH Algorithm in the mixed-integer case

In order to incorporate the L_1 and L_∞ -norm into the algorithm an adapted multiplier update step is proposed. Additionally a heuristic extension of the aggregation step and an adaptive penalty parameter update scheme from [9] is investigated. Each modification to the PH algorithm is explained separately in the following.

An important implementation issue of penalty-based methods, like PH, is the normalization of constraints, this will be explained in Sect. 3.2.1. In Sect. 3.2.2 it is shown how the L_1 and L_∞ -norm can be used as penalty functions in the PH algorithm. In Sect. 3.2.3 a heuristic extension to the aggregation step is proposed. Finally in Sect. 3.2.4 a heuristic penalty parameter update step from the literature is considered.

3.2.1 Constraint normalization

For penalty methods, a common implementation issue is the need for scaling or normalizing the constraints $\mathbf{g}(\mathbf{x})$ in Eq. (20). This ensures that no single constraint has a disproportional effect on the objective function. In order to avoid such issues in the context of the PH algorithm a simple normalization scheme for the non-anticipativity constraints is proposed.

An intuitive choice for a scaling parameter α is simply the absolute maximum range for each constraint g_i

$$\alpha_i = \max \{ |g_i(\mathbf{x})_{\max}|, |g_i(\mathbf{x})_{\min}| \}, \quad \forall \mathbf{x} \in \mathcal{X}. \quad (38)$$

With this scaling parameter the normalized constraints $\mathbf{h}(\mathbf{x})$ can be computed as

$$\mathbf{h}(\mathbf{x}) = \text{diag}(\alpha)^{-1} \mathbf{g}(\mathbf{x}). \quad (39)$$

The augmented Lagrangian function (22) and the update step for the Lagrange multipliers (25) can then be written as

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda^T \mathbf{h}(\mathbf{x}) + \rho^{(k)} \psi(\mathbf{h}(\mathbf{x})), \quad (40)$$

and

$$\lambda^{(k+1)} = \lambda^{(k)} + \rho^{(k)} \partial \psi(\mathbf{h}(\mathbf{x}^{(k+1)})). \quad (41)$$

3.2.2 Choice of the penalty function

In the standard PH algorithm, the non-anticipativity constraints are relaxed using a quadratic penalty function of the form

$$\psi(\mathbf{u}) = \frac{1}{2} \sum_i u_i^2. \quad (42)$$

It is essentially the squared L_2 -norm multiplied by one half, resulting in a *quadratic* term in the cost function of the augmented subproblems.

However, for solving mixed-integer problems with an augmented Lagrangian [17] suggests to use different penalty functions. In [17, Thm. 4] it is stated that a, in the mixed-integer case, applicable penalty function has to fulfil:

$$\psi(\mathbf{0}) = 0 \quad (43a)$$

$$\psi(\mathbf{u}) \geq \zeta > 0, \quad \mathbf{u} \notin V \quad (43b)$$

$$\psi(\mathbf{u}) \geq \gamma \|\mathbf{u}\|_\infty, \quad \mathbf{u} \in V \quad (43c)$$

where V is a neighbourhood of $\mathbf{0}$ and $\zeta, \gamma > 0$ are positive scalars. It is shown that if the penalty function $\psi(\mathbf{u})$ is *any* norm, it results in a *sharp augmented Lagrangian*,

which is beneficial in the mixed-integer case. It is easy to see that (43) does not hold for squared norms, as [17] states. In [17] an example is shown where the duality gap cannot be closed when using a quadratic penalty function for a MILP with a positive finite cost coefficient sequence $\rho^{(k)}$, but other norms are able to do so. In [17, Thm. 5] the class of penalty functions leading to a sharp augmented Lagrangian is extended to penalty functions of the form $\|\mathbf{u}\|^r$, where $0 < r < 1$ and $\|\cdot\|$ is again any norm.

Building on this idea, in this paper two different penalty functions for the PH algorithm are investigated. The different functions only influence Algorithm 1 by changes necessary in the objective function of the augmented subproblems (Algorithm 1 step 5) and in the update step of the duals (Algorithm 1 step 7). In order to stay within the realm of mixed-integer linear programming the only norms that can be used are the L_1 -norm

$$\psi(\mathbf{u}) = \sum_i |u_i|. \quad (44)$$

and the L_∞ -norm

$$\psi(\mathbf{u}) = \max(|u_1|, \dots, |u_N|). \quad (45)$$

Since the L_1 -norm is not continuously differentiable the Lagrange multiplier update step (41) is proposed to be done in the following way:

$$\lambda^{(k+1)} = \lambda^{(k)} + \rho^{(k)} \overline{\text{sign}}(\mathbf{h}(\mathbf{x}^{(k+1)})). \quad (46)$$

Here $\overline{\text{sign}}$ is a *smooth approximation* of the signum function, which is considered to be applied element-wise on each constraint in $\mathbf{h}(\mathbf{x})$:

$$\overline{\text{sign}}(u) = \frac{u}{\sqrt{u^2 + \varepsilon^2}}, \quad (47)$$

where $\varepsilon > 0$ is a small positive constant. Without this smooth approximation, the algorithm was not able to achieve convergence in some of the test instances.

The L_∞ -norm, on the other hand, is highly *non-smooth*. Hence, using the *smooth approximation* $\overline{\text{max}}(\mathbf{u})$ is proposed:

$$\overline{\text{max}}(\mathbf{u}) = \frac{\sum_{i=1}^N u_i e^{\alpha u_i}}{\sum_{i=1}^N e^{\alpha u_i}}, \quad (48)$$

where $\alpha > 0$ is a positive constant. Since for large values of α , (48) will be numerically unstable, moderate values of 1 to 10 should be used. Using (41) and (48), the dual update step is proposed to be done in the following way:

$$\lambda^{(k+1)} = \lambda^{(k)} + \rho^{(k)} \begin{pmatrix} \frac{e^{\alpha|h_1(\mathbf{x})|}}{\sum_{i=1}^N e^{\alpha|h_i(\mathbf{x})|}} (1 + \alpha(|h_1(\mathbf{x})| - \overline{\max(|\mathbf{h}(\mathbf{x})|)}) \\ \vdots \\ \frac{e^{\alpha|h_N(\mathbf{x})|}}{\sum_{i=1}^N e^{\alpha|h_i(\mathbf{x})|}} (1 + \alpha(|h_N(\mathbf{x})| - \overline{\max(|\mathbf{h}(\mathbf{x})|)}) \end{pmatrix} \text{sign}(\mathbf{h}(\mathbf{x})), \quad (49)$$

where the sign function is again considered to be applied element-wise and $\mathbf{x}$ is $\mathbf{x}^{(k+1)}$. Again, without this smooth approximation, the algorithm was not able to achieve convergence in some of the test instances.

In [40] the authors eliminate the need for a mixed-integer quadratic solver by approximating the squared L_2 -norm by a piecewise affine function. Since the function is minimized and concave, this can be done very easily with *cuts* from below, without the need for additional integer variables. Normalizing the constraints as in Sect. 3.2.1 is even more important in this context. If the piece-wise affine (PWA) approximation is done in a way such that $\psi(\mathbf{0}) = 0$, it is easy to see that the penalty function now fulfils (43) and is hence applicable for mixed-integer problems. This PWA approximation-based version of the PH algorithm will be included in the numerical study.

3.2.3 Computation of $\bar{\mathbf{x}}$ for discrete decision variables

If mixed-integer problems are considered, one problem that arises is the computation of (28) if $\mathbf{x} \in \mathbb{R}^n \times \mathbb{Z}^m$. Since the interpretation of a *weighted mean* for integer variables is not obvious, a rounding scheme towards the nearest integer value is proposed. Hence the computation of the non-anticipative solution becomes:

$$\bar{\mathbf{x}}^{(k)} = \sum_{s \in S} p_s \mathbf{x}_s^{(k)} \quad (50a)$$

$$\bar{x}_i^{(k)} = \hat{x}_i, \quad \text{if } x_{s,i}^{(k)} \in \mathbb{Z} \text{ and } |x_{s,i}^{(k)} - \hat{x}| < \kappa, \quad (50b)$$

where $\hat{x}_i = \left\lfloor \sum_{s \in S} p_s x_{s,i}^{(k)} \right\rfloor$ and $0 < \kappa \leq 1/2$. The idea is that if the probability weighted mean of an integer value is close enough to an integer value it may be rounded towards it. The experiments that were conducted for this paper showed that this heuristic has a positive impact on the convergence speed.

3.2.4 Adaptation of the penalty parameter

In the literature it is often stated that the PH algorithm is sensitive to the choice of penalty parameter, see e.g. [2, 12, 39]. In the context of the augmented Lagrangian method, the penalty parameter $\rho^{(k)}$ is often increased in each iteration, see e.g. [5, 6, 12]. However, increasing $\rho^{(k)}$ too much results in an ill-conditioned problem, see e.g. [6]. In [39] the penalty parameter ρ is proposed to be computed element-wise for each element of $\mathbf{x}$. The authors go as far as that ρ is no longer a parameter of the algorithm

but is computed from the problem itself. However, the penalty parameter is computed once in the beginning and is otherwise kept constant during the PH iterations. In contrast, an adaptive penalty parameter update scheme can be found in [9]. The idea is that primal and dual progress of the algorithm should be balanced

$$\rho^{(k+1)} = \begin{cases} \rho^{(k)} \tau_{\text{incr}} & \text{if } \|\mathbf{r}^{(k)}\|_2 > \mu \|\mathbf{s}^{(k)}\|_2 \\ \rho^{(k)} 1/\tau_{\text{decr}} & \text{if } \|\mathbf{s}^{(k)}\|_2 > \mu \|\mathbf{r}^{(k)}\|_2 \\ \rho^{(k)} & \text{otherwise,} \end{cases} \quad (51)$$

where $\mathbf{r}^{(k)}$ is the vector of *primal residuals* and $\mathbf{s}^{(k)}$ is the vector of *dual residuals*. The respective primal and dual residuals can be computed as:

$$\mathbf{r}^{(k)} = \begin{pmatrix} \mathbf{x}_1^{(k)} - \bar{\mathbf{x}}^{(k)} \\ \vdots \\ \mathbf{x}_{|S|}^{(k)} - \bar{\mathbf{x}}^{(k)} \end{pmatrix}, \quad \mathbf{s}^{(k)} = \begin{pmatrix} -\rho(\bar{\mathbf{x}}^{(k)} - \bar{\mathbf{x}}^{(k-1)}) \\ \vdots \\ -\rho(\bar{\mathbf{x}}^{(k)} - \bar{\mathbf{x}}^{(k-1)}) \end{pmatrix}. \quad (52)$$

The positive constants τ_{incr} , τ_{decr} and μ are parameters of the scheme. The update scheme is intended for the ADMM algorithm, but since the PH algorithm is just a variant of consensus-based ADMM, see [9], it is reasonable to use it in this context as well.

3.2.5 Convergence check

The classical PH algorithm is terminated if the residual (31) is small or if the iteration limit k_{max} is reached. However, this only checks primal convergence, which can be easily achieved by choosing a high value for the penalty parameter ρ . In [9] it is suggested to terminate the algorithm if the norm of both the *primal* and the *dual residuals* are small

$$\|\mathbf{r}^{(k)}\|_2 < \epsilon_{\text{pri}} \quad \text{and} \quad \|\mathbf{s}^{(k)}\|_2 < \epsilon_{\text{dual}}, \quad (53)$$

Hence taking dual and primal convergence in consideration. The convergence thresholds $\epsilon_{\text{pri}} > 0$ and $\epsilon_{\text{dual}} > 0$ are design parameters.

4 Numerical study

In this section four different versions of the PH algorithm are investigated on three examples of multi energy systems for real-time EMS. All versions use the proposed constraint normalization from Sect. 3.2.1, the rounding scheme from Sect. 3.2.3 and the adaptation of the penalty parameter $\rho^{(k)}$ from Sect. 3.2.4. The versions only differ in the penalty function that is used and therefore differ in the augmented subproblems as well as in the update step of the duals, as explained in Sect. 3.2.2. In addition to implementations with the squared L_2 -norm, the L_1 -norm and the L_∞ -norm, an implementation using a piece-wise affine (PWA) approximation of the squared L_2

-norm is used for comparison. Apart from the PWA approximation in the augmented subproblems, this version is exactly the same as for the squared L_2 -norm penalty function.

The four different implementations are compared with respect to their *convergence time* and *solution quality*. The quality is assessed in terms of the absolute error with respect to the solution obtained by solving the extensive form of the problem. After convergence is achieved, the time limit or iteration limit is reached, rounding of all integer first-stage variables is enforced and a final optimization step is performed in order to compute the real objective function value. This last step is included in the convergence time evaluation.

In the simulations that have been performed for this paper it has been observed that, depending on the problem data, the computational time until convergence is quite different. Hence, the evaluation will consist of 100 runs of the same problems but with different data. The problems are essentially solved for data of 100 randomly chosen days of a year.

All results shown were obtained using a standard laptop with an Intel Core i5 and 8 GB of memory and Gurobi 9.0.1 [21] as MILP and MIQP solver. The algorithms were implemented in Julia [7] and the optimization problems were algebraically modelled using JuMP [13]. The considered problems have 50 scenarios each, and a time limit of 900 s for each run of the algorithms was imposed together with an iteration limit of 40. The primal and dual convergence thresholds ϵ_{pri} and ϵ_{dual} have been set to 10^{-2} and 10^{-3} . For the adaptive penalty parameter update strategy the following values have been chosen: $\mu = 10$, $\tau_{\text{incr}} = 2$ and $\tau_{\text{decr}} = 2$. The initial penalty parameter ρ_0 has been set to 1.

Next the four considered implementations of the PH algorithm are tested on three different examples of multi energy systems for real-time EMS: a small district heating network with one single gas engine and a backup gas boiler (see Sect. 4.1), a scaled version with five different gas engines (see Sect. 4.2) and a scaled version with eight identical gas engines (see Sect. 4.3).

4.1 Small district heating network

The first example models a real-world installation, comprising a *gas engine*, a *gas boiler*, a *solar thermal panel* and a *thermal energy storage tank* all provide heat for a small district heating network, which is represented by one conglomerate *heat consumer*. The gas engine is modelled as a CHP plant, see Sect. 2.1, producing up to 1 MW thermal power and up to 900 kW electrical power. It has maintenance costs of 7 euro per hour of run time and a limit on the startups per day of four. The gas boiler serves as a backup with an additional 1 MW of thermal power. Both, the gas motor and gas boiler have *minimal up* and *down time* constraints of one hour and a minimal commitment of 15%. A *gas source* models the connection to the public gas network and a *power sink* models a connection to electricity network where the electrical power produced by the gas motor is sold at a fixed price. The thermal energy storage has a volume of 210 m³ and the solar thermal panel a maximal yield of 500 kW. The

Fig. 2 Performance of the algorithms for the small district heating network example. The two *box-plots* show the accuracy of the solution with respect to the solution obtained by solving the extensive form, as well as the time it took the algorithms to compute the solution. The box plot shows the upper and lower quartile i.e. 50 % of the data lies within the box. The *median* is indicated with a gray line. The so-called whiskers show the maximum and minimum values of all data points that lie within the upper and lower quartile plus respectively minus 1.5 times the box length (colour figure online)

scenario generation and reduction was performed as explained in Sect. 2.3.1 and the forecast method from [33] was used for obtaining the point-forecasts for the heat demand and the method from [38] for the point-forecasts of the solar thermal yield. In both cases using 60 days of historical measurements. The resulting SP comprises 6 first-stage variables, two of which are binary. As explained in Sect. 2.3.2, these represent the first time instance of the commitment $u_{\text{on/off},t}$, dispatch $u_{d,t}$ and setpoint z_t control variables of the gas motor and gas boiler, respectively. The statistical evaluation of 100 runs of randomly chosen days of a year are shown in Fig. 2.

The depicted *box-plots* show that the quality of the obtained solutions with respect to the extensive form solution is quite good. The median absolute error is below 0.5 % for each implementation. While the PWA approximated version has the smallest absolute error, the median computational time is the longest. Although the chosen example seems simple at a first glance, solving it via its extensive form proved to be challenging and time consuming—almost every run took the full 900 s, as can be seen by the median value. This clearly shows the necessity of decomposing mixed-integer stochastic programmes in order for them to be used within real-time EMS. The L_1 and L_∞ -norm-based implementations show a clear improvement in the solution time with almost the same quality of the solution, hence, improving the real-time capability of the algorithm.

Fig. 3 Performance of the algorithms for the EMS example with several different producers

4.2 EMS with multiple different producers

In order to test the performance on an instance with more first stage variables, the second considered example is a scaled version of the first example. In addition to the gas boiler, five different gas engines are used. The resulting SP comprises 15 first-stage variables, five of which are binary. The maximal electrical power output of the each gas engines is: 900 kW, 1.2 MW, 1.5 MW, 2 MW, and 2.4 MW are used. The maximal thermal output of the engines was scaled accordingly: 1 MW, 1.3 MW, 1.6 MW, 2 MW, and 2.2 MW. The thermal load has been increased by a factor of 4.5 to match the increased production. The thermal energy storage and the solar thermal panel capacity have also been increased by a factor of 4.5. The constraints on each engine are the same as for the simple example.

The statistical evaluation of the conducted 100 runs is shown in figure 3.

The box-plots show that the median absolute error of the obtained solutions is again below 0.5 %. The L_∞ -norm-based and PWA approximated versions have the lowest median absolute error, while the L_1 -norm-based version performed worst with the highest variance of the absolute error. The median time it takes the different implementations to reach convergence is below 60 s. While all PH implementations have a similar median time, the L_1 -norm-based implementation has the lowest variance. The squared L_2 -norm implementation seems to perform worse on some test instances, hence having a larger variance in solution time. However, this is suspected, since solving mixed-integer quadratic programmes is difficult and in general slower than solving mixed-integer linear programmes. The PWA approximated version of this implementation and the L_∞ -norm-based version have a higher variance on the convergence time than the L_1 -norm-based implementation. In contrast to the first example even the EF time is below the time limit of 900 s.

Fig. 4 Performance of the algorithms for the EMS example with several identical producers

4.3 EMS with several identical producers

The third example is similar to the second, except that eight identical engines with a maximal thermal power output of 1 MW and a maximal electrical output of 900 kW are used in addition to the gas boiler, used in the first and second example. It is reasonable to expect that a consensus is harder to achieve in this case. As for the second example, the resulting SP comprises 24 first-stage variables, eight of which are binary. The thermal load, the solar collector and the thermal energy storage are scaled as in the second example.

The statistical evaluation of the conducted 100 runs is shown in figure 4.

The box-plots show that the median absolute error of the obtained solutions is very similar for all considered implementations and again below 0.5 %. Since mixed-integer quadratic programmes have to be solved in the case of the square L_2 -norm-based version, it is expected to be slower, which can be seen in Fig. 4. In terms of convergence time, the extensive form took by far the longest.

The presented results show that the L_1 and L_∞ -norm can be used in the progressive hedging algorithm, reducing the solution time while achieving similar solution qualities. However, incorporating the L_1 or L_∞ -norm into PH, renders the objective function of the augmented subproblems (25) non-continuously differentiable. This proved to be a challenging obstacle since the *subgradient scheme*, used for the Lagrange multiplier update, can no longer be explicitly stated and a heuristic solution had to be used. The PWA approximated version of the L_2 -norm is easy to implement and performed better than the classical PH algorithm in our tests. Not only in solution time, this is expected since no mixed-integer quadratic programmes have to be solved but also in terms of the quality of the solution. According to [17] and as explained in Sect. 3.2.2, if the penalty function satisfies (43), resulting in a sharp Lagrangian, it is guaranteed that there exists a finite penalty ρ such that there is no

duality gap in the mixed-integer case. This is true for all norms, however, as shown also for the PWA approximation of the squared L_2 -norm, if done correctly.

5 Conclusion and outlook

The obtained results in this paper show the practical relevance of the PH-algorithm for real-time EMS problems where tight time constraints have to be met and sub-optimal solutions are acceptable. Additionally the relatively simple implementation and the ability to parallelize certain parts of the algorithm are attractive features from a practical point of view. Since an EMS is a supervisory controller and depends on the underlying low-level-controllers of the producers, it is not necessary to obtain solutions with very tight tolerances. Building on existing theoretical results; that any norm results in a sharp Lagrangian, which is beneficial in the mixed-integer case, the paper demonstrates that the L_1 -norm and L_∞ -norm can be used as penalty functions in the progressive hedging algorithm, leading to an improved convergence speed in the mixed-integer case. It is also shown that a piece-wise linear approximation of the squared L_2 -norm, as used in the classical PH algorithm, should be favoured in the mixed-integer case. Additionally this eliminates the necessity of solving *mixed-integer quadratic programmes*.

In this paper only the two-stage case was investigated, but the results are expected to be similar in the multi-stage case. In order to decompose larger problems even further, not only a scenario-wise decomposition, but also a prosumer-wise decomposition is expected to be beneficial. It may also be possible to split the considered multi-energy system into just slightly coupled parts, where in turn the connecting balancing equations could be relaxed. The idea would be essentially the same since the balance equations that represent the connections of individual assets are also simple equality constraints and can thus be handled with the same method as the non-anticipativity constraint.

Acknowledgements Open access funding provided by Graz University of Technology. The funding has been received from Forschungsförderungsgesellschaft with Grant No. 869341, Österreichische Forschungsförderungsgesellschaft with Grant No. 868852.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Achterberg, T., Wunderling, R.: Mixed integer programming: analyzing 12 years of progress. In: facets of combinatorial optimization, vol. 9783642381, pp. 449–481. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
2. Atakan, S., Sen, S.: a progressive hedging based branch-and-bound algorithm for mixed-integer stochastic programs **15**, (2018). <https://doi.org/10.1007/s10287-018-0311-3>
3. Beale, E.M.: Branch and bound methods for mathematical programming systems. *Ann. Discrete Math.* **5**(C), 201–219 (1979). [https://doi.org/10.1016/S0167-5060\(08\)70351-0](https://doi.org/10.1016/S0167-5060(08)70351-0)
4. Bemporad, A., Morari, M.: Control of systems integrating logic, dynamics, and constraints. *Automatica* **35**(3), 407–427 (1999). [https://doi.org/10.1016/S0005-1098\(98\)00178-2](https://doi.org/10.1016/S0005-1098(98)00178-2)
5. Bertsekas, D.P.: Multiplier methods: a survey. *Automatica* **12**, 133–145 (1976)
6. Bertsekas, D.P.: Chapter 2-The method of multipliers for equality constrained problems. In: constrained optimization and lagrange multiplier methods (Academic Press, 1982), pp. 95–157. <https://doi.org/10.1016/B978-0-12-093480-5.50006-4>
7. Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B.: Julia: a fresh approach to numerical computing. *SIAM Rev.* **59**(1), 65–98 (2017)
8. Bischi, A., Taccari, L., Martelli, E., Amaldi, E., Manzolini, G., Silva, P., Campanari, S., Macchi, E.: A detailed MILP optimization model for combined cooling, heat and power system operation planning. *Energy* (2014). <https://doi.org/10.1016/j.energy.2014.02.042>
9. Boyd, S., Parikh, N., Chu, E., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.* **3**(1), 1–122 (2010). <https://doi.org/10.1561/22000000016>
10. Carøe, C.C., Schultz, R.: Dual decomposition in stochastic integer programming. *Oper. Res. Lett.* **24**(1–2), 37–45 (1999). <https://doi.org/10.1016/j.orl.2013.02.003>
11. Carøe, C.C., Tind, J.: L-shaped decomposition of two-stage stochastic programs with integer recourse. *Math. Program.* **83**(1–3), 451–464 (2007). <https://doi.org/10.1007/bf02680570>
12. Crainic, T.G., Fu, X., Gendreau, M., Rei, W., Wallace, S.W.: Progressive hedging-based metaheuristics for stochastic network design. *Networks* **58**(2), 114–124 (2011). <https://doi.org/10.1002/net.20456>
13. Dunning, I., Huchette, J., Lubin, M.: Jump: a modeling language for mathematical optimization. *SIAM Rev.* **59**(2), 295–320 (2017). <https://doi.org/10.1137/15M1020575>
14. Dvorkin, Y., Wang, Y., Pandzic, H., Kirschen, D.: Comparison of scenario reduction techniques for the stochastic unit commitment. In: 2014 IEEE PES General Meeting|Conference and Exposition, vol. 2014, pp. 1–5. IEEE (2014). <https://doi.org/10.1109/PESGM.2014.6939042>
15. Everitt, B., Skrondal, A.: The cambridge dictionary of statistics, vol. 4, 4 edn. Cambridge University Press, Cambridge (2010). <https://doi.org/10.1017/CBO9781107415324.004>
16. Fakhri, A., Ghatee, M., Fragkogios, A., Saharidis, G.K.: Benders decomposition with integer subproblem. *Expert Syst. Appl.* **89**, 20–30 (2017). <https://doi.org/10.1016/j.eswa.2017.07.017>
17. Feizollahi, M.J., Ahmed, S., Sun, A.: Exact augmented Lagrangian duality for mixed integer linear programming. *Math. Program.* **161**(1–2), 365–387 (2017). <https://doi.org/10.1007/s10107-016-1012-8>
18. Gabrielli, P., Gazzani, M., Martelli, E., Mazzotti, M.: Optimal design of multi-energy systems with seasonal storage. *Appl. Energy* **219**(2017), 408–424 (2018). <https://doi.org/10.1016/j.apene.2017.07.142>
19. Gade, D., Hackebeil, G., Ryan, S.M., Watson, J.P., Wets, R.J., Woodruff, D.L.: Obtaining lower bounds from the progressive hedging algorithm for stochastic mixed-integer programs. *Math. Program.* **157**(1), 47–67 (2016). <https://doi.org/10.1007/s10107-016-1000-z>
20. Guo, G., Hackebeil, G., Ryan, S.M., Watson, J.P., Woodruff, D.L.: Integration of progressive hedging and dual decomposition in stochastic integer programs. *Operat. Res. Lett.* **43**(3), 311–316 (2015). <https://doi.org/10.1016/j.orl.2015.03.008>
21. Gurobi Optimization, L.: Gurobi Optimizer Reference Manual (2019). <https://www.gurobi.com>. Accessed 8 May 2020
22. IBM, C.O.: IBM ILOG CPLEX User Manual (2019). <https://www.cplex.com>. Accessed 8 May 2020
23. Kall, P., Wallace, S.W.: Stochastic programming. Wiley, Chichester (1994)

24. Kim, K., Zavala, V.M.: Algorithmic innovations and software for the dual decomposition method applied to stochastic mixed-integer programs. *Math. Program. Comput.* **10**(2), 225–266 (2018). <https://doi.org/10.1007/s12532-017-0128-z>
25. Klotz, E., Newman, A.M.: Practical guidelines for solving difficult mixed integer linear programs (2013). <https://doi.org/10.1016/j.sorms.2012.12.001>
26. Lemaréchal, C.: Lagrangian relaxation. In: Jünger, M., Naddef, D. (eds.) *Lecture notes in computer science*, vol. 2241, pp. 112–156. Springer, Berlin, Heidelberg (2001)
27. Li, Z., Zang, C., Zeng, P., Yu, H., Zhongwen, L., Chuanzhi, Z., Peng, Z., Haibin, Y.: Combined two-stage stochastic programming and receding horizon control strategy for microgrid energy management considering uncertainty. *Energies* **9**(7) (2016). <https://doi.org/10.3390/en9070499>
28. Lokketangen, A., Woodruff, D.: Progressive hedging and tabu search applied to mixed integer (0,1) multistage stochastic programming. *J. Heurist.* **2**(2), 111–128 (2004). <https://doi.org/10.1007/bf00247208>
29. Moghimi, M., Leskarac, D., Bennett, C., Lu, J., Stegen, S.: Rule-based energy management system in an experimental microgrid with the presence of time of use tariffs. *matec web of conferences* **70** (2016). <https://doi.org/10.1051/mateconf/20167010011>
30. Morales-Espana, G., Correa-Posada, C.M., Ramos, A.: Tight and compact MIP formulation of configuration-based combined-cycle units. *IEEE Trans. Power Syst.* **31**(2), 1350–1359 (2016). <https://doi.org/10.1109/TPWRS.2015.2425833>
31. Morales-España, G., Latorre, J.M., Ramos, A.: Tight and compact MILP formulation for the thermal unit commitment problem. *IEEE Trans. Power Syst.* **28**(4), 4897–4908 (2013). <https://doi.org/10.1109/TPWRS.2013.2251373>
32. Moser, A., Muschick, D., Göllles, M., Nageler, P., Schranzhofer, H., Mach, T., Ribas Tugores, C., Leusbrock, I., Stark, S., Lackner, F., Hofer, A.: A MILP-based modular energy management system for urban multi-energy systems: performance and sensitivity analysis. *Appl. Energy* **261**(2020), 114342 (2020). <https://doi.org/10.1016/j.apenergy.2019.114342>
33. Nigitz, T., Göllles, M.: A generally applicable, simple and adaptive forecasting method for the short-term heat load of consumers. *Appl Energy* **241**(2018), 73–81 (2019). <https://doi.org/10.1016/j.apenergy.2019.03.012>
34. Ottesen, S.Ø., Tomasgard, A., Fleten, S.E.: Prosumer bidding and scheduling in electricity markets. *Energy* **94**, 828–843 (2016). <https://doi.org/10.1016/j.energy.2015.11.047>
35. Rockafellar, R.T., Wets, R.J.B.: Scenarios and Policy aggregation in optimization under uncertainty. *Math. Oper Res* **16**(1), 119–147 (1991)
36. Rujeerapaiboon, N., Schindler, K., Kuhn, D., Wiesemann, W.: Scenario reduction revisited: fundamental limits and guarantees. *Math. Program.* **41**, 1–36 (2018). <https://doi.org/10.1007/s10107-018-1269-1>
37. Sen, S.: Algorithms for stochastic mixed-integer programming models. *Handb. Oper. Res. Manag. Sci.* **12**(C), 515–558 (2005). [https://doi.org/10.1016/S0927-0507\(05\)12009-X](https://doi.org/10.1016/S0927-0507(05)12009-X)
38. Unterberger, V., Nigitz, T., Luzzu, M., Muschick, D., Göllles, M.: Adaptive methods for energy forecasting of production and demand of solar-assisted heating systems. In: *Theory and applications of time series analysis* (Springer International Publishing 2019), pp. 287–299. https://doi.org/10.1007/978-3-030-26036-1_20
39. Watson, J.P., Woodruff, D.L.: Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems. *Comput. Manag. Sci.* **8**(4), 355–370 (2011). <https://doi.org/10.1007/s10287-010-0125-4>
40. Watson, J.P., Woodruff, D.L., Hart, W.E.: PySP: modeling and solving stochastic programs in Python. *Math. Program. Comput.* **4**(2), 109–149 (2012). <https://doi.org/10.1007/s12532-012-0036-1>