A Stabilised Scenario Decomposition Algorithm Applied to Stochastic Unit Commitment Problems

Tim Schulze[a,*], Andreas Grothey[a], Ken McKinnon[a]

[a] *The University of Edinburgh, School of Mathematics, Peter Guthrie Tait Road, Edinburgh, EH9 3FD, United Kingdom*

Abstract

In recent years the expansion of energy supplies from volatile renewable sources has triggered an increased interest in stochastic optimization models for hydro-thermal unit commitment. Several studies have modelled this as a two-stage or multi-stage stochastic mixed-integer optimization problem. Solving such problems directly is computationally intractable for large instances, and alternative approaches are required. In this paper we use a Dantzig-Wolfe reformulation to decompose the stochastic problem by scenarios. We derive and implement a column generation method with dual stabilisation[1] and novel primal and dual initialisation techniques. A fast, novel schedule combination heuristic is used to construct very good primal solutions, and numerical results show that knowing these from the start improves the convergence of the column generation method significantly. We test our method on a central scheduling model based on the British National Grid and illustrate that convergence to within 0.1% of optimality can be achieved quickly.

Keywords: stochastic programming, mixed-integer column generation, Dantzig-Wolfe decomposition, Lagrangian relaxation, heuristics

2010 MSC: 90C11, 90C15,

1. Introduction

For many years, problems in the electric power industry have posed challenges to state of the art solvers and stimulated development of new optimization techniques. Optimization models for efficient scheduling and operation of power systems often involve integer decision variables which make them very challenging. A classical problem in this area is the unit commitment (UC) problem which is used to schedule power plants over a short (e.g. one or two day) planning horizon so as to minimise costs while satisfying electricity demand and keeping sufficient reserve to operate the system reliably. In recent years, growing power supplies from volatile renewable energy sources have introduced infeed uncertainty into the system. These developments have triggered a renewed interest in applying stochastic optimization techniques. In particular, there has been a growing interest in the formulation of stochastic unit commitment (SUC) problems and dedicated solution techniques. Various case studies have modelled this problem as two-stage model with (mixed-) integer recourse [1, 2] or as multi-stage model [3, 4, 5]. Many SUC studies focus on markets in the United States where this is a topical theme, despite the pricing issues which still impede its implementation for day to day market clearing purposes. [6, 7, 8]

Despite efficient reformulations and improvements in general mixed-integer programing (MIP) software, even

*Corresponding author

Email addresses: timschulze@gmx.net (Tim Schulze), k.mckinnon@ed.ac.uk (Ken McKinnon)

[1] Throughout this paper stability refers to stability of dual iterates in Lagrangian decomposition methods rather than power system stability.

deterministic UC is a challenging problem [9, 10, 11, 12, 13], and this motivated the application of decomposition techniques. A common approach for both deterministic and stochastic UC problems is to decompose by generation units. Individual units are linked by the load balance and spinning reserve constraints, and if those are relaxed, e.g. via Lagrangians [3] or augmented Lagrangians [14], the problem becomes separable by generators (in the case without network constraints). Single unit subproblems can be solved efficiently by dynamic programming, or stochastic dynamic programming if the original model is a stochastic model. Both Lagrangian relaxation and Dantzig-Wolfe (DW) decomposition have been proposed to decompose SUC problems by generation units [4, 15]. The number of linking demand and reserve constraints which have to be dualised grows linearly in the number of included scenarios. The Lagrangian dual problem is often solved by a cutting plane or proximal bundle method [1, 3, 4], in which case Lagrangian decomposition becomes the dual equivalent of column generation (ColGen). Briant et al. [16] assess the usefulness of stabilisation for improving the convergence of these methods, and [17] describes the use of stabilised Lagrangian decomposition in an industry solver.

An alternative way of decomposing stochastic UC problems is by scenarios. If the non-anticipativity property is formulated via constraints and these constraints are relaxed, then the problem becomes separable by scenarios. Scenario decomposition yields deterministic UC problems as subproblems, and any solution technique suitable to them can be used as a subproblem solver, e.g. embedded Lagrangian decomposition by generation units as proposed in [5] or standard mixed integer programming techniques. Various approaches have been proposed to achieve separability by scenarios. Papavasiliou et al. [6, 7] perform Lagrangian relaxation of non-anticipativity constraints in a two-stage model and solve the dual problem by a subgradient method, while Carøe and Schultz [1] use a proximal bundle method for the dual. Takriti and Birge [5] apply the popular Progressive Hedging (PH) algorithm [18], which was developed for convex stochastic programs and can be used as a heuristic (without guarantee of convergence) in the mixed-integer case. More recently, Watson and Woodruff [19] improve the PH heuristic for mixed-integer problems by tuning its parameters and deriving heuristics to find integer feasible solutions faster. Gade et al. [20] demonstrate how valid lower bounds can be derived in each step of the PH algorithm, thus overcoming the drawback that bounds are not readily available in PH. They also explore how the quality of these bounds changes with the algorithm parameters. Finally, Cheung et al. [8] tune the PH approach towards UC applications and report on its performance when applied to solve North American test instances with up to 340 generators to within 2.5%-1% of optimality.

In the special case where the problem has only two stages, a third decomposition approach can be taken: Benders decomposition, or the L-Shaped Method was originally devised for problems with continuous recourse but was subsequently generalised for integer recourse models [21]. Benders decomposition can be combined with DW or Lagrangian approaches to obtain cross-decomposition methods which sometimes yield better performance [22]. Zheng et al. [23] use Benders decomposition to decompose two-stage SUC problems by stages. Unlike other two-stage formulations [1, 2], theirs has binary variables on both the first and second stages. More recently, Ghaddar et al. [24] proposed another way of decomposing the problem: they apply Lagrangian relaxation to decompose a hydro plant scheduling problem by time periods.

Since Lagrangian and DW methods solve the dual problem or a convex relaxation of the primal problem, they do not necessarily find a primal feasible solution, and additional work is required to achieve that. Gröwe-Kuska et al. [3, 4] use various Lagrangian based heuristics and an additional Economic Dispatch stage to find good primal solutions. Shiina and Birge [15] use a schedule reduction technique to generate integer solutions

from a restricted master problem. Additionally, for problems with integer variables, there may be a duality gap, in which case a branching scheme such as Branch & Price is needed to verify that a global solution has been found. Carøe and Schultz [1] embed their dual algorithm in a branching scheme and use a primal rounding heuristic to produce intermediate solutions and accelerate convergence.

Finally, Goez et al. [2] review two-stage and multi-stage stochastic formulations of the UC problem and test various solution techniques: Lagrangian relaxation as in [1], Progressive Hedging as in [5] and a heuristic based on successively decreasing the number of relaxed variables in a linear programming (LP) relaxation. A comprehensive review of decomposition techniques for energy problems is also given in [25].

Our solution approach is based on DW scenario decomposition and can be applied in the two-stage and multi-stage case. A generic framework for this type of decomposition is described by Lulli and Sen [26], and a stochastic version of this algorithm for convex continuous multistage stochastic programs has recently been proposed in Higle et al. [27]. Non-anticipativity constraints are separated from the remaining constraints by relaxing them and dealing with them in the master problem. We use a proximal bundle approach to stabilise the master problem and accelerate convergence. Our method is equivalent to solving the Lagrangian dual by a proximal bundle method, which generally results in faster convergence than e.g. applying a subgradient update to the dual. We derive a dual initialisation procedure and construct primal feasible solutions using a fast, novel schedule combination heuristic which uses the schedules generated by the ColGen procedure. Although theoretically, in order to guarantee optimality, the ColGen approach needs to be embedded in a Branch & Price framework, we demonstrate that for the problems considered, very small duality gaps are achieved without branching.

We apply our algorithm to both two-stage and multi-stage SUC problems and report on its computational performance in a central scheduling model based on the British (GB) National Grid under wind power supply uncertainty. The model uses estimated data for 2020, with a wind penetration of 30% in terms of installed capacity, an aggregated model of the transmission system, and a sophisticated pump storage model [28]. We solve test instances with roughly 150 generation units including pump storage, 17 transmission regions and up to 50 scenarios to an optimality gap of 0.1%. We also explore the behaviour of our method on 50 scenario cases when the required gap is as small as 0.01%. Due to the requirement for small gaps we use a MIP solver as reference for the performance evaluation of our method.

The remainder of this paper is organised as follows: Section 2 has a formulation of a generic SUC model; Section 3 has a derivation of the basic decomposition principle; Section 4 has an explanation of practical algorithmic issues including initialisation and stabilisation; Section 5 has numerical test results from applying this method to our model of the GB power system; and Section 6 has the conclusions.

2. Stochastic Unit Commitment Models

We consider the day-ahead and intraday UC problem under wind uncertainty. Our day-ahead planning model is a two-stage recourse model, while the intraday model is a multi-stage recourse model. Day-ahead scheduling problems have a two-stage decision structure where a single commitment decision is implemented before observing the various wind outcomes, while intraday scheduling implies recurring schedule updates throughout the day, resulting in a multi-stage decision structure. Both day-ahead and intraday problems cover 24 periods of an hour each, but the number of periods covered by each stage differs between the two: in two-stage problems the

second stage covers 24 periods, while in n-stage problems with $n > 2$, stages $2, \ldots, n-1$ cover 3 periods each and stage n covers the remainder of the planning horizon. We present a general mathematical model here, which contains both situations as special cases, depending on the choice of non-anticipativity constraints. Let $\mathcal{S}$ be the set of scenarios. Non-anticipativity constraints are included for every bundle $b \in \mathcal{B}$ to force decision variables across all scenarios contained in the subset $\mathcal{S}_b \subseteq \mathcal{S}$ to be equal during periods where they are indistinguishable. This allows for situations where one stage covers multiple time periods. A simple stochastic unit commitment formulation is given below.

$$\sum_{s\in\mathcal{S}} P_s \left(\sum_{t=1}^{T} \sum_{g\in\mathcal{G}} \left(C_g^{st}\gamma_{gts} + C_g^{up}\alpha_{gts} + C_g^{m} p_{gts} \right) \right) \tag{1}$$

$$\sum_{g\in\mathcal{G}} p_{gts} = D_{ts},\ \forall t = 1, \ldots, T,\ s \in \mathcal{S} \tag{2}$$

$$P_g^{min}\alpha_{gts} \leq p_{gts} \leq P_g^{max}\alpha_{gts},\ \forall t = 1, \ldots, T,\ g \in \mathcal{G},\ s \in \mathcal{S} \tag{3}$$

$$\alpha_{gts} - \alpha_{g(t-1)s} = \gamma_{gts} - \eta_{gts},\ \forall t = 1, \ldots, T,\ g \in \mathcal{G},\ s \in \mathcal{S} \tag{4}$$

$$\gamma_{gts} + \eta_{gts} \leq 1,\ \forall t = 1, \ldots, T,\ g \in \mathcal{G},\ s \in \mathcal{S} \tag{5}$$

$$p_{gts}^{gen} - p_{g(t-1)s}^{gen} \leq P_g^{ru}\alpha_{g(t-1)s} + P_g^{su}\gamma_{gts},\ \forall t = 1, \ldots, T,\ g \in \mathcal{G},\ s \in \mathcal{S} \tag{6}$$

$$p_{g(t-1)s}^{gen} - p_{gts}^{gen} \leq P_g^{rd}\alpha_{gts} + P_g^{sd}\eta_{gts},\ \forall t = 1, \ldots, T,\ g \in \mathcal{G},\ s \in \mathcal{S} \tag{7}$$

$$\sum_{i=t-T_g^u+1}^{t} \gamma_{gis} \leq \alpha_{gts},\ \forall t = 1, \ldots, T,\ g \in \mathcal{G},\ s \in \mathcal{S} \tag{8}$$

$$\sum_{i=t-T_g^d+1}^{t} \eta_{gis} \leq 1 - \alpha_{gts},\ \forall t = 1, \ldots, T,\ g \in \mathcal{G},\ s \in \mathcal{S} \tag{9}$$

$$x_{bs} = \bar{x}_b,\ b \in \mathcal{B},\ s \in \mathcal{S}_b \tag{10}$$

The objective (1) is to minimise the expected generation cost, consisting of unit startup costs C_g^{st}, running costs C_g^{up} and marginal costs C_g^m. Here P_s is the probability of scenario s, $\mathcal{G}$ is the set of generators and T is the time horizon. Binary variables α_{gts}, γ_{gts} and η_{gts} indicate whether generator g is on, is being started up, or is being shut down in period t under scenario s, respectively. This 3-Binary formulation was found to be the most efficient in [12]. Continuous variables p_{gts} denote the generator's power output. Load balance equations (2) require the total power output to be equal to the residual load D_{ts} at any time under every scenario. Generator bounds (3) say that a generator has to either be off ($\alpha_{gts} = 0$, $p_{gts} = 0$) or operate between its lower and upper power output margins P_g^{min} and P_g^{max}. Constraints (4) establish the connection between binaries: for $\alpha_{gts} = 1$ and $\alpha_{g(t-1)s} = 0$ we have that $\gamma_{gts} = 1$ and $\eta_{gts} = 0$, so the generator is being started up. For $\alpha_{gts} = 0$ and $\alpha_{g(t-1)s} = 1$ we have that $\gamma_{gts} = 0$ and $\eta_{gts} = 1$ and the generator is being shut down. The remaining two cases are $\alpha_{gts} = \alpha_{g(t-1)s} \in \{0, 1\}$, so the generator is on or off for successive periods, and in the presence of (6) and (7), (5) is needed to impose $\gamma_{gts} = \eta_{gts} = 0$ and avoid simultaneous startups and shutdowns. Contraints (6) and (7) are ramp up (down) rates which limit the change between power outputs in sequential periods to P_g^{ru} (P_g^{rd}) when the generator is on, and to P_g^{su} (P_g^{sd}) when the generator is started up (shut down). Constraints (8) say that generator g has to be on in period t if it has been started up within the T_g^u preceding periods. Similarly, constraints (9) require the generator to be off in period t if it has been shut down in the preceding T_g^d periods. These are called minimum up and downtimes. Finally we have non-anticipativity constraints (10). These force subsets of variables to be equal across the scenarios $s \in \mathcal{S}_b$ of bundle b at all times where they are

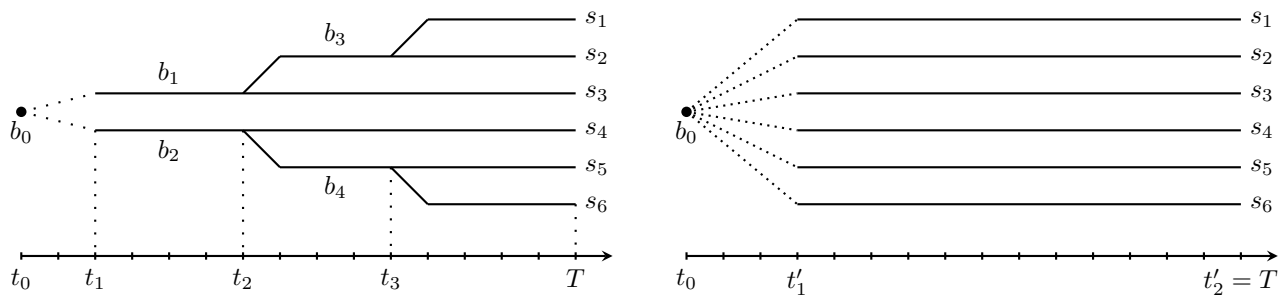


Figure 1: Left: Four-stage decision tree with bundles $\mathcal{B} = \{b_0, \ldots, b_4\}$. The first stage decisions, made at t_0, are the unit commitments between t_1 and t_2. These are identical in b_0, i.e. for $\mathcal{S}_{b_0} = \{s_1, \ldots, s_6\}$. The second stage decisions, made at t_1, are the power outputs between t_1 and t_2. These are identical in b_1, i.e. for $\mathcal{S}_{b_1} = \{s_1, s_2, s_3\}$, and identical in b_2, i.e. for $\mathcal{S}_{b_2} = \{s_4, s_5, s_6\}$. The third stage decisions, made at t_2, are all the unit commitments and power outputs between t_2 and t_3. These are identical in b_3 and in b_4, i.e. for $\mathcal{S}_{b_2} = \{s_1, s_2\}$ and $\mathcal{S}_{b_4} = \{s_5, s_6\}$, respectively. The fourth stage decisions, made at t_3, are all the unit commitments and power outputs between t_3 and t_4. These are not covered by a bundle so are independent between scenarios.

Right: Two-stage decision tree with one bundle b_0. The first stage decisions, made at t_0, are the unit commitments between t'_1 and t'_2. These are identical in b_0, i.e. for $\mathcal{S}_{b_0} = \{s_1, \ldots, s_6\}$. The second stage decisions, made at t'_1, are the power outputs between t'_1 and t'_2. These are not covered by a bundle so are independent in different scenarios.

indistinguishable. As is explained later, it is sufficient to impose non-anticipativity on subsets of the variables α, γ, η and p. For ease of exposition we condense those α, γ, η and p, variables of scenario s that need to satisfy the non-anticipativity condition in bundle b into a vector x_{bs}, and define a separate copy $\bar{x}_b$ for their common value. Figure 1 has an example of how to use these data structures to impose a non-anticipative structure on the decisions. For further examples of two-stage and multi-stage models we refer to Goez et al. [2]. Where equations imply variables with indices $t < 1$ we assume these to be input data which define the power system's initial state. The problem shown here is a basic UC problem, and the full formulation of the model used to evaluate our decomposition method also includes a simplified transmission network model, system-wide spinning reserve, and a detailed model of pump storage reservoir operation. A detailed description can be found in [28]. Alternative UC formulations are discussed in [12, 10, 13, 9].

Uncertainty Model And Extensive Formulation. The *multi-stage* version of our model uses a scenario tree as shown in Figure 1 (left). The root node represents the point at which the schedule is prepared, which is some time before the operation starts. Consequently we cannot assume that the wind or system state is known when operations are due to start, so we use multiple scenarios in the second stage to capture this uncertainty and ensure feasibility of the schedule. We require that the commitment decisions are the same for these second stage scenarios, but the recourse decisions can be different. These recourse decisions include the generator power output, pump storage operation and open-cycle gas turbine (OCGT) operation. At later stages we assume that all the variables including commitment variables can take different values in different scenarios, subject to a notification time for startups of large generators. A bundle is required for every distinct group of scenarios to ensure non-anticipativity of all decisions, and between t_1 and t_2 an additional bundle is required to make commitment decisions unique. This formulation is not standard in the UC literature, where multi-stage trees are often restricted to a *single* scenario for the first few hours of the planning horizon. Further details on the non-anticipativity model can be found in [28].

An example of a *two-stage* decision tree is shown in Figure 1 (right). This is eqivalent to the first two

stages of a multi-stage model. All the commitment decisions are the same throughout the second stage and, because the second stage is also the final stage, the recourse decisions are made under perfect information. In the remainder of this paper we use the following condensed notation for the *extensive* formulation (1) - (10) of the SUC problem, i.e. the formulation which covers all scenarios at once

$$\text{SUC:} \quad \min_{c,x} \quad \sum_{s\in\mathcal{S}} P_s c_s \tag{11}$$

$$\text{s.t.} \quad (c_s, x_s) \in \mathcal{X}_s, \ \forall s \in \mathcal{S} \tag{12}$$

$$x_{bs} = \bar{x}_b, \ b \in \mathcal{B}, \ s \in \mathcal{S}_b. \tag{13}$$

Here P_s is again the probability of scenario s and c_s is the total cost under that scenario. All decision variables have been condensed into a single vector x. For every scenario $s \in \mathcal{S}$ the sets $\mathcal{X}_s$ contain all tuples of decision variables (c_s, x_s) such that x_s is a vector of feasible operational decisions subject to (2) - (9) and c_s is the associated cost. We assume that individual generators' cost functions are piecewise linear (PWL) so that $\mathcal{X}_s$ can be described by a finite number of linear constraints and integrality restrictions. In UC problems the $\mathcal{X}_s$ are bounded. The non-anticipativity constraints (13) are formulated in terms of additional vectors of variables $\bar{x}_b$ which can be interpreted as common target values for all scenarios in a bundle. This is analogous to Lulli and Sen's formulation with common target variables [26].

Non-Anticipativity and Degeneracy. Non-anticipativity constraints in UC problems typically have a high degree of redundancy, which causes dual degeneracy. In the numerical examples in Section 5, we explore the extent of dual degeneracy and its effect on the decomposition method. In the following we demonstrate typical causes of degeneracy.

In UC problems, on/off, startup and shutdown decisions can be modelled using one [10], two [2] or three [12] binary variables per generator, period and scenario. If multiple sets are used, like in formulation (1) - (10), the values of on/off variables α uniquely determine the values of startup and shutdown decisions γ and η, so non-anticipativity constraints are only required for α, and our algorithm works best with this single set. Other reasons for redundancy are not as obvious. Consider the generator power output restrictions (3) where $0 < P_g^{min} < P_g^{max}$ are the generation limits of unit g. With α being binary, the relation between power output and on/off variables is

$$p_{gts} = 0 \Leftrightarrow \alpha_{gts} = 0 \quad \text{and} \quad p_{gts} \in \left[P_g^{min}, P_g^{max}\right] \Leftrightarrow \alpha_{gts} = 1, \tag{14}$$

so non-anticipativity of p variables implies non-anticipativity of α variables. Consequently, non-anticipativity constraints for α are redundant if they are included for p as well. With appropriate dual stabilisation, however, our scenario decomposition technique is more successful when both types of constraints are present. An additional cause of redundancy are the minimum up and downtime constraints: non-anticipativity of α_{gts} for all $s \in \mathcal{S}$ often implies non-anticipativity of $\alpha_{g(t+k)s}$, for some integer k smaller than the minimum up and downtime. However, in this case it is not possible to know a priori which constraints are redundant. Since there is no easy way of finding a non-redundant subset of constraints that guarantees non-anticipativity, the solution techniques must be able to deal with redundant constraints.

3. Dantzig-Wolfe Scenario Decomposition

Dantzig-Wolfe (DW) decomposition was originally developed for linear programs with block-angular structure of the constraint matrix and later generalised to the mixed integer case [29, 30]. We apply it to decompose the SUC problem (11) to (13) by scenarios. Relaxing non-anticipativity to derive decomposition schemes is a common concept in two-stage and multi-stage stochastic programming: Lagrangian decomposition [27, 31], Progressive Hedging [18] and Dantzig-Wolfe decomposition [26] have all been used in the context of scenario decomposition.

In DW decomposition, the original problem is replaced by a master problem. Since the $\mathcal{X}_s$ are bounded, their convex hulls $\text{conv}(\mathcal{X}_s)$ have a finite number of extreme points and can be represented by a convex combination of them. Let $\hat{\mathcal{I}}_s$ denote the (finite) index set of all extreme points of $\text{conv}(\mathcal{X}_s)$, and $\mathcal{I}_s \subseteq \hat{\mathcal{I}}_s$ a subset of them. At the i-th extreme point, let the cost be c_{si} and the value of the x_{bs} be X_{bsi}. We introduce a (scalar) weight variable w_{si} and formulate a restricted master problem (RMP) whose columns are those extreme points of $\text{conv}(\mathcal{X}_s)$ which are included in the index subset $\mathcal{I}_s$:

$$\text{RMP}: \quad \min_{w,\bar{x}} \quad \sum_{s\in\mathcal{S}} P_s \sum_{i\in\mathcal{I}_s} c_{si} w_{si} \tag{15}$$

$$\text{s.t.} \quad \sum_{i\in\mathcal{I}_s} X_{bsi} w_{si} = \bar{x}_b, \ b \in \mathcal{B}, \ s \in \mathcal{S}_b \tag{16}$$

$$\sum_{i\in\mathcal{I}_s} w_{si} = 1, \ \forall s \in \mathcal{S} \tag{17}$$

$$w \geq 0, \ \bar{x} \text{ free.} \tag{18}$$

The unrestricted master problem which contains all the columns in $\hat{\mathcal{I}}$ is a convex relaxation of the SUC problem, because its feasible region is given by $\text{conv}(\mathcal{X}_s)$ for all s. In the following we refer to the unrestricted master problem as MP. As in the original problem, the objective (15) is to minimise the expected total cost. The non-anticipativity constraints for convex combinations of individual columns are (16) and we denote their vectors of dual variables by λ_{bs}. The use of common target variables $\bar{x}$ in the original SUC problem and the master problem does not increase the amount of degeneracy. In comparison to a formulation without common target variables there is exactly one additional equality constraint and one additional variable, so the dimensions of both the primal and dual feasible spaces remain unchanged. Like Lubin et al. [32], we find that the scenario decomposition derived from this formulation performs better than one derived from an equivalent formulation without common target variables. Constraints $w \geq 0$ together with (17) are convexity constraints, and the dual variables of (17) are denoted by σ_s. The MP is optimal if it contains no non-basic columns which have negative reduced cost. The reduced cost of a column is given by

$$\bar{c}_{si} := P_s c_{si} - \sum_{b\in\mathcal{B}:s\in\mathcal{S}_b} \lambda_{bs}^T X_{bsi} - \sigma_s, \tag{19}$$

however testing for optimality this way requires that all the columns in $\hat{\mathcal{I}}_s$ for all scenarios s have been generated, which is impractical as the number of them is typically exponential [33]. The ColGen approach is to solve the RMP with the current set of columns to find the multipliers λ_{bs}, then search for columns with negative reduced cost by solving the following pricing subproblem for each scenario s

$$\bar{c}_s^* := \min_{(c_s,x_s)\in\mathcal{X}_s} \left(P_s c_s - \sum_{b\in\mathcal{B}:s\in\mathcal{S}_b} \lambda_{bs}^T x_{bs} \right) - \sigma_s. \tag{20}$$

If the pricing subproblems are solved to optimality and $\bar{c}_s^* \geq 0$ for all s, then the current optimal solution of the RMP is also optimal for the MP. Otherwise new columns are generated for a subset of solutions of (20) with the smallest $\bar{c}_s^* < 0$. These are added to the $\mathcal{I}_s$ sets to give a new RMP and the procedure is repeated. In this scenario decomposition method, the pricing problems are deterministic UC problems.

Since the MP is a convex relaxation of the SUC problem, solving it provides a *lower bound* on the optimal objective value [33]. Any feasible solution of the SUC gives an *upper bound*, and the optimal solution gives the best possible upper bound.

The gap between the best possible upper bound and the lower bound from the optimal MP solution is the *duality gap*. In practice the gap between the best known bounds is bigger than the duality gap: the optimal solution of the SUC may not have been found and only a lower bound on the optimal MP solution may be known. If the gap is not sufficiently small, then if we have not yet solved the MP we can try to improve its lower bound, or we can try to find a better feasible solution using heuristics. If it is not possible to reduce the gap by improving either the upper or lower bound, then it is necessary to perform branching on the RMP, resulting in Branch & Price (B&P) [29]. If the duality gap by itself is too large, then branching cannot be avoided. However, in all our test examples our algorithm achieved sufficiently small gaps at the root node of the B&P tree, so no branching was ever needed.

4. Practical Aspects of the Algorithm

A plain ColGen procedure has practical drawbacks concerning its convergence speed. These are well known and described e.g. in Vanderbeck [29], from where we adopt the following terminology. In the first iterations the RMP lacks a sufficient variation of columns to produce good primal and dual solutions (Heading-in effect). Further, while there are multiple dual optimal solutions to the RMP, its primal solution often remains constant over multiple iterations (Plateau effect). Dual RMP solutions jump from one extreme value to another and cause fluctuations of the lower bound (Bang-bang effect). Finally, after some initial progress, many additional columns need to be generated before optimality can be proven (Tailing-off effect). The effects must be accounted for by appropriate algorithmic modifications: we address the former two by deriving a powerful heuristic to find primal solutions and hot-starting our procedure with dual estimates from an LP relaxation. The latter two issues are alleviated by using a bundle method to stabilise the dual iterates. Since deterministic UC problems are challenging problems themselves, the subproblems cannot always be solved to optimality, and we explain how this is handled in our approach.

4.1. Dual Stabilisation of the RMP

DW decomposition and Lagrangian relaxation (LR) are closely related: both can be used to obtain separability in the same way, and both give the same lower bound for a given set of multipliers [34, 33]. If the Lagrangian dual problem is solved by a cutting plane algorithm, the LR and ColGen procedures are identical since the RMP and the cutting plane problem form a primal-dual pair of LPs. We work with the cutting plane problem because it provides a natural motivation for dual stabilisation via proximal bundle methods. Let λ_{bs} and σ_s be the dual variables of constraints (16) and (17), respectively. The stabilised cutting plane problem is

given by

$$\text{dRMP}(\epsilon): \quad \max_{\sigma,\lambda} \quad \sum_{s\in\mathcal{S}} \sigma_s - \frac{\epsilon}{2}\|\lambda - \hat{\lambda}\|_2^2 \tag{21}$$

$$\text{s.t.} \quad \sigma_s \leq P_s c_{si} - \sum_{b\in\mathcal{B}: s\in\mathcal{S}_b} \lambda_{bs}^T X_{si},\ \forall s \in \mathcal{S},\ i \in \mathcal{I}_s \tag{22}$$

$$\sum_{s\in\mathcal{S}_b} \lambda_{bs} = 0,\ \forall b \in \mathcal{B} \tag{23}$$

$$\lambda,\ \sigma \text{ free}, \tag{24}$$

with a quadratic regularisation term centered on the current dual iterate $\hat{\lambda}$ and a steplength parameter ϵ.

When $\epsilon = 0$ this is the LP dual of the RMP. The artificial variables $\bar{x}_b$ of (16) translate to dual constraints (23), which force non-anticipativity multipliers to sum up to zero among bundled scenarios. These constraints remove the additional dimension introduced to the dual problem by the additional non-anticipativity constraint in every bundle in formulation (16). The maximum possible value of $\sum_s \sigma_s$ subject to constraints (22) and (23) gives an upper PWL approximation to the original problem's Lagrangian dual function, and when $\mathcal{I} = \hat{\mathcal{I}}$ it gives the exact Lagrangian dual function [34, 33]. Every column of the RMP corresponds to one supporting hyperplane. Without a sufficient variety of cutting planes (22), the PWL model of the Lagrangian is too optimistic and encourages too large dual steps. The duals *lambda* are unrestricted and have zero coefficients in the objective, so arbitrarily large values are possible if they are compensated for by small values which ensure that (23) holds. The columns or cuts generated from such large multipliers are usually not meaningful in that they do not form part of a good primal solution.

With $\epsilon > 0$ the procedure is stabilised through the quadratic bundle term centered on the current dual iterate $\hat{\lambda}$. Controlled by the steplength parameter ϵ, dRMP(ϵ) strikes a balance between maximising the current PWL model of the Lagrangian and not moving too far from the last successful iterate. It produces a candidate solution $\tilde{\lambda}$ which is either accepted by setting $\hat{\lambda} := \tilde{\lambda}$ (serious step) or rejected by keeping the old iterate (null step). A serious step occurs when the new iterate improves the current lower bound on the MP solution. This bound is obtained by solving the subproblems (cf. next section), and irrespective of whether the step is serious or not, cutting planes are added to dRMP(ϵ) for a subset of subproblem solutions which have negative reduced cost. After adding the cuts, dRMP(ϵ) is solved again. For additional convergence improvements, we use a variable steplength logic: we increase ϵ if the lower bound has deteriorated, keep it constant if the bound has improved, and decrease it if the bound has not improved over multiple iterations.

4.2. Lower Bounds for the MP

The Lagrangian dual function for our SUC problem is

$$L^D(c,x,\lambda) = \min_{(c,x)\in\mathcal{X}} \left(\sum_{s\in\mathcal{S}} P_s c_s - \sum_{b\in\mathcal{B}} \sum_{s\in\mathcal{S}_b} \lambda_{bs}^T (x_{bs} - \bar{x}_b) \right) \tag{25}$$

$$= \min_{(c,x)\in\mathcal{X}} \sum_{s\in\mathcal{S}} \left(P_s c_s - \sum_{b\in\mathcal{B}: s\in\mathcal{S}_b} \lambda_{bs}^T x_{bs} \right) + \sum_{b\in\mathcal{B}} \bar{x}_b^T \sum_{s\in\mathcal{S}_b} \lambda_{bs} \tag{26}$$

$$\overset{(23)}{=} \sum_{s\in\mathcal{S}} \min_{(c_c,x_s)\in\mathcal{X}_s} \left(P_s c_s - \sum_{b\in\mathcal{B}: s\in\mathcal{S}_b} \lambda_{bs}^T x_{bs} \right) \tag{27}$$

$$= \sum_{s\in\mathcal{S}} L_s^D(c_s, x_s, \lambda_s), \tag{28}$$

and for feasible multipliers λ this provides a lower bound on the optimal MP value. The term in $\bar{x}_b$ vanishes due to constraints (23). For given multipliers λ and σ, we have that $L_s^D(c_s, x_s, \lambda_s) = \bar{c}_s^* + \sigma_s$, so the lower and upper bounds v^{lb} and v^{ub} on the optimal MP objective value v^* are given by

$$v^{lb} := L^D(c, x, \lambda) = \sum_{s \in \mathcal{S}} (\sigma_s + \bar{c}_s^*) \leq v^* \leq \sum_{s \in \mathcal{S}} \sigma_s =: v^{ub}. \tag{29}$$

This shows that the lower bound v^{lb} is obtained by solving the pricing subproblems (20) and adding their (negative) reduced costs to the current RMP objective value v^{ub}. For the special choice $\lambda = 0$, the scenario subproblems are solved independently under perfect information, and the resulting bound v^{lb} is called the expected value *under* perfect information. The gap between the optimal SUC objective value and this bound is known as the expected value *of* perfect information (EVPI).

We solve the subproblems (20) with a Branch & Cut solver and since they are large MIPs, optimality cannot always be achieved within a reasonable timeframe. The solver terminates with a set of (sub)optimal solutions and a lower bound on the objective, i.e. the reduced cost. Let this bound be $\bar{c}_s^{lb} \leq \bar{c}_s^*$. Then, instead of (29) we use

$$v^{lb} = v^{ub} + \sum_{s \in \mathcal{S}} \bar{c}_s^{lb} \tag{30}$$

as a valid lower bound for the MP. To decide if the (sub)optimal solutions should be added as columns to the RMP, we evaluate their reduced costs individually and test them for negativity. The termination criterion for the ColGen procedure is adapted to allow for subproblem non-optimality: we stop when the following overestimate of the relative MP gap

$$\delta^{MP} := \frac{-\sum_{s \in \mathcal{S}} \bar{c}_s^{lb}}{v^{ub}} \tag{31}$$

satisfies a pre-defined optimality tolerance. The smallest achievable gap δ^{MP} depends on the MIP gaps of the pricing subproblems.

4.3. Dual Initialisation of the RMP

There are multiple strategies for setting the initial stability center $\hat{\lambda}$ in dRMP(ϵ), e.g. it can be set to zero or we can use the dual solution of the SUC problem's LP relaxation. In the latter case we can solve the extensive formulation of the SUC problem's LP relaxation exactly or apply the same scenario decomposition as in the integer case, and in Section 5 we report on tests with both strategies. Dual solutions of the SUC LP relaxation are similarly degenerate as those of its mixed-integer version. We wish to explore the effect of using different dual solutions of the relaxation as starting points for the ColGen algorithm, and this requires a way of controlling which dual solution is obtained. In the following we outline briefly how a quadratic penalty term can be included in the *primal* SUC LP relaxation in order to obtain approximately optimal *dual* solutions of various sizes. This is more convenient to implement than a dual formulation of the SUC LP relaxation. We first show how to do this for a general LP before applying it to the SUC problem. Consider the following primal-dual pair of quadratic programs (QP) where the primal has two sets of equality constraints:

$$\begin{aligned} \min_{x,\lambda} \ & c^T x + \tfrac{1}{2}\mu\lambda^T\lambda \\ \text{s.t. } & Ax = b \\ & Bx + \mu\lambda = d \\ & x \geq 0, \ \lambda \text{ free} \end{aligned} \tag{32}$$

$$\begin{aligned} \max_{\gamma,\lambda} \ & b^T\gamma + d^T\lambda - \tfrac{1}{2}\mu\lambda^T\lambda \\ \text{s.t. } & A^T\gamma + B^T\lambda \leq c \\ & \gamma, \lambda \text{ free} \end{aligned} \tag{33}$$

When $\mu = 0$ these are a primal-dual pair of LPs where γ are the duals of $Ax = b$, and λ are the duals of $Bx = d$. We think of $Bx = d$ as the non-anticipativity constraints of the SUC LP relaxation and assume that they are degenerate in that there is a continuum of dual optimal solutions λ. To favour λ of smaller magnitude, we have included a quadratic penalty term with small $\mu > 0$ in the dual (33) and equivalent terms in the primal (32) so that the problems form a primal-dual pair of QPs. The primal QP has a penalty term on λ in the objective, and the added term $\mu\lambda$ in the constraints $Bx = d$ which λ is associated with as multiplier.

By varying the size of μ in (32) we obtain different dual solutions λ. In the following we explain how to monitor whether the size of μ is appropriate to obtain λ which can be used to warm-start dRMP(ϵ). For easier interpretation, (32) can be re-written as (34) with $v := \mu\lambda$. In (34), $v = d - Bx$ is the violation of $Bx = d$ and must be paid for in the objective, but its price decreases with increasing μ. In SUC problems v is bounded, so we have $\frac{1}{2\mu}v^T v \to 0$ for $\mu \to \infty$. Thus, for large μ the constraints $Bx = d$ are relaxed, since violating them is free. Then the solution of (34) converges to the solution of (35) while the duals of the relaxed constraints approach zero: $\lambda = \frac{1}{\mu}v \to 0$. As we are interested in these duals, μ must be chosen with care. The magnitude of the violation $\|v\|_2$ indicates if the level of μ is appropriate: if it is non-negligible, then μ is too large.

$$\begin{aligned} \min_{x,v}\ & c^T x + \tfrac{1}{2\mu}v^T v && (34) \\ \text{s.t. } & Ax = b \\ & Bx + v = d \\ & x \geq 0,\ \hat{z} \text{ free} \end{aligned} \qquad \begin{aligned} \min_{x}\ & c^T x && (35) \\ \text{s.t. } & Ax = b \\ & x \geq 0 \end{aligned}$$

We use this approach to regularise the duals of the non-anticipativity constraints in the SUC LP relaxation by including the required terms in μ and λ. The stabilised LP relaxation of the SUC problem is shown below. Here $\bar{\mathcal{X}}_s$ is obtained by relaxing the integrality restrictions of $\mathcal{X}_s$.

$$\begin{aligned} \text{LPR}(\mu): \quad \min_{c,x}\ & \sum_{s\in\mathcal{S}} P_s c_s + \tfrac{1}{2}\mu\lambda^T\lambda && (36) \\ \text{s.t. } & (c_s, x_s) \in \bar{\mathcal{X}}_s,\ \forall s \in \mathcal{S} && (37) \\ & x_{bs} + \mu\lambda_{bs} = \bar{x}_b,\ \forall b \in \mathcal{B},\ s \in \mathcal{S}_b && (38) \end{aligned}$$

4.4. MIP Heuristics

A central issue of LR and DW decomposition for mixed-integer problems is that integer primal solutions may not be found unless the master problem is branched on [33]. Primal solutions are required to find upper bounds for the problem and eventually solve it, and we use MIP-based heuristics to construct them. This is done in the beginning of the ColGen process and repeated regularly throughout it. The results discussed in Section 5 confirm that the decomposition is sped up significantly if a good solution is known early on.

A Schedule Combination Heuristic. Our core heuristic is a MIP-based schedule combination (SC) heuristic which constructs a non-anticipative solution for the SUC problem. It is inspired by Takriti and Birge's refinement heuristic for deterministic problems [35]. The idea is to solve the SUC problem as a MIP, but with the solution space reduced to a set of pre-defined schedules. Here, a schedule is a plan of binary on and off decisions for a *single generator* for the *whole planning horizon*. As before, let α_{gts} be the binary on/off decision of generator g in

period t in scenario s. By n_g we denote the number of known schedules for generator g and by A_{gjt} the on/off decision at time t under its j-th schedule. Note that A_{gjt} has no scenario index s, since the same universe of schedules is used for *all scenarios*. We formulate a stochastic mixed-integer schedule selection problem, which – under every scenario – picks one of the schedules for each of the generators, subject to satisfying non-anticipativity. It uses weight variables w_{gjs} which take a value of one if generator g chooses schedule j under scenario s, and zero otherwise. The schedule selection problem is obtained by *adding* variables w_{gjs} and the following constraints to the original SUC problem (1) -(10) :

$$\sum_{i=j}^{n_g} w_{gjs} A_{gjt} = \alpha_{gts},\ \forall g \in \mathcal{G},\ t = 1, \ldots, T,\ s \in \mathcal{S} \tag{39}$$

$$\sum_{i=j}^{n_g} w_{gjs} = 1,\ \forall g \in \mathcal{G},\ s \in \mathcal{S} \tag{40}$$

$$w_{gjs} \in \{0, 1\},\ \forall g \in \mathcal{G},\ s \in \mathcal{S},\ i = 1, \ldots, n_g. \tag{41}$$

Constraints (39) ensure that if schedule j is chosen for generator g under scenario s, its binary decisions α_{gts} are equal to A_{gjt} at *all* times $t = 1, \ldots, T$. For a given generator g and scenario s, constraints (40) say that exactly one of the binary weights w_{gjs} can be one, while all others are zero. The weights then form a specially ordered set of order 1 (SOS1). As the generators must follow their chosen schedules for the whole planning period, some of the constraints which are typically formulated in $\mathcal{X}_s$ can be dropped. For instance minimum up/down times will be satisfied by every schedule, so the constraints are no longer required.

The schedules A_{gjt} are generated through ColGen iterations: after solving the subproblems, we add all distinct schedules that appear in their solutions to A_{gjt}. This is done individually for each generator $g \in \mathcal{G}$. Schedules from different scenario subproblems are pooled in A_{gjt} and made available to all other scenarios. Thus scenarios can exchange their own schedule for one proposed by another scenario on a generator-by-generator basis. The tests confirm that this produces good solutions even if A_{gjt} is populated from a pass of ColGen with multipliers $\lambda = 0$, i.e. a perfect foresight solution. However, better solutions are obtained if the multipliers are such that they encourage the subproblems to generate schedules which are a compromise between bundled scenarios. The SC heuristic can be used to:

1. Initialise dRMP(ϵ) with a non-anticipative solution: we populate A_{gjt} from subproblem solutions of the first pass of ColGen, performed with multipliers equal to zero or estimated from LPR(μ).

2. Find solutions during the ColGen process: in every iteration, A_{gjt} is extended by new schedules from the subproblems. The heuristic is repeated and improved solutions are added to dRMP(ϵ).

The SC problem is solved with a general purpose MIP solver. We apply integrality restrictions to the SOS1 weights w_{gjs} and relax the on/off variables, since this improved solver performance. Additionally, the problem is pre-processed: we iterate over all on/off variables, and whenever the schedule set allows this generator only to be on or only to be off at a certain time, the corresponding variable is fixed and removed from the problem. For generators with only one schedule, the weights and on/off variables are all fixed and eliminated. In the test runs described in Section 5, roughly 70% of the weights and on/off variables are eliminated in this way.[2] Many MIP solvers use local search heuristics to improve existing solutions, so it is often beneficial for the convergence

[2]The solver's presolve phase can remove redundant variables, but we choose to maintain control of how it is done.

speed if an initial solution is provided. Since we solve the SC problem with a MIP solver, we present two cheaper heuristics here, which we use to construct a starting solution for the SC problem.

An Over-Commitment Heuristic. The following over-commitment heuristic attempts to repair a conflicting schedule obtained from the scenario subproblems by committing more than the required generation capacity:

1. Estimate values for λ, e.g. by solving LPR(μ) or setting them to zero
2. Solve the column generator subproblems with these dual estimates
3. Consider the obtained schedule: for each generator g, proceed in chronological order $t = 1, \dots, T$:
   - Find bundles b covering period t, whose members $s \in \mathcal{S}_b$ disagree on the commitment α_{gts}
   - For all members $s \in \mathcal{S}_b$ of these bundles, set $\alpha_{g(t+k)s} := 1$ with $k = 0, \dots, T_g^u - 1$ to satisfy (8)
4. If the solution violates minimum downtimes (9), repair it by keeping the generators on instead
5. The schedule is now feasible and non-anticipative. Solve a dispatch with fixed binaries for power output decisions.

The solution quality depends on the multiplier estimate: for $\lambda = 0$, many schedule adjustments are necessary and we obtain expensive solutions, while estimating the multipliers from LPR(μ) leads to few adjustments and nearly optimal solutions. Steps 1. and 2. are required before solving the SC heuristic anyway, so applying the over-commitment heuristic adds only a negligible computational cost. For multi-stage UC problems we obtained reasonable starting solutions with both $\lambda = 0$ and λ estimated from LPR(μ). However for two-stage problems many adjustments are necessary and the following alternative usually performs better.

Constructing Solutions for Two-Stage Problems. In the two-stage setting it is straightforward to construct a solution. It is possible to solve a deterministic problem, e.g. with an average or low wind scenario, and find the recourse action for the obtained schedule by solving scenario-specific dispatch problems. Feasibility of the approach is guaranteed if the model has relatively complete recourse. In the model which we use for our tests this is warranted by the fact that load shedding and not satisfying reserve are feasible options on the second stage [28]. For our test runs, we initialise the SC heuristic with a solution obtained from a deterministic problem which was augmented with a few additional power output variables. The additional variables are used to approximate the recourse cost: they inform the problem that sufficient capacity must be scheduled to cope even with the lowest wind scenario, or additional costs will be incurred to satisfy the remaining demand via standby reserve. Despite the additional variables, the computational effort is similar to a simple deterministic problem [36].

4.5. The Stabilised Scenario Decomposition Algorithm

Our scenario decomposition scheme for SUC problems is summarised in Figure 2. It solves the DW master problem at the root node of a Branch & Price tree. Dual stabilisation and initialisation techniques are included, as well as the heuristic to find integer feasible solutions. In all numerical tests the gap between the best integer SUC solution and the RMP objective value was within our required optimality tolerance after solving the root node, so no branching was performed. If branching is necessary, the ColGen procedure can be repeated at every node of the Branch & Price tree, with appropriate modifications so that the master and subproblems comply with the branch decisions.

Input: iteration counter $k \leftarrow 0$, optimality tolerance Δ, stabilisation levels ϵ, μ, upper bound $v_0^{ub} \leftarrow \infty$, lower bound $v_0^{lb} \leftarrow -\infty$, gap $\delta_0^{MP} \leftarrow \infty$, iterations until heuristic r is repeated, maximum number N of cuts added to dRMP(ϵ) per iteration per subproblem;

solve LPR(μ) to find $\hat{\lambda}$ or set $\hat{\lambda} \leftarrow 0$;

for $s \in \mathcal{S}$ **do**

  solve subproblem (20) with $\hat{\lambda}$ and $\sigma = 0$;

  for $g \in \mathcal{G}$ **do** add new schedules from opt. subproblem solution to SC heuristic;

  for *the best subproblem solutions* $i = 1, \ldots, N$ **do**

    calculate $\bar{c}_{si}$;

    if $\bar{c}_{si} < 0$ **then** generate a cut and add it to dRMP(ϵ);

  end

end

calculate lower bound v_k^{lb} from (30);

repeat

  if $k \mod r \equiv 0$ **then**

    run the heuristic to find $\tilde{v}_k^{ub}$, $\tilde{x}^k$;

    if $\tilde{v}_k^{ub} < v_k^{ub}$ **then**

      set current best solution $\left(v_k^{ub}, x^*\right) \leftarrow \left(\tilde{v}_k^{ub}, \tilde{x}^k\right)$ and calculate δ_k^{MP} from (31);

      if $\delta_k^{MP} < \Delta$ **then** terminate;

      for $s \in \mathcal{S}$ **do** generate a cut from $\tilde{x}_s^k$ and add it to dRMP(ϵ);

    end

  end

  set $k \leftarrow k + 1$;

  solve dRMP(ϵ) to find $\tilde{\lambda}$, $\tilde{\sigma}$, v_k^{ub}, x^k;

  if x^k *is integer feasible* **then** set current best solution $x^* \leftarrow x^k$;

  for $s \in \mathcal{S}$ **do**

    solve subproblem (20) with $\tilde{\lambda}$, $\tilde{\sigma}$;

    for $g \in \mathcal{G}$ **do** add new schedules from opt. subproblem solution to SC heuristic;

    for *the best subproblem solutions* $i = 1, \ldots, N$ **do**

      calculate $\bar{c}_{si}$;

      if $\bar{c}_{si} < 0$ **then** generate a cut and add it to dRMP(ϵ);

    end

  end

  calculate v_k^{lb} from (30) and δ_k^{MP} from (31);

  if $v_k^{lb} > v_{k-1}^{lb}$ **then** set $\hat{\lambda} \leftarrow \tilde{\lambda}$;

until $\delta_k^{MP} < \Delta$;

Output: dual solution $\hat{\lambda}$, primal solution x^*, MP objective value v^{ub};

Figure 2: Scenario decomposition method. The algorithm consists of an initialisation loop and a main loop. The main loop contains the schedule combination heuristic which is repeated every r iterations, and the stabilised ColGen logic.

5. Numerical Experiments

We implemented the ColGen procedure shown in Figure 2 and tested it on two-stage and multi-stage versions of a central scheduling model of the British power system with 24 hour-long time steps and up to 50 wind power scenarios. The largest problems (50 scenario multi-stage cases) have roughly 1.08M variables (350k binaries), 1.5M constraints and 5.9M non-zeros. The model has a 30% wind penetration in terms of installed capacity, 130 thermal units and four pump storage plants with a total of 16 pump-turbines. It uses a loss-free real power transmission network model with 17 zones and 27 transmission links between them. Additional restrictions limit the sum of transmissions across a pre-defined set of 17 boundaries. In order to ensure as realistic a test environment as possible, the generator data, pump storage model, reserve requirements and transmission restrictions are based on assumptions which suit the GB power system. Some of these are described as part of regular publications by National Grid. We provide a detailed description of the model and reference to the various data sources in [28]. In this application all the uncertainty arises from the unpredictability of wind generation. However, the decomposition technique applies equally well to cases with unpredictable demands or contingency scenarios. The following sections give details of our implementation and summarise our experience with the method.

5.1. Details of the Implementation

Algorithm 2 is implemented as a script in AMPL version 20120629 [37]. All LPs and MIPs are solved with CPLEX version 12.4 [38]. We perform the tests on a Dual 8 Core Dell Power Edge C6220 machine with 128 Gb RAM, running 64 bit Linux. The pricing problems are solved in serial, as parallel implementations are not supported by AMPL. However, we use the parallel option for CPLEX, with a maximum of 16 threads. Up to 50 cuts are added to dRMP(ϵ) per iteration per subproblem if multiple solutions were found. We set the overall optimality tolerance to $\Delta = 0.1\%$ and the subproblem tolerance to 0.05%, so that at most half of the overall gap is due to the subproblem MIP gaps. The overall gap strikes a balance between computational effort and a sensible accuracy for scheduling under wind uncertainty: on average, 0.1% of the daily cost correspond to increasing the output by 36MW for the duration of a day. In comparison, the uncertainty in a 3h wind forecast is 840MW, while 24h ahead it is already 2.8GW. The SC problem is solved to an optimality tolerance of 0.01%.

5.2. Multi-Stage Results

Figure 3 shows timings for solving multi-stage stochastic problems with the ColGen method, and solving the extensive form of the same model with CPLEX. We vary the number of scenarios between 3 and 50 and the number of stages between 2 and 4, where each stage covers 3 hours and the final stage covers the remainder of the 24 hours. On small problems, CPLEX and our decomposition work similarly well, while on larger instances the decomposition is superior. The factor by which the ColGen method outperforms CPLEX increases roughly linearly in the number of included scenarios. The behaviour of the decomposition method suggests that the set of multi-stage test problems can be separated in two groups on which different strategies lead to faster convergence:

1. Problems with small EVPI. For these 'easy' problems, $\hat{\lambda} = 0$ is a good estimate, so LPR(μ) is not solved. The SC heuristic finds sufficiently good primal solutions to satisfy our optimality criteria after being populated initially with schedules from a ColGen pass with zero multipliers. Then, since the first lower bound is the expected value under perfect information, the method terminates after a single iteration.

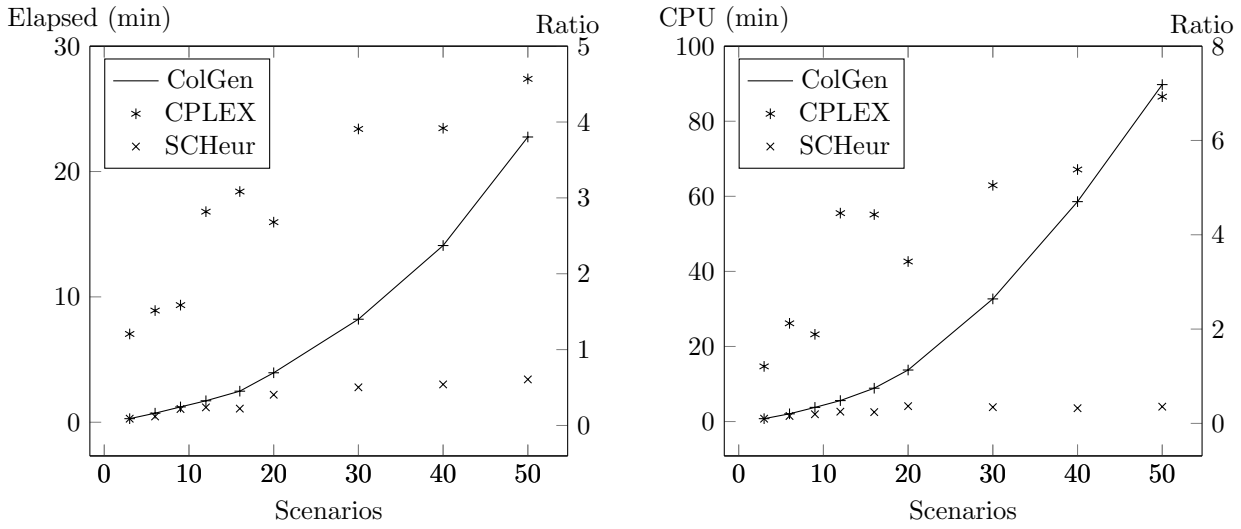


Figure 3: Elapsed times (left) and CPU times (right) to solve the multi-stage stochastic problem to within 0.1% of optimality, as a function of the number of scenarios. Times shown are the average of four different instances of each problem size. For the ColGen method the total time requirement is shown (left y-axes). Additionally, the timings for solving the extensive formulation via CPLEX and for solving the SC problem (SCHeur) are shown as multiple/proportion of the time required by the ColGen method (right y-axes).

2. Problems with large EVPI. If attempted to solve with zero initial multipliers, these 'hard' problems have gaps between 0.5% and 1% after the first iteration. However, if we solve them with multiplier estimates from LPR(μ), we still achieve a tolerance of 0.1% in one iteration.

In the second case, the gap with zero multipliers is firstly due to a worse lower bound, i.e. the EVPI being non-negligible, and secondly due to a non-optimal primal solution obtained from the SC problem. The feasible region of the SC problem does not contain an optimal primal solution unless we provide a good multiplier estimate to the subproblem pass from which we populate the SC problem. However, estimating these multipliers with LPR(μ) is time consuming for large instances, and where possible we avoid this by applying the following heuristic rule to separate easy problems from hard ones: we first solve the subproblems with zero multipliers and use the overcommitment heuristic to produce a primal solution. This provides an upper bound and a Lagrangian lower bound, and if the gap between them exceeds 1% then the problem is hard, otherwise it is easy. For easy problems we continue with the ColGen algorithm by solving the SC heuristic, while for hard problems we first solve LPR(μ) and re-solve the subproblems before continuing. The separation of easy and hard problems works well on all test cases: on hard problems the gap between overcommitment solution and first lower bound was always at least 3%, while easy problems seldomly reached 0.6%.

To estimate the proportion of hard and easy problems in our GB model, we evaluate a pool of 3,000 multi-stage problems obtained from a long term rolling horizon study [28]. This showed that roughly 25% of the problems are hard, while 75% are easy. The timings in Figure 3 are weighted accordingly: every data point corresponds to the average solve time of one hard problem and three easy ones, i.e. they represent an expected solve time for an 'average' problem.

Solving hard problems via decomposition requires solving LPR(μ) with CPLEX's barrier solver, whose CPU[3] timings scale unfavourably in the number of scenarios (cf. next section), and the overall CPU times are affected by this. The other major factor that contributes to the time spent in the decomposition is the SC problem, and solution times for that are shown separately in Figure 3. The amount of parallelism in the decomposition increases slightly in the number of scenarios: the CPU to elapsed time ratio is between 3:1 on small problems and 4:1 on larger ones, and this increase is due to the barrier solver which we use to solve LPR(μ) in the hard cases. Further parallel speedups are possible if the subproblems are solved in parallel. Without decomposition, CPLEX achieves a larger increase in parallelism with the number of scenarios: the CPU to elapsed time ratio increases from 3:1 to 6:1, so the superiority of the decomposition is more evident on the CPU time graph.

The fact that the first run of the SC heuristic often provides a good enough solution to terminate suggests trying another simple way of obtaining a bound: pass the SC solution to CPLEX and ask it to solve the extensive form with the given starting point, by solving the root node LP relaxation and adding MIP cuts to tighten the bound. The resulting timings, however, are not better than when solving the whole problem via CPLEX without an initial solution, so we omit them here.

5.3. Two-Stage Results

Figure 4 shows elapsed times and CPU times required to solve test instances of two-stage SUC problems with 3 to 50 scenarios. As above, we compare the scenario decomposition method to solving the extensive form via CPLEX. For the extensive form we use a single set of first stage variables instead of scenario-specific copies and non-anticipativity constraints. In general solution times are longer than in the multi-stage setting. Again, the factor by which the decomposition outperforms CPLEX increases roughly linearly in the number of included scenarios.

In our test set there were no two-stage cases with negligible EVPIs, so all cases are 'hard' for the decomposition. Our ColGen method still converged in the first iteration, however, to achieve this it was not sufficient to initialise the duals to zero. With zero duals, the optimality gap achieved after solving the RMP for the first time is typically 1.5% or worse. To find good initial dual estimates, we solve LPR(μ). Figure 4 shows the proportion of ColGen time spent for solving LPR(μ) and the SC heuristic. Both contribute significantly to the overall CPU time required by the decomposition, and the time saving resulting from using decomposition instead of CPLEX is smaller than in the multi-stage case. In terms of elapsed time, however, the contribution from LPR(μ) to the overall ColGen time is smaller. Also, the ratio of decomposition CPU time to elapsed time is now higher: it varies between 4:1 in the smallest case and 6:1 in the largest case. This is due to CPLEX's barrier solver which achieves high parallel speed-ups when used to solve LPR(μ).

Since most of the ColGen CPU time is spent solving LPR(μ), we experiment with the following *alternative* method to estimate initial duals by solving the relaxation in a decomposed way:

1. Apply ColGen with $\hat{\lambda} = 0$ initially, but relax integrality restrictions of the subproblems. Solve this relaxation to a gap of $\Delta = 0.05\%$.

2. Use the final multipliers as initial stability center $\hat{\lambda}$ for the integer decomposition.

[3]We show the sum of CPU times on all cores.

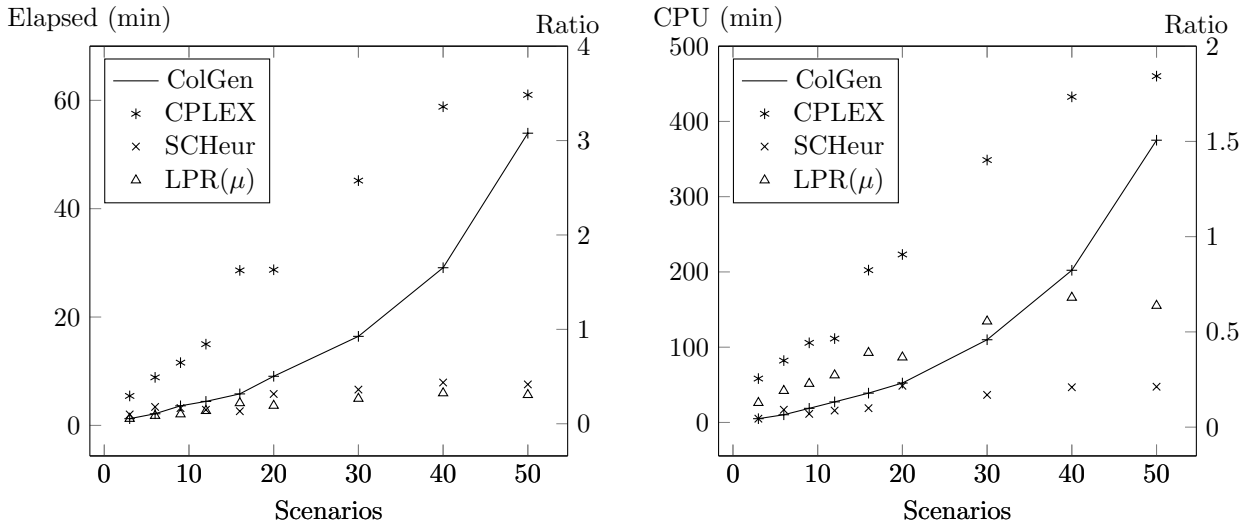


Figure 4: Elapsed times (left) and CPU times (right) to solve the two-stage stochastic problem to within 0.1% of optimality, as a function of the number of scenarios. Times shown are the average of three different instances of each problem size. For the ColGen method the total time requirement is shown (left y-axes). Additionally, the timings for solving the extensive formulation via CPLEX, solving the SC problem (SCHeur) and solving LPR(μ) are shown as multiple/proportion of the time required by the ColGen method (right y-axes). SCHeur and LPR(μ) cause the majority of the time spent in the decomposition. For LPR(μ) we used $\mu = 10^{-6}$.

The resulting duals are a better initial point for the integer decomposition than zero: the gap obtained after solving the first master problem is typically 0.3%. Smaller gaps can be achieved by solving the relaxation to a better accuracy than 0.05%, however the convergence of the relaxed decomposition is slow, and solving the LP relaxation to a sufficient tolerance so as to obtain multipliers which yield a 0.1% bound in the first integer ColGen iteration is achieved faster by solving LPR(μ) exactly with CPLEX's barrier solver. Therefore the results in Figure 4 show timings for the exact approach.

As before, the SC heuristic scales well in the number of scenarios. However, it requires good dual estimates: with schedules generated from a ColGen pass with $\lambda = 0$, the SC heuristic solution is up to 0.2% worse than with dual estimates from LPR(μ).

Solution times for our set of two-stage problems are longer than those for multi-stage problems – regardless of the solution method. From the point of view of scenario decomposition this is intuitive since the number of non-anticipativity constraints is larger and they cover later time periods where the scenario sets are more diverse, therefore making it harder to find a non-anticipative 'compromise' solution since the scenarios favour different solutions. The EVPI is a suitable measure for this effect, since it captures the degree of conflict between the scenarios and expresses it in terms of expected costs. It may be that the degree of conflict among scenarios is also what leads to more branching in the Branch & Cut algorithm and therefore makes the problems harder for CPLEX.

5.4. Convergence of Bounds

In the tests described in the previous sections, all problems were solved after a single ColGen pass. The major computational work was in estimating a dual solution from LPR(μ) and constructing a primal solution with the heuristic. While this is sufficient to obtain 0.1% gaps, more work is required to achieve smaller gaps. In the following we briefly discuss convergence properties of CPLEX and the decomposition method with a

target of 0.01%. In the decomposition this requires that subproblems are solved to a gap of at most 0.01%. Figure 5 shows the convergence of upper and lower bounds as a function of elapsed time on a typical 50 scenario example.

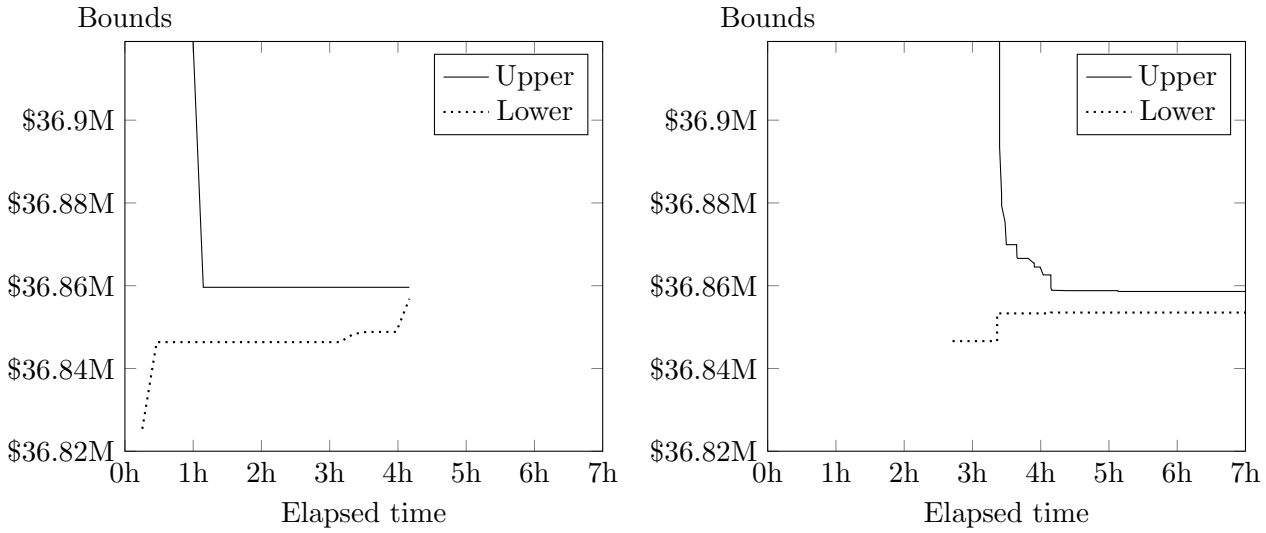
The decomposition obtains the first lower bound after 15 minutes by solving LPR(μ) with CPLEX's barrier method. The lower bound is improved swiftly through a pass of ColGen, i.e. an evaluation of the Lagrangian dual. The first upper bound is obtained immediately thereafter, when an initial solution is constructed for the SC heuristic. This bound is not shown on the graph since it exceeds \$36.9M. The upper bound is first improved when the SC heuristic finds a solution of roughly \$36.86M which is also the final solution the decomposition terminates with. At that stage the gap is below 0.1%, but it takes multiple iterations of column generation before the lower bound is raised further. Finally, a gap of 0.007% is achieved after four hours, and the method terminates.

CPLEX achieves the first lower bound after 2.5 hours, by solving the root relaxation with a simplex method and adding the first set of MIP cuts. After another hour of root node processing, the cut generation procedure improves the lower bound, and the MIP heuristics produce the first good solution, which is then gradually improved. After 4 hours, CPLEX achieves a solution that is within 0.013% of optimality, however the lower bound cannot be raised any further by adding MIP cuts. Branching is required to close the gap, and this takes a very long time. After 12 hours, we interrupt CPLEX, and at this stage the bounds have not changed in comparison to those shown on the graph: the procedure has stalled.

The example demonstrates the strengths of the decomposition in comparison to CPLEX's standard Branch & Cut procedure. The first bound and a near-optimal solution are obtained early in the process, resulting in an initial gap below 0.1%. CPLEX, on the other hand, requires a lot longer to produce the first solution and lower bound. To achieve the 0.1% tolerance, it requires roughly 3.5 hours. Following the initial effort, the decomposition takes a while to converge to the stricter 0.01% tolerance, and CPLEX achieves almost the same accuracy in the same overall time (4 hours), by further improving the primal solution. However, no more progress in the lower bound is achieved, so that it stalls with a gap of 0.013%. At all times the gap of the decomposition method is smaller than that of CPLEX.

We further explore the behaviour of CPLEX and the decomposition method when converging them to very small gaps, by applying them to one of the test problem sets used to obtain the results in Figure 4, with a target tolerance of 0.005%. Both CPLEX and the decomposition achieve the target for all but the two largest cases. On the large cases, CPLEX stalls at an average gap of 0.01% and the decomposition at 0.006% due to subproblem non-optimality. On cases where the methods did not stall, the elapsed time in the decomposition algorithm was on average half the time spent in CPLEX. On average, ten ColGen iterations were required.

Lower Bounds and Optimal Cutting Planes. One of the major steps in the convergence process of the decomposition is to construct a near-optimal primal solution, and doing this is useful in its own right, since ultimately it is that solution which we are after. Besides that, however, we also explore the effect of this near-optimal cutting plane on the lower bound obtained with the duals produced by dRMP(ϵ). To do this, we run the ColGen method without providing dRMP(ϵ) with a cutting plane derived from a heuristically constructed primal solution. We compare the resulting lower bound to the bound that would have been obtained from the first solve of dRMP(ϵ) if the near-optimal cutting plane had been included. The same stability center is used both times. For two-stage problems, the bounds without the cutting plane are between 0.5% and 3% worse than the



(a) Convergence of bounds in ColGen.

(b) Convergence of bounds in CPLEX.

Figure 5: Convergence of upper and lower bounds in decomposition (5a) and in CPLEX (5b), as a function of elapsed time. The shown example is one of the two-stage 50 scenario cases that were evaluated for the performance comparison in Figure 4, however in this case we solve it to a much smaller gap.

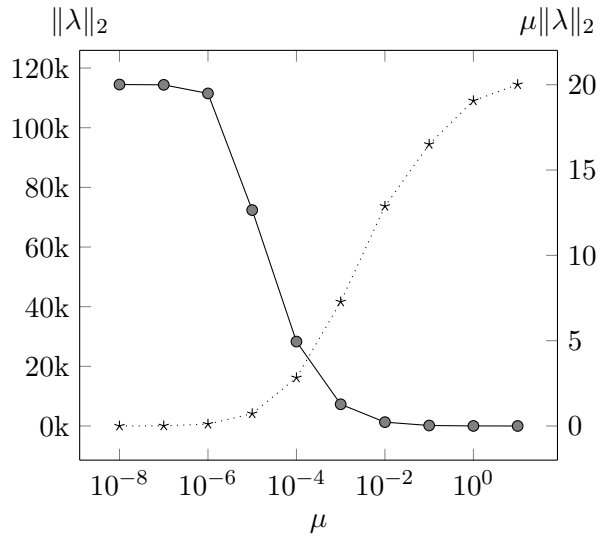
ones obtained with it. In multi-stage problems, excluding the optimal plane leads to even worse lower bounds, some of which are negative. This demonstrates that knowing a very good primal solution is also beneficial to the convergence of dual bounds.

5.5. Initial Multiplier Estimates

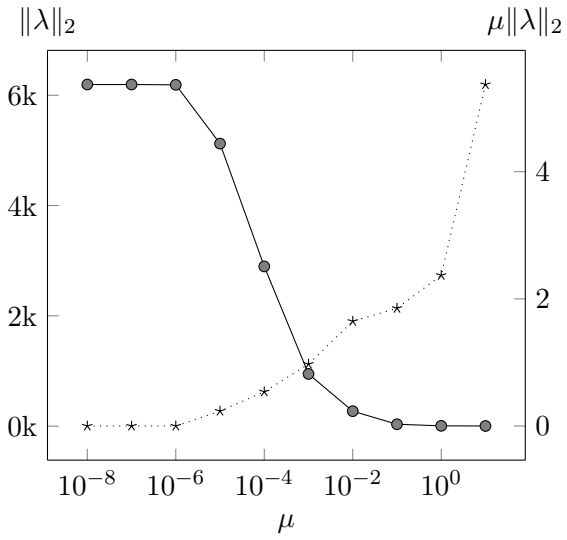
Due to the redundancy of non-anticipativity constraints, their dual optimal solutions are degenerate in the SUC problem and its relaxation. In Section 4.3 we introduced LPR(μ) with $\mu \geq 0$ as a means of controlling the size of the initial multipliers. For any small enough μ the obtained dual solution is *optimal* for the relaxation of the SUC problem, i.e. if we solve the *relaxed* subproblems with these duals, we are guaranteed to obtain a tight Lagrangian lower bound for the SUC relaxation. The results described in the previous sections suggest that, at least with the employed value of $\mu = 10^{-6}$, the optimal duals from LPR(μ) are also optimal for the *integer* SUC problem, and the duality gap vanishes or is negligibly small. In this section we explore whether this is the case for other values of μ, including zero.

Figure 6 shows the norm of the optimal duals of LPR(μ) as a function of μ for a multi-stage and a two-stage example. On the secondary axis, we also map the non-anticipativity constraint violation resulting from the same value of μ. The duals are stable for various μ, but approach zero as the perturbation level is increased and the non-anticipativity violation increases. The duals obtained with $\mu = 0$ are not included on the graphs for scaling issues: in the two-stage case their norm was twice as large as the largest shown value, and in the multi-stage case it was 50 times larger. For $\mu > 0$ the duals obtained from either case are quite different: there are more duals in the two-stage problem, covering all scenarios and time steps, and individual entries are typically larger, resulting in a much larger norm than in the multi-stage case.

In the two-stage example, we observe that any of the duals obtained with $\mu \in [0, 10^{-5}]$ provide a good enough lower bound to terminate after the first ColGen pass. For any values larger than that, the non-anticipativity violation increases swiftly, and the obtained bounds are inferior, so that the decomposition does not terminate



(a) Effect of μ on λ in LPR(μ) (two-stage).



(b) Effect of μ on λ in LPR(μ) (multi-stage).

Figure 6: Further results concerning the choice of μ in LPR(μ) in the two-stage case (6a) and the multi-stage case (6b). The graphs show the effect of μ on the magnitude of the dual solution $\|\lambda\|_2$ (left axes) and on the violation of non-anticipativity constraints $\mu\|\lambda\|_2$ (right axes) in LPR(μ). The values shown here are taken from a nine scenario example.

immediately. Too large a choice of μ relaxes the non-anticipativity constraints and removes essential information from the multipliers, which is also needed to encourage the subproblems to produce useful schedules: with $\mu > 10^{-5}$ the primal solutions found by the SC heuristic become increasingly worse.

In contrast to the two-stage example, the shown multi-stage example is one of the 'easy' cases for which zero duals produce a sufficiently tight lower bound, due to a negligible EVPI. In this case any of the values for μ shown on the graph provided a dual solution which proved optimality of the primal solution in the next ColGen pass. The same is true for duals obtained with $\mu = 0$. The degenerate dual solutions of vastly different size extracted from LPR(μ) all produce tight bounds for the integer SUC problem.

Since the duals estimated from LPR(0) were sufficient to obtain a gap of 0.1% after the first ColGen iteration for both two-stage and multi-stage problems, it may appear unnecessary to control their size by choosing $\mu > 0$.

However, there are two reasons why we found it preferable to work with $\mu > 0$:

- When more than one ColGen iteration is necessary, the gap obtained after the first run of dRMP(ϵ) is significantly better if the initial stability center is obtained by solving LPR(μ) with $\mu > 0$. Many of the cases shown in Figure 4 obtained very poor lower bounds if dRMP(ϵ) was solved with a stability center provided by LPR(0), and it took about ten subsequent iterations to find the next multipliers which improved the bound beyond 0.1%.

- CPLEX's barrier method solves LPR(μ) faster with $\mu > 0$ rather than $\mu = 0$. On cases with ≥ 30 scenarios this saves up to a third of the required CPU time.

6. Conclusion & Further Research

We derived a mixed-integer column generation algorithm for two-stage and multi-stage stochastic unit commitment problems, based on a Dantzig-Wolfe decomposition of the underlying scenario set. In theory, DW

decomposition readily applies to scenario decomposition, but a series of additional issues arise in practical applications. We stabilise the scenario decomposition algorithm with a dual proximal bundle term and estimate dual hot-starts from a perturbed LP relaxation. We also derive a fast, novel, MIP-based schedule combination heuristic which is used to construct primal solutions. Our tests suggest that knowing this solution at the start of the column generation process also enables quick convergence of lower bounds. The scenario decomposition method can solve SUC problems to optimality if optimal solutions of the deterministic subproblems can be obtained. Where this is not the case, a valid lower bound can be derived from the lower bounds of the subproblems. The ability to achieve small gaps is a definite advantage over alternative decompositions such as Progressive Hedging: in all test runs the duality gaps vanish or are well below a 0.1% optimality tolerance. And no branching is necessary, since sufficiently good primal solutions can be constructed by a heuristic.

Our implementation was tested on two-stage and multi-stage instances of a model based on the British power system. The ColGen method terminates in just one iteration and takes significantly less time than solving the extensive formulation directly. The key reasons why this works well are the primal and dual warm-starts.

Dual warm starts are required for cases with large EVPI, to populate the SC heuristic so that its solution space contains an optimal solution of the original problem, and to obtain tight lower bounds. The stabilised LP relaxation LPR(μ) can be solved to obtain such multipliers, but in terms of CPU time CPLEX's barrier solver does not scale well on LPR(μ) if the number of scenarios is increased. Approximate dual solutions can be obtained quickly by applying scenario decomposition to the LP relaxation, but the lower accuracy has an adverse effect on the quality of the obtained lower bounds. How to solve LPR(μ) quickly *and* accurately remains an open question and a potential direction of further research.

The other key element in our method are the primal warm starts provided by the SC heuristic. The heuristic scales very well on the problems we solved for this study, but since it is a MIP itself, it will fail to do so eventually if the problem size is increased sufficiently. In that case cheaper heuristics will be required. The SC heuristic can be used as a starting point to develop cheaper heuristics, and this is another possible direction of future research.

The strategy adopted in this study aims to reduce the work required to solve SUC problems: we seek accurate primal and dual hot starts to minimise the number of ColGen iterations. This results in a favourable reduction of CPU time, which is suitable for environments with moderate computational power. Alternative strategies may seek to reduce elapsed times by increasing parallelism: subproblems can be solved in parallel, and each subproblem can again be solved by parallel Branch & Bound. The resulting ColGen iterations are cheaper, and it may be worthwhile to reduce the accuracy of the initial duals at the expense of doing more iterations. Less accurate duals can then be obtained via scenario decomposition of the initial LP relaxation, which can again be parallelised. This may be a more suitable strategy if large computer clusters are available, or even if less accurate lower bounds are required.

Acknowledgements

We would like to thank the reviewers for their detailed comments and constructive suggestions which helped us to improve this paper. The first author acknowledges funding through the Principal's Career Development Scholarship scheme of the University of Edinburgh.

References

[1] C. C. Carøe, R. Schultz, A two-stage stochastic program for unit commitment under uncertainty in a hydro-thermal power system, Tech. Rep. SC 98-11, Konrad-Zuse-Zentrum für Informationstechnik (1998).

[2] J. Goez, J. Luedtke, D. Rajan, J. Kalagnanam, Stochastic unit commitment problem, Tech. Rep. RC24713 (W0812-119), IBM Research Division (2008).

[3] N. Gröwe-Kuska, W. Römisch, Stochastic Unit Commitment in Hydro-thermal Power Production Planning, Preprints aus dem Institut für Mathematik, Humboldt Universität Berlin, 2002, Ch. 30, pp. 605–624.

[4] M. P. Nowak, W. Römisch, Stochastic Lagrangian relaxation applied to power scheduling in a hydro-thermal system under uncertainty, Annals of Operations Research 100 (1-4) (2000) 251–272.

[5] S. Takriti, J. R. Birge, E. Long, A stochastic model for the unit commitment problem, IEEE Transactions on Power Systems 11 (3) (1996) 1497–1508.

[6] A. Papavasiliou, S. S. Oren, R. P. O'Neill, Reserve requirements for wind power integration: A scenario-based stochastic programming framework, IEEE Transactions on Power Systems 26 (4) (2011) 2197–2206.

[7] A. Papavasiliou, S. S. Oren, Multiarea stochastic unit commitment for high wind penetration in a transmission constrained network, Operations Research 61 (3) (2013) 578–592.

[8] K. Cheung, D. Gade, C. Silva-Monroy, S. M. Ryan, J. P. Watson, R. J. B. Wets, D. L. Woodruff, Toward scalable stochastic unit commitment part 2: Solver configuration and performance assessment, Energy Systems 6 (3) (2015) 417–438.

[9] D. Rajan, S. Takriti, Minimum up/down polytopes of the unit commitment problem with start-up costs, Tech. Rep. RC23628 (W0506-050), IBM Research Division (2005).

[10] M. Carrión, J. M. Arroyo, A computationally efficient mixed-integer formulation for the thermal unit commitment problem, IEEE Transaction on Power Systems 21 (3) (2006) 1371–1378.

[11] R. Jiang, Y. Guan, J.-P. Watson, Cutting planes for the multi-stage stochastic unit commitment problem, Tech. Rep. SAND2012-9093J, Sandia National Laboratories (2012).

[12] J. Ostrowski, M. F. Anjos, A. Vannelli, Tight mixed integer linear programming formulations for the unit commitment problem, IEEE Transactions on Power Systems 27 (1) (2012) 39–46.

[13] E. Krall, M. Higgins, R. P. O'Neill, RTO unit commitment test system, Tech. Rep. AD10-12-002, Federal Energy Regulatory Commission (FERC) (2012).

[14] S. J. Wang, S. M. Shahidehpour, D. S. Kirschen, S. Mokhtari, G. D. Irisarri, Short-term generation scheduling with transmission and environmental constraints using an augmented Lagrangian relaxation, IEEE Transactions on Power Systems 10 (3) (1995) 1294–1301.

[15] T. Shiina, J. R. Birge, Stochastic unit commitment problem, International Transactions in Operational Research 11 (1) (2004) 19–32.

[16] O. Briant, C. Lemaréchal, P. Meurdesoif, S. Michel, N. Perrot, F. Vanderbeck, Comparison of bundle and classical column generation, Mathematical Programming 113 (2) (2008) 299–344.

[17] G. Hechme-Doukopoulos, S. Brignol-Charousset, J. Malick, C. Lemaréchal, The short-term electricity production management problem at edf, Optima 84 (2010) 2–6.

[18] R. T. Rockafellar, R. J. B. Wets, Scenarios and policy aggregation in optimization under uncertainty, Mathematics of Operations Research 16 (1) (1991) 119–147.

[19] J. P. Watson, D. Woodruff, Progressive hedging innovations for a class of stochastic resource allocation problems, Computational Management Science 8 (4) (2011) 355–370.

[20] D. Gade, G. Hackebeil, S. M. Ryan, J. P. Watson, R. J. B. Wets, D. L. Woodruff, Obtaining lower bounds from the progressive hedging algorithm for stochastic mixed-integer programs, Mathematical Programming 157 (1) (2016) 47–67.

[21] C. C. Carøe, J. Tind, L-shaped decomposition of two-stage stochastic programs with integer recourse, Mathematical Programming 83 (1-3) (1998) 451–464.

[22] E. Ogbe, X. Li, A new cross decomposition method for stochastic mixed-integer linear programming, European Journal of Operational Research 256 (2) (2017) 487–499.

[23] Q. P. Zheng, J. Wang, P. M. Pardalos, Y. Guan, A decomposition approach to the two-stage stochastic unit commitment problem, Annals of Operations Research 210 (1) (2013) 387–410.

[24] B. Ghaddar, J. Naoum-Sawaya, A. Kishimoto, N. Taheri, B. Eck, A lagrangian decomposition approach for the pump scheduling problem in water networks, European Journal of Operational Research 241 (2) (2015) 490–501.

[25] C. Sagastizábal, Divide to conquer: decomposition methods for energy optimization, Mathematical Programming 134 (1) (2012) 187–222.

[26] G. Lulli, S. Sen, A branch-and-price algorithm for multistage stochastic integer programming with application to stochastic batch-sizing problems, Management Science 50 (6) (2004) 786–796.

[27] J. Higle, B. Rayco, S. Sen, Stochastic scenario decomposition for multi-stage stochastic programs, IMA Journal of Management Mathematics 21 (1) (2009) 39–66.

[28] T. Schulze, K. McKinnon, The value of stochastic programming in day-ahead and intraday generation unit commitment, Energy 101 (1) (2016) 592–605.

[29] F. Vanderbeck, Implementing mixed integer column generation, in: Column Generation, Springer, 2005, pp. 331–158.

[30] F. Vanderbeck, M. W. P. Savelsbergh, A generic view of Dantzig-Wolfe decomposition in mixed integer programming, Operations Research Letters 34 (3) (2006) 296–306.

[31] W. K. Klein Haneveld, M. H. van der Vlerk, Stochastic integer programming: General models and algorithms, Annals of Operations Research 85 (0) (1999) 39–57.

[32] M. Lubin, J. A. J. Hall, C. G. Petra, M. Anitescu, Parallel distributed-memory simplex for large-scale stochastic lp problems, Computational Optimization and Applications 55 (3) (2013) 571–596.

[33] M. E. Lübbecke, J. Desrosiers, Selected topics in column generation, Operations Research 53 (6) (2005) 1007–1023.

[34] C. Lemaréchal, Lagrangian relaxation, in: Computational Combinatorial Optimization, Springer, 2001, pp. 112–156.

[35] S. Takriti, J. R. Birge, Using integer programming to refine Lagrangian-based unit commitment solutions, IEEE Transactions on Power Systems 15 (1) (2000) 151–156.

[36] T. Schulze, Stochastic programming for hydro-thermal unit commitment, Ph.D. thesis, The University of Edinburgh, `http://hdl.handle.net/1842/15775` (2015).

[37] R. Fourer, D. M. Gay, B. W. Kernighan, AMPL – A Modeling Language for Mathematical Programming, Brooks/Cole, 2003.

[38] IBM Ilog, IBM Ilog CPLEX v.12.1 – User's Manual for CPLEX (2009).