PLAN4RES : Synergistic Approach of Multi-Energy Models for an European Optimal Energy System Management Tool

Deliverable D5.4

Improvement of the decomposition-based primal heuristic framework

Deliverable No.	D5.4	Work Package No.	WP5	Task No.	Subtask 5.1.1.
Work Package Title		Methods and Algorithms			
Linked Task/s Title		Problem-specific primal heuristics for time-indexed MIP formulations			
Status		Final	Draft / Draft Final / Final		
Dissemination level		PU	PU-Public / CO-Confidential		
Due date deliverable		2019-11-30	Submission date		2019-11-26
Deliverable version		`Plan4Res_D5.4_name_vx.doc`			

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 773897

Deliverable Contributors:	Name	Organisation	Date
Deliverable Leader	Antonio Frangioni	ICOOR (UniPi)	
Work Package Leader	Ambros Gleixner	ZIB	
Contributing Author(s)	Laura Galli	ICOOR (UniPi)	
Reviewer(s)	Wim van Ackooj Inci Yueksel-Erguen	EDF ZIB	
Final Review and Approval	Sandrine Charousset	EDF	

History of Changes:

Release	Date	Reason for Change	Status
0.1	19/11/ 2019	Initial Version	Draft
1.0	22/11/2019	Incorporating Reviewers' comments	Draft Final
1.0	25/11/2019	Changes approved by Reviewers	Final

List of Figures

List of Tables

List of acronyms used in this document

- EUC: European Unit Commitment
- UC: Unit Commitment
- MILP: Mixed-Integer Linear Program
- SSV: Seasonal Storage Valuation
- NDO: NonDifferentiable Optimization

DISCLAIMER / ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 773897

Executive Summary

The goal of plan4res is to develop a modelling framework that allows to obtain a holistic assessment of the energy system. Several (possibly, nested) stages of the framework posses different forms of structure. It is very common that some of these are "orthogonal", as in the case of a "horizontal" structure regarding spatial partitioning of decisions (e.g., those corresponding to a certain unit and/or geographical area) and the "vertical" one regarding temporal partitioning of decisions (e.g., those corresponding to a certain hour or day). This structure lends itself well to decomposition-based primal heuristics, that have already been used successfully in energy optimization such as deterministic [1, 11, 12, 13] and stochastic [19] Unit Commitment problems.

However, previous incarnations of these approaches have been highly problem-specific. This starts from the fact that already decomposition, even using the `NDOSolver / FiOracle` software framework, required a substantial amount of coding. Even more so when constructing the heuristic itself, which requires (or at least highly benefits from) using specialized approaches for solving the subproblems corresponding to each "time instant".

The unique modelling capabilities of the `SMS++` framework make it possible to implement a more general version of the decomposition-based primal heuristic that is largely (albeit not completely) independent from the fine details of the model to be solved, while retaining the ability to use specialized solvers for subproblems, both in the 'horizontal" and in the "vertical" structure. This will be crucial for implementing the "The EUC simulation mode" foreseen in the project [16, §1.1.3].

The aim of this document is to describe the basic algorithmic ideas of the decomposition-based primal heuristic and how specific capabilities of the `SMS++` framework allow it to be implemented in a flexible and general, yet efficient, way. Complete documentation of the "master" branches of the `SMS++` framework at this time is included in order to provide complete references to the specific software components recalled in the presentation.

All the material described in this report is available at the "umbrella" repository

`https://gitlab.com/smspp/smspp-project`

which relies on the individual repositories

`https://gitlab.com/smspp/smspp`
`https://gitlab.com/smspp/milpsolver`
`https://gitlab.com/smspp/ucblock`
`https://gitlab.com/smspp/mcfblock`
`https://gitlab.com/smspp/smilpblock`
`https://gitlab.com/egorgone/bundlesolver`
`https://gitlab.com/frangio68/ndosolver_fioracle_project`
`https://github.com/frangio68/Min-Cost-Flow-Class`

The documentation is also available online at

`https://smspp.gitlab.io/smspp/`

Key Words: Heuristics, Time-indexed MIP

Chapter 1

Introduction

Many problems in large-scale optimization, and in particular several related to the energy system, have several (possibly, nested) form of structure. Due to their large size and intrinsic difficulty, exploiting all these structures is crucial for efficiently approaching them. However, this is also difficult in that the structures can be "orthogonal". That is, if a structure-exploiting approach like a decomposition method is used which exploits one of them, typically the same approach is not capable of tackling the other. This is a severe limitation for general-purpose solvers. Indeed, while supporting some form of decomposition (Lagrangian relaxation a.k.a. column generation, Benders' decomposition) has recently become more common, general-purpose solvers focus on at most one structure at a time, thereby necessarily neglecting the other(s).

The unique modelling capabilities of `SMS++` allow to design general and flexible, yet efficient, heuristic schemes capable of exploiting multiple forms of structure at once.

To discuss the main algorithmic ideas of such a scheme we refer to the following idealised representation of the target model:

$$\min \sum_{i \in I} \sum_{t \in T} c_{it}(u_{it}, x_{it}) \tag{1.1}$$

$$(u_{it}, x_{it}) \in X_{it} \qquad i \in I \;\;,\;\; t \in T \tag{1.2}$$

$$(u_{it-1}, x_{it-1}, u_{it}, x_{it}) \in H_{it} \qquad i \in I \;\;,\;\; t \in T \tag{1.3}$$

$$\sum_{i \in I} V(u_{it}, x_{it}) = v_t \qquad t \in T \tag{1.4}$$

$$u_{it} \in \mathbb{Z}^{n_{it}} \qquad i \in I \;\;,\;\; t \in T \tag{1.5}$$

Such model has two distinct forms of structure. We distinguish an "horizontal" structure, denoted by the index set I, which typically regards some spatial partitioning of decisions (e.g., those corresponding to a certain unit and/or geographical area i), and a "vertical" one, denoted by the index set T, which typically regards some temporal partitioning of decisions (e.g., those corresponding to a certain hour or day t). Note that, assuming the first time instant in T is denoted by $t = 1$, constraints (1.3) make reference to *fixed* values (u_{i0}, x_{i0}); these represent the state of the system at the beginning of the time horizon, and typically constrain its behaviour in the initial time step(s). Furthermore, we single out the further structure corresponding to the fact that while some variables represent continuous decisions (x_{it}), other represent discrete/logical ones (u_{it}).

This structure is found in many energy optimization contexts such as deterministic [1, 11, 12, 13] and stochastic [19] Unit Commitment (UC) problems. There, T is a set of time instants in a (usually, short-term) horizon (say, hours in a day or week), while I is a set of geographically dispersed generating units. Each unit has some continuous output (amount of power and possibly reserve that it generates) which is subject to complex constraints, requiring logical conditions to be checked. For instance, if a unit starts producing at some time instant, it typically has to keep on producing for a given number of subsequent time instants (min up-time constraints, and similarly for down-time). Also, the power produced by a unit (when on) cannot change too much from a time instant to the next one (ramp-up and ramp-down constraints). In this application, the fixed values (u_{i0}, x_{i0}) represent the commitment (on/off) status and output power of the units prior to the beginning of the time horizon, which are necessary to ensure that min up- and down-time constraints and ramp-up/ramp-down ones are satisfied in the initial time instant(s). To keep with this motivation, which is very relevant for the plan4res project, we will generally refer to I as the set of "units".

The shape of the model is characterized by the fact that decisions pertaining to an unit at some point in time t have a relatively short "direct" effect on decisions for the same unit at a different instant t'; we represent this in the simplified form (1.3) where only consecutive time instants directly interacts, although more general cases may have to be considered (however, note that the definition of T is totally flexible, meaning that t could actually represent a set of consecutive time instants). On the other hand, the "global" constraints (1.4) link decisions for all units corresponding to the same time instant; while they provide the spatial coupling, they are typically not temporally coupled. In the UC case they correspond e.g. to satisfaction of global energy/reserve demand, possibly mediated by the transmission network.

The horizontal structure lends itself well to decomposition approaches, in particular Lagrangian ones; in fact, these have a very long history of applications in UC in particular (cf. [20] and the many references therein). The vertical structure rather lends itself to decomposition approaches of the (nested) Bender's type, which not coincidentally also have been applied many times to UC (cf. again [20]). However, combining the two approaches in a general way is far from obvious. This is partly due to the inherent complexity of the resulting algorithm, but even more so an effect of implementation challenges. Indeed, these models are large-scale and complex; it is already nontrivial to implement them in a "monolithic" fashion and solve them with general-purpose tools, even more so to devise decomposition approaches. While current state-of-the-art general-purpose solvers like `Cplex` and `SCIP` do offer some automatic decomposition capabilities that can be exploited, these are limited to one structure at the time. To the best of our knowledge, no current system meaningfully supports exploiting two structures simultaneously.

In the following we will first present a general algorithmic scheme for a decomposition-based primal heuristic suited for (1.1)–(1.5), borrowing and extending ideas from [1, 11, 19], and then discuss which components of `SMS++` make it possible to implement a generic yet efficient form of such a scheme.

1.1 A generic decomposition-based primal heuristic

The first ingredient of the decomposition-based primal heuristic is the ability of efficiently solving "tight" relaxations of (1.1)–(1.5), which provide both lower bounds and useful primal solutions. This is obtained by relaxing (1.4) in a Lagrangian fashion with multipliers λ_t, yielding the Lagrangian Dual

$$\max \Big\{ \textstyle\sum_{t\in T} \lambda_t v_t + \sum_{i\in I} \min \big\{ \sum_{t\in T} c_{it}(u_{it}, x_{it}) + \lambda_t V(u_{it}, x_{it}) \;:\; (1.2) \;,\; (1.3) \;,\; (1.5) \big\} \Big\} \tag{1.6}$$

The advantage of (1.6) over the ordinary continuous relaxation of (1.1)–(1.5)—that is, relaxing (1.5)—lies in the fact that it provides a (possibly, much) stronger bound: in fact, it is well-known that, if the objective $c_{it}(\cdot)$ is linear, the bound is equivalent to that of

$$\begin{aligned} \min\ & \textstyle\sum_{i\in I} \sum_{t\in T} c_{it}(u_{it}, x_{it}) && && (1.1)\\ & (u_{it}, x_{it}) \in conv\big((1.2) \;,\; (1.3) \;,\; (1.5) \big) && i \in I && (1.7)\\ & \textstyle\sum_{i\in I} V(u_{it}, x_{it}) = v_t && t \in T && (1.4) \end{aligned}$$

i.e., the "convexified relaxation" corresponding to taking the convex hull of the subproblems and intersecting them with the relaxed constraints [7]. For the nonlinear case, a similar result holds where the objective is the close-convexification of $c_{it}(\cdot)$ over the feasible region [15]. Of course, whether the potential bound advantage actually translates into a computational advantage for the solution of the original (1.1)–(1.5) depends both on how significant the advantage is (which is problem-dependent), and on the computational effort required to solve (1.6) as opposed to the ordinary continuous relaxation. Two ingredients are crucial for solving (1.6) efficiently:

1. the capability of efficiently solving the independent subproblems for each $i \in I$, which are in principle combinatorial ones (possibly with some exceptions, as discussed later on) and therefore may require specialized algorithms (cf. e.g. [10] for the Unit Commitment);
2. the capability of efficiently finding the optimal dual solution λ^* only provided with an "oracle" solving the subproblem(s) at designated iterates λ^ℓ; for this task bundle-type methods [6, 8] are the approach of choice for their numerous advantages over subgradient-type methods [4, 9], among which:

(a) the capability of exploiting the structure of specific subproblems that are "easy" [14] (just small Linear Programs), which happens n plan4res models and has been shown to be very useful in the UC context [19];

(b) the capability of only approximately solve the subproblems [21, 22], which is crucial in particular for nested approaches whereby the subproblems themselves are solved by decomposition.

The `SMS++` structure, and the components developed so far, are ideal for providing both ingredients, as discussed below. However, for the purpose of decomposition-based primal heuristics, a further advantage of bundle-type methods have to be commented upon: the capability of producing *two* different types of primal solutions along the iterations. In particular, for each fixed λ^ℓ, one typically has available for each $i \in I$:

1. a solution $(\,\bar{u}_i^\ell\,,\,\bar{x}_i^\ell\,) = (\,[\,\bar{u}_{it}^\ell]_{t\in T}\,,\,[\,\bar{x}_{it}^\ell\,]_{t\in T}\,)$ satisfying (1.2), (1.3) and (1.5) (hence, in particular $\bar{u}_{it}^\ell \in \mathbb{Z}^{n_{it}}$) but *not* (1.4);
2. a "convexified" solution $(\,\tilde{u}_i^\ell\,,\,\tilde{x}_i^\ell\,) = (\,[\,\tilde{u}_{it}^\ell\,]_{t\in T}\,,\,[\tilde{x}_{it}^\ell]_{t\in T}\,)$, computed as a convex combination of the previously obtained $(\,\bar{u}^{\bar{\ell}}\,,\,\bar{x}^{\bar{\ell}}\,)$ for $\bar{\ell} < \ell$; this is typically *not* integer (i.e., (1.5) is not satisfied), but it often "almost" satisfies (1.4)—in the sense that it does at least at termination, at least within the set tolerances, and that the violation of the constraint is much less that what the typical $(\,\bar{u}^\ell\,,\,\bar{x}^\ell\,) = [\,(\,\bar{u}_i^\ell\,,\,\bar{x}_i^\ell\,)\,]_{i\in I}$ entails.

This behaviour is well-known, and in general it does not depend on the details of the algorithm used to solve (1.6); indeed, it depends on the theoretical coincidence between (1.6) and (1.1),(1.7),(1.4), with $(\,\tilde{u}^\ell\,,\,\tilde{x}^\ell\,) = [\,(\tilde{u}_i^\ell, \tilde{x}_i^\ell)\,]_{i\in I}$ typically converging to the optimal solution of the latter as $\ell \to \infty$ [7]. However, the actual numerical properties of $(\tilde{u}^\ell, \tilde{x}^\ell)$ (say, how quickly it becomes feasible) do vary significantly with the choice of the algorithm. In particular, bundle-type methods occupy a convenient middle-ground between subgradient methods—where $(\tilde{u}^\ell, \tilde{x}^\ell)$ becomes feasible only exceedingly slowly—and pure cutting-plane ones—where $(\tilde{u}^\ell, \tilde{x}^\ell)$ is always feasible, but at the cost of much-lesser-quality $(\bar{u}^\ell, \bar{x}^\ell)$ for most iterations. Hence, they have repeatedly proven to be a convenient starting point for the implementation of *Lagrangian heuristics* which, using *both* $(\tilde{u}^\ell, \tilde{x}^\ell)$ and $(\bar{u}^\ell, \bar{x}^\ell)$, strive to produce solutions that satisfy *both* (1.5) and (1.4). These have shown good success for "simple" deterministic UC [1, 11, 12, 13], and can be extended to more complex stochastic versions of the problem [19]. Interestingly, these heuristics share many traits with the *feasibility pump* methods, that have been shown to be wildly successful for both linear and nonlinear (possibly nonconvex) mixed-integer optimization (cf. [3] and the references therein). It should also be remarked that the computation of $(\tilde{u}^\ell, \tilde{x}^\ell)$ entails little extra cost w.r.t. that required anyway for the solution of (1.6); there could be a more significant memory cost due to the need of storing a large number of $(\bar{u}^\ell, \bar{x}^\ell)$ solutions, but this can be countered in a number of ways, including aggregation [7, 8].

All this suggests that a general algorithmic scheme along these lines could be quite useful for the solution of the many practical problems that share the structure (1.1)–(1.5). A possible form of the general scheme is now described.

The strategy is based on the idea of exploiting the time structure of the problem: in other words, instead of solving one multi-period problem at once, many smaller subproblems, each for one or more periods of time (one subproblem represents one or more periods) are solved in sequence. The subproblems are solved from the initial time period to the end, using the solution of the previous subproblem as input to the current one, in a sliding window fashion. While the idea is reasonably simple, some qualifying aspects of the implementation are:

- Each subproblem, corresponding to one consecutive interval of time instants, is actually characterized by *three* "windows":
  1. the "main window" (in blue in Figure 1.1) is the part that has both the x and the u variables free, and where the u are actually constrained to be integer-valued; it is the part of the time horizon for which the subproblem actually takes the *integer* decisions (u);
  2. the "backward window" (dashed red in Figure 1.1, if any)) is a set of time instants preceding the main window, where the u variables are *fixed* but the x ones are still free; this allows some recourse for decisions taken at an earlier stage (if any) that may have proven themselves inappropriate in the light of new information corresponding to the subsequent time periods, but recourse is limited to the "easy" continuous decisions, leaving the "hard" combinatorial part of the decisions untouched;

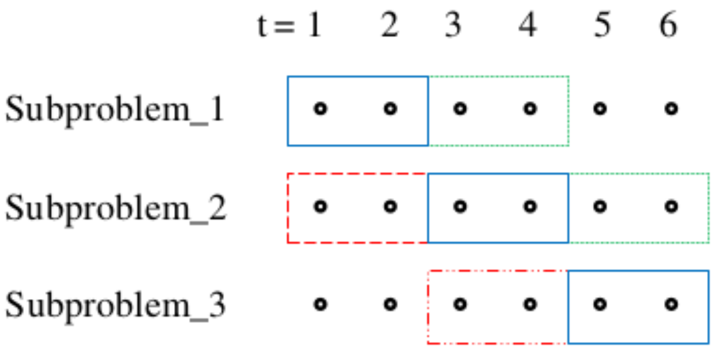

Figure 1.1 Main, backward and forward windows

3. the “forward window” (in green in Figure 1.1, if any)) is a set of time instants following the main window, where both the x and the u variables are free, but the u variables are *not* constrained to be integer-valued; this provides the main window with some forward insight on the future consequences of the decisions, but with in a relaxed form that does not increase too much the cost of the subproblems.

Clearly, the idea is to solve the subproblems in increasing ordering of time, so that each subproblem takes the “hard” decisions about the interval of time instants it governs (those of the main window), with some possible recourse on some of the previous ones but taking into account, at least partly, the impact on decisions of subsequent subproblems.

- In addition, the objective function of each subproblem is modified with the introduction of a (nontrivial) proximal term, that could be quadratic or linear (depending on the chosen norm). The proximal term tries to reduce the distance to the available solutions (the convexified one $(\tilde{u}^\ell, \tilde{x}^\ell)$ and the integer unfeasible one $(\bar{u}^\ell, \bar{x}^\ell)$), which hopefully introduces some “global” information to guide the “local” choice of the variables for the current period. Indeed, the main issue of this kind of approaches has been shown to be that individual periods tend to take “selfish” decisions that may reveal themselves highly counter-productive “in the long run”; while the introduction of the backward and forward windows tries to account for this, unless the windows cover all (or a large part) of the remaining horizon the “lack of perspective” may still be significant. With the proximal term, the objective has the general form

$$\min \sum_{t\in S}\sum_{i\in I} c_{it}(u_{it}, x_{it}) + \alpha \left[\beta\left(\gamma\|x_{it} - \bar{x}_{it}\| + (1-\gamma)\|u_{it} - \bar{u}_{it}\|\right) + (1-\beta)\left(\gamma\|x_{it} - \tilde{x}_{it}\| + (1-\gamma)\|u_{it} - \tilde{u}_{it}\|\right)\right] \quad ,$$

where $S \subset T$ is the set of time instants covered by the specific subproblem. In plain words, the parameter α governs the strength of the proximal term w.r.t. that of the original objective, the parameter $\beta \in [0, 1]$ governs the relative strength of proximity with the (integer unfeasible) solution $(\bar{x}, \bar{u})$ w.r.t. proximity with the (continuous almost feasible) solution $(\tilde{x}, \tilde{u})$, while the parameter $\gamma \in [0, 1]$ governs the relative strength of proximity to the continuous part (x) w.r.t. proximity to the integer one (u). Some weighting may be added to account for the different “units of measure” of the different parts (for instance, variables x may have vastly different numerical values from variables u), and some further minor generalization of the approach may be possible; however, the above already depicts a quite general algorithmic scheme whose numerous parameter can be tailored to adapt to a vast range of trade-offs.

Indeed, the actual behaviour of the approach will strongly depends on the setting of its many parameters, among which:

1. the number of subproblems in which the horizon T is divided, and therefore the size of the main window;
2. the size (possibly 0) of the backward and forward windows;
3. the choice of the norms in the proximal term, which may impact on the type of problem (for instance, if $c_{it}(\cdot)$ is already nonlinear then a 2-norm is likely appropriate but if the objective function is linear then probably the 1-norm or ∞-norm are more appropriate in order not to lose the property);
4. the choice of the parameters in the proximal term, which again may impact on the type of problem (for instance, $\gamma = 0$ means that the norm only have to be taken of binary vectors, and this allows using specific modelling tricks that may favourably impact its structure [2, 22]).

Thus, a flexible framework should allow to find the right trade-off between subproblem cost and quality of the obtained solution for each specific application. The typical trade-off working well in applications is that where the heuristic is inexpensive, so that it can be ran frequently during the solution process of (1.6), thereby benefiting from the *diversification* naturally provided by the fact that $(\bar{x}, \bar{u})$ tend to be quite different at each iteration (while $(\tilde{x}, \tilde{u})$ tend to converge towards one point, which on the other hand provides a useful source of *intensification*).

However, a further degree of flexibility is provided by the possibility of augmenting the scheme with a backward feedback loop. This has shown to be useful in some application [19] where, despite all the parameter tuning, it was still difficult to obtain feasible solutions out of the process since the "greedy" nature of the approach made it very likely that the scarce resources to be carefully distributed across the horizon (in the application, water of hydro reservoirs) were consumed to early, leaving not enough resource at the final stages that therefore ended up too often infeasible. To remedy this, the general idea is to add to the subproblem a function that better represents the cost-to-go for the decision made therein, to complement the information provided by the proximal terms (when this shows not to be enough). This naturally leads to a scheme very akin to nested Benders' decomposition, which—by the way—is already the cornerstone of the management of the Seasonal Storage Valuation in plan4res.

The issue is of course that Benders' approach requires continuous subproblems (for one needs to apply duality), whereas in general the subproblems will be mixed-integer ones. Hence, in general one can't expect Benders' approach to provide an optimal solution in our setting; yet, this is not likely to be a major issue since the approach is heuristic from the start. Therefore, potentially useful cuts can be obtained in different ways, among which:

1. by standard duality arguments when solving the continuous relaxation of the subproblem, say if it is already empty (feasibility cut);
2. by exploiting in the subproblem the same "horizontal" structure upon which (1.6) is based, which induces a convexification and therefore allows to produce Benders' cuts out of a stronger formulation than the simple continuous relaxation;
3. if the subproblem turns out to be empty at integrality (but not when the continuous relaxation is solved), by employing *no-good cuts* [2] (also known as *combinatorial Benders' cuts*).

A number of variants of such an approach can be devised, differing in other crucial details like when cuts are generated (only when the last subproblem is solved, or possibly at every subproblem) and how backtracking to previous subproblems is managed (cuts are added backward all the way up to the first subproblem, in a full-forward/full-backward fashion, or backtracking is interrupted earlier in order to avoid re-solving many subproblems). Again, while the focus of Lagrangian heuristics has traditionally been on fast approaches repeated many times (and therefore not too complex), the generality of the scheme allows to explore a vast landscape of trade-offs, hopefully providing means for finding variants well-suited for each different application, among which these specific of plan4res.

However, the generality of the approach also implies that efficient implementation is also absolutely non-trivial. In the next section we discuss which components of SMS++ provided in this Deliverable allow for this to happen.

1.2 SMS++ components involved in the decomposition-based heuristic

The following SMS++ components in particular, described in details in the following Chapters of this Deliverable, are crucial for the efficient, effective and flexible implementation of the general decomposition-based heuristic framework described above:

- The base Block class, the cornerstone of the whole SMS++ system, is perfectly suited for representing the concept of a problem with the structure of (1.1)–(1.5) and operate on it independently on the details of the sub-Block representing the individual "units". In particular, the optimal arrangement for the decomposition-based heuristic is that of a Block having the natural structure required for (1.6), i.e., in which each sub-Block represent one "unit", but where the sub-Block themselves have the explicit temporal structure, i.e., they are further divided in sub-Block corresponding one to each time instant.

- In particular, the `LagBFunction` component allows to generally represent the Lagrangian function of a generic `Block` (which is the sub-`Block` of `LagBFunction`, itself both a `C05Function` and a `Block`), completely irrespectively of the `Block` itself and only provided that its `Objective` is a `FRealObjective` with a `LinearFunction` inside. This means that any specialized, structure-exploiting `Solver` can be attached to the `Block` for efficiently computing the Lagrangian function—a fundamental step in the solution of the Lagrangian Dual—without the `LagBFunction` needing to know anything of the details of the `Block` and of the solution process, thanks to the completely general `Solver` interface.

- For the case where the required structure is not naturally there—say, only the "horizontal" one is apparent, but not the "vertical one"—it is possible to exploit unique capability of `SMS++`, i.e., is its support for explicit *reformulation* of a `Block` under the form of an `R3 Block`. This functionality is in particular supported by the (virtual) methods

  ```
  Block::get_R3_Block()
  Block::map_back_solution()
  Block::map_forward_solution()
  Block::map_back_Modification()
  Block::map_forward_Modification()
  ```

  as described in this document. This allows to construct an `R3 Block` that represent the "vertical structure" of the original `Block`, and even attach specialized `Solver` to its sub-`Block` if available. However, it has to be remarked that specific support is required by the `Block` for this, under the form of the capability of producing an `R3 Block` with the required structure. Yet, since the construction of `R3 Block` is controlled by a general `Configuration` object, the decomposition-based heuristic can still be kept completely general and independent on the details of the original `Block` and its `R3 Block`, only provided that the right `Configuration` is provided at runtime.

- Since in several cases the subproblems corresponding to either the "horizontal" or "vertical" structure will not have a structure specific enough to admit specialized `Solver`, general-purpose `Solver` are provided by `SMS++` that can tackle large classes of optimization problems. In particular, `MILPSolver` provides a general interface for solvers of problems with all linear constraints and objective, be them continuous or mixed-integer; the specific derived class `CplexMILPSolver` implements the interface via calls to the state-of-the-art commercial solver Cplex, and a `SCIPMILPSolver` is under development by ZIB to allow using the open-source SCIP solver [17] instead. Also, `MCFBlock` provides a full-featured interface for Min-Cost Flow problems (notoriously, at the interface between continuous and combinatorial optimization), and the corresponding `MCFSolver` provides access to several efficient open-source implementations of solution algorithms for Min-Cost Flow problems.

- Another crucial algorithmic requirement is the efficient solution of NonDifferentiable Optimization problems such as typically is (1.6). For this `BundleSolver`, the SMS++-native implementation of the algorithms available in the `NDOSolver / FiOracle` project [18]. As foreseen by the project, `BundleSolver` exploits some existing software modules from the `NDOSolver / FiOracle` project, in particular the `MPSolver` generic interface for solvers of the Master Problem in bundle-type algorithms and its two implementations `QPPenaltyMP` (using a specialized QP solver [5]) and `OSIMPSolver`, rather relying on a `OSISolverInterface`. `BundleSolver` provides state-of-the-art capabilities that are crucial for the efficient solution of (1.6), among which sophisticated management of the inexact computation of the function [21] and support for "easy" components [14], that have been shown to be invaluable in energy applications [19]. Remarkably, the "easy components" feature of `BundleSolver` relies on `LagBFunction` for specifying the interface and on `MILPSolver` for the construction of the coefficient matrices, underlying the modularity of the project.

- Crucially, `BundleSolver` is not specific for Lagrangian functions, but can solve `Block` sporting any (sum of) c(s, and a single `LinearFunction`). While `LagBFunction` is a `C05Function`, another highly relevant specimen of the class is `BendersBFunction`, which provides the same functionalities of `LagBFunction` for the other main form of decomposition (dual to the Lagrangian one), that of Benders. Again, this allows to compute the value function of any general `Block` subject to the right-hand-side of an arbitrary set of its `Constraint` to be a linear function of an arbitrary set of `Variables` without any assumption on the nature of the `Block`, and allowing the use of specialized `Solver` where available (and of general-purpose ones where not). The very same advanced capabilities of `BundleSolver` that are crucial for the efficient solution of (1.6) can then be exploited for tackling nested Benders' decomposition approaches like the ones envisioned in the advanced implementations of the decomposition-based heuristic.

- Another general structure, somewhat minor but still relevant, that is necessary in particular for the implementation of nested Benders' decomposition approaches is that of `PolyhedralFunction`, a simple convex (or concave) function defined by the pointwise maximum (or minimum) of a "small" set of linear forms. Such a structure has two different representations that are useful to different `Solver`; one as a NDO function (hence, a `C05Function`), and the other as an explicit linear program. Automatic handling of these two different representation, so that—for instance—the same `PolyhedralFunction` can be optimized upon natively by either a `BundleSolver` or a `MILPSolver`, is provided by `PolyhedralFunctionBlock`. Besides being crucial in the implementation of the Stochastic Dual Dynamic Programming approach foreseen for the Seasonal Storage Valuation in plan4res, such component is also useful for the Benders' decomposition part of advanced implementations of the decomposition-based heuristic

- Finally, since the main application of the decomposition-based heuristic within plan4res is the European Unit Commitment, "basic" `Block` representing the problem must be available. These are provided in this Deliverable. In particular, `UCBlock` represents the main EUC problem, with abstract classes `UnitBlock` and `NetworkBlock` defining the interface between the two main structures in the problem: units ("horizontal structure", coupled over time but geographically decomposable) and transmission network ("vertical structure", geographically coupled but decomposable over time). In turn, several classes derived from `UnitBlock` are defined to cater the different units required by plan4res, among which chiefly `ThermalUnitBlock` with a specialized `DPSolver` [10] and `HydroUnitBlock`, but also `BatteryUnitBlock`, `HeatBlock` and `IntermittentUnitBlock`. Also, `BusNetworkBlock` and `DCNetworkBlock` are defined for representing the different types of transmission network. Finally, `UCBlock` also contains a single `PolyhedralFunctionBlock` to represent the future-value-of-water function for hydro reservoirs, which is the crucial interface between the EUC (short-term) and the SSV (mid-term) levels of plan4res.

To summarise, the components provided by this Deliverable allow the efficient implementation of general and flexible decomposition-based heuristic approaches.

Bibliography

[1] A. Borghetti, A. Frangioni, F. Lacalandra, and C.A. Nucci. Lagrangian heuristics based on disaggregated bundle methods for hydrothermal Unit Commitment. *IEEE Transactions on Power Systems*, 18(1):313–323, February 2003.

[2] C. d'Ambrosio, A. Frangioni, L. Liberti, and A. Lodi. On interval-subgradient cuts and no-good cuts. *Operations Research Letters*, 38:341–345, 2010.

[3] C. d'Ambrosio, A. Frangioni, L. Liberti, and A. Lodi. A storm of feasibility pumps for nonconvex minlp. *Mathematical Programming*, 36(2):375–402, 2012.

[4] G. d'Antonio and A. Frangioni. Convergence analysis of deflected conditional approximate subgradient methods. *SIAM Journal on Optimization*, 20(1):357–386, 2009.

[5] A. Frangioni. Solving semidefinite quadratic problems within nonsmooth optimization algorithms. *Computers & Operations Research*, 21:1099–1118, 1996.

[6] A. Frangioni. Generalized bundle methods. *SIAM Journal on Optimization*, 13(1):117–156, 2002.

[7] A. Frangioni. About Lagrangian methods in integer optimization. *Annals of Operations Research*, 139(1):163–193, 2005.

[8] A. Frangioni. *Standard Bundle Methods: Untrusted Models and Duality*, chapter 2. Springer, 2019.

[9] A. Frangioni, B. Gendron, and E. Gorgone. On the computational efficiency of subgradient methods: a case study with Lagrangian bounds. *Mathematical Programming Computation*, 9(4):573–604, 2017.

[10] A. Frangioni and C. Gentile. Solving nonlinear single-Unit Commitment problems with ramping constraints. *Operations Research*, 54(4):767–775, 2006.

[11] A. Frangioni, C. Gentile, and F. Lacalandra. New Lagrangian heuristics for ramp-constrained Unit Commitment problems. In *Proceedings 19th Mini-EURO Conference in Operational Research Models and Methods in the Energy Sector –ORMMES 2006*. INESC Coimbra, 2006.

[12] A. Frangioni, C. Gentile, and F. Lacalandra. Solving Unit Commitment problems with general ramp contraints. *International Journal of Electrical Power and Energy Systems*, 30:316–326, 2008.

[13] A. Frangioni, C. Gentile, and F. Lacalandra. Sequential Lagrangian-MILP approaches for Unit Commitment problems. *International Journal of Electrical Power and Energy Systems*, 33:585–593, 2011.

[14] A. Frangioni and E. Gorgone. Generalized bundle methods for sum-functions with "easy" components: Applications to multicommodity network design. *Mathematical Programming*, 145(1):133–161, 2014.

[15] C. Lemaréchal and A. Renaud. A geometric study of duality gaps, with applications. *Mathematical Programming*, 90:399–427, 2001.

[16] plan4res partners. Deliverable D3.2: Functional and mathematical description, 2018.

[17] plan4res partners. Deliverable D5.1: New version of the SCIP solver adapted and integrated in plan4res platform, 2019.

[18] plan4res partners. Deliverable D5.3: New version of the NDOSolver/FiOracle software, 2019.

[19] M.R. Scuzziato, E.C. Finardi, and A. Frangioni. Comparing spatial and scenario decomposition for stochastic hydrothermal Unit Commitment problems. *IEEE Transactions on Sustainable Energy*, 9(3):1307–1317, 2018.

[20] W. van Ackooij, I. Danti Lopez, A. Frangioni, F. Lacalandra, and M. Tahanan. Large-scale Unit Commitment under uncertainty: an updated literature survey. *Annals of Operations Research*, 271(1):11–85, 2018.

[21] W. van Ackooij and A. Frangioni. Incremental bundle methods using upper models. *SIAM Journal on Optimization*, 28(1):379–410, 2018.

[22] W. van Ackooij, A. Frangioni, and W. de Oliveira. Inexact stabilized benders' decomposition approaches, with application to chance-constrained problems with finite support. *Computational Optimization and Applications*, 65(3):637–669, 2016.