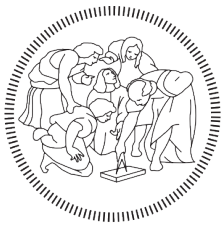


POLITECNICO
MILANO 1863

School of Industrial and Information Engineering

Master of Science in Energy Engineering

A novel MILP decomposition algorithm ensuring global optimality for the predictive design of complex Multi-Energy Systems

Advisor: Prof. Marco ASTOLFI

Co-Advisor: Prof. Giuseppe CALAFIORE

Candidates:

Paolo BARBATO ID. 892333
Carlo Alberto GAETANIELLO ID. 893477

Academic Year 2018 - 2019

Acknowledgements

At first, we would like to thank the institution that allowed the realization of this Thesis work: Politecnico di Milano. We are grateful to prof. Marco Astolfi and to Dr. Luca Moretti for giving us the opportunity to discover the fascinating world of energy systems optimization. Having the possibility to rely on their continuous support, perfectly balanced between guidance and freedom of action, has been determining in reaching our results. A special acknoledgement goes to prof. Giuseppe Calafiore for his precious and expert advisory and for the sincere interest shown in our work. Moreover, we would also like to thank Alta Scuola Politecnica for providing us with both personal and professional skills that strongly helped us in carrying on this Thesis work with the right attitude. Thanks to all the SEI Pioneers for being our extended family during the last year, both inside and outside university. Finally, a special thanks goes to all the Wiseair team, which supported us in the hardest moments by always being there to lend a hand.

Milan, December 2019 Paolo and Carlo Alberto

Voglio ringraziare innanzitutto i miei genitori e mio fratello Simone per avermi continuamente supportato e sopportato dal primo giorno di lezione fino all'ultimo esame di Magistrale. Ho percepito il vostro caldo sostegno in ogni singolo momento, non sentendomi mai solo di fronte agli ostacoli incontrati.

Grazie Alessandra per aver condiviso con me gioie, paure, successi e frustrazioni di questi intensi anni universitari. Il nostro percorso insieme mi ha reso una persona migliore di quella che avrei mai immaginato di essere oggi.

Grazie Antonio, Laura P., Andrea, Federico, Nicolò, Matilde e Stefano per aver reso Milano una bellissima città in cui vivere in questi cinque anni.

Infine grazie Potena, Laura T., Tommaso, Nicola, Raimondo, Michelangelo, Giovanna e tutti gli amici "di giù" per essere rimasti nonostante i tanti chilometri che ci separano e i tanti anni che sono passati dai bei tempi in cui stare insieme era ancora una bella abitudine.

Milano, Dicembre 2019 Paolo

Prima di tutto vorrei ringraziare i miei genitori, per avermi dato la possibilità di affrontare il mio percorso accademico a Milano, per avermi supportato e aver sempre creduto nelle mie capacità. Questo successo parte da loro. Un ringraziamento va anche a mio fratello Francesco e mia sorella Maria Elena, per avermi sempre fatto sentire a casa durante i miei rientri dalle lunghe settimane milanesi.

Vorrei ringraziare anche i *coinqulini di Viale Nazario Sauro* Riccardo, Francesco e Filippo. Insieme abbiamo condiviso i migliori momenti dei nostri anni universitari, trasformando un orribile appartamento in un ricordo prezioso. Un ringraziamento speciale va agli amici e colleghi Andrea, Domenico e Alex, che mi hanno sempre accompagnato in questi cinque lunghi anni di lezioni ed esami. Studiare al Politecnico è stata un'esperienza indimenticabile e in gran parte è merito loro.

Grazie a Sara. Lei mi ha indirizzato cinque anni fa e più di chiunque altro è sempre stata al mio fianco per consigliarmi e sostenermi. Ha reso speciali i successi e meno amari i fallimenti. E più di tutto, ha avuto la forza per tirare fuori sempre il meglio di me.

Milano, Dicembre 2019 Carlo Alberto

Contents

List of Figures

List of Tables

Abstract

The scope of this Thesis is to present a novel MILP (Mixed-Integer Linear Programming) decomposition algorithm for the predictive design of complex Multi-Energy System (MES). MES are becoming extremely relevant for the energy sector, opening to high renewable penetration without giving up good conversion efficiencies, system flexibility and optimal market interaction. Nevertheless, due to their inherent complexity, finding effective optimization methods that allow to properly design and consequentially operate MES is not a trivial task.

In the first part of this work, different optimization methods are reviewed in view of their main advantages and drawbacks and, among these, MILP optimization methods are identified as one of the most promising approaches, mainly because of their ability to ensure global optimality. MILP optimization of energy systems is introduced both from a mathematical and from a numerical perspective, and a novel MILP modeling environment for MATLAB named POLIMIP is presented, adding relevant functionalities with respect to state-of-the-art softwares such as YALMIP [73] and allowing for the implementation of advanced MILP optimization techniques that have been used for developing the proposed algorithm.

The second part of the Thesis is focused on tackling one of the main challenges of MILP optimization of MES design: reducing computational complexity. This is initially done by comparing two of the most advanced decomposition methods found in literature and by highlighting their main potentialities and limitations. A novel decomposition paradigm based on the innovative concept of *local auxiliary problem* is then presented allowing for a higher modeling flexibility with respect

to previous methods without compromising computational performances. Finally, the new decomposition algorithm is implemented in POLIMIP and tested on a real case-study involving the optimal design of a multi-energy microgrid satisfying the electric, thermal and cooling loads of a university campus in Northern Italy. Three different time horizons are considered in the case-study: 3, 7 and 14 typical days. In all cases, the computational times of the proposed method outperform a general-purpose state-of-the-art MILP solver in reaching the global optimum by one order of magnitude, hence proving the effectiveness of the novel decomposition approach.

Keywords: Optimization, MILP, Decomposition, Design, Multi-Energy Systems, Microgrids

A novel MILP decomposition algorithm ensuring global optimality for the predictive design of complex Multi-Energy Systems

Paolo Barbato[1], Carlo Alberto Gaetaniello[1]

Abstract
The scope of this Thesis is to present a novel MILP (Mixed-Integer Linear Programming) decomposition algorithm for the predictive design of complex Multi-Energy System (MES). MES are becoming extremely relevant for the energy sector, opening to high renewable penetration without giving up good conversion efficiencies, system flexibility and optimal market interaction. Nevertheless, due to their inherent complexity, finding effective optimization methods that allow to properly design and consequentially operate MES is not a trivial task. In the first part of this work, different optimization methods are reviewed in view of their main advantages and drawbacks and, among these, MILP optimization methods are identified as one of the most promising approaches, mainly because of their ability to ensure global optimality. MILP optimization of energy systems is introduced both from a mathematical and from a numerical perspective, and a novel MILP modeling environment for MATLAB named POLIMIP is presented, adding relevant functionalities with respect to state-of-the-art softwares such as YALMIP [1] and allowing for the implementation of advanced MILP optimization techniques that have been used for developing the proposed algorithm. The second part of the Thesis is focused on tackling one of the main challenges of MILP optimization of MES design: reducing computational complexity. This is initially done by comparing two of the most advanced decomposition methods found in literature and by highlighting their main potentialities and limitations. A novel decomposition paradigm based on the innovative concept of *local auxiliary problem* is then presented allowing for a higher modeling flexibility with respect to previous methods without compromising computational performances. Finally, the new decomposition algorithm is implemented in POLIMIP and tested on a real case-study involving the optimal design of a multi-energy microgrid satisfying the electric, thermal and cooling loads of a university campus in Northern Italy. Three different time horizons are considered in the case-study: 3, 7 and 14 typical days. In all cases, the computational times of the proposed method outperform a general-purpose state-of-the-art MILP solver in reaching the global optimum by one order of magnitude, hence proving the effectiveness of the novel decomposition approach.

Keywords
Optimization - MILP - Decomposition - Design - Multi-Energy Systems - Microgrids

[1] *Department of Energy, Politecnico di Milano, MIlan, Italy*

Contents

Introduction

In a world calling for important and fast changes in our energy infrastructures [2], the ability to effectively design systems whereby electricity, heat, cooling, fuels, and transport optimally interact with each other becomes decisive to fully exploit the potential of new paradigms and technologies nowadays emerging in the energy sector. In this context, Multi-Energy Systems (MES) play an important role [3]. In particular, being able to effectively

design their architecture in consideration of the combined effect of capital and operation costs is becoming more and more important both from a technical and from an economical perspective. Such task is a challenging one due to the inherent complexity of most mathematical models of Multi-Energy Systems, which typically require advanced numerical techniques that in many cases are not able to ensure the global optimality of the solution, hence strongly limiting the quality of the proposed design. The main reasons behind the mathematical complexity of MES models are: discreteness of the equipment, binary logics of the unit commitment, strong inter-dependency between design and operation, non-linearities, necessity to consider a sufficiently long time horizon in the simulation of the system's operation. Being an enumerative optimization method, Mixed-Integer Linear Programming (MILP) represents one of few effective approaches ensuring the global optimality of the solution. Moreover, by means of integer variables MILP models can accurately describe the behaviour of the energy system. The non-negligible drawback of MILP is its numerical complexity. In fact, integer programming is NP-complete, which essentially means that its computational time tends to explode with increasing number of discrete variables [4]. So called *hierarchical decomposition methods* are an effective way to significantly improve the computational performances of MILP optimization algorithms for MES [5] [6]. These approaches exploit some characteristic features of the problem structure to improve the search of the combinatorial solution space and allow for the optimization of more complex system architectures simulated on longer time horizons, thus increasing the quality of the design solution. The core contribuition of this Thesis work is a novel decomposition algorithm for MES design problem allowing for a higher modeling flexibility with respect to previous methods. The new algorithm is introduced in chapter 3 after a brief review of the main approaches to MES design (chapter 1) followed by an in-depth analysis of MES MILP optimization (chapter 2). Finally, in chapter 4, the novel decomposition algorithm is tested on a real case-study involving an on-grid MES with electric, thermal and cooling loads. The results of the test are then compared with that of a state-of-the-art commercial MILP solver.

1. Multi-Energy Systems (MES): an overview

1.1 MES design: a complex task

A comprehensive definition of a Multi-Energy System (MES) can be found in [3]:

> *"..an integrated energy system consisting of distributed energy resources and multiple energy loads operating as a single, autonomous grid either in parallel to or "islanded" from the existing utility grid"*

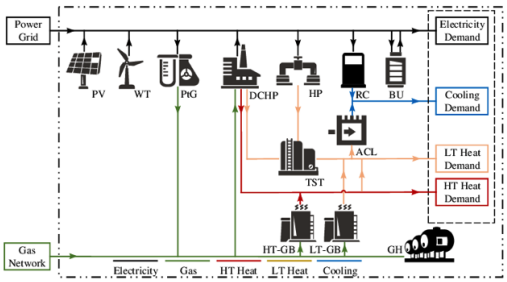


Figure 1. Typical architecture of a multi-energy microgrid

Although initially designed for the production of electricity, today MES (see fig.1) are becoming the most promising model of distributed generation of electric, thermal and cooling power, for both on-grid and off-grid applications [7]. The main advantages ranges from an efficient exploitation of the primary energy sources (e.g. through heat recovery), to a significant flexibility and a potentially high penetration of renewable energy sources. Of course, all these potentialities are effectively exploited only if both the design and the operation of the system are conducted in an optimal way. For this reason, an increasing attention is rising towards MES modeling and optimization both in the academic and in the industrial fields.

The main challenges encountered when approaching the optimal design of a MES originate from the inherent coupling between design and operation. In fact, due to the high level of interconnection between generating units, design choices regarding a single variable may dramatically change the optimal value of all the others. Moreover, many internal interconnections also result in many possible operation strategies that inevitably affect the optimality of the design solution. The result is a single, huge, and often untreatable mathematical problem.

1.2 Main approaches to MES design optimization

Modern approaches to MES design optimization may be clustered in two main categories: *two-layer algorithms* vs. *one-shot algorithms* [8]. In the firsts, the problem is solved by means of two nested but separate algorithms. The outer loop iteratively generates a potential design solution, while the inner loop optimizes the unit committment and estimates the operating expenses for the corresponding system configuration. Conversely, one-shot algorithms make use of a single model containing both design and operation variables. System sizing and dispatch are solved simultaneously in a single optimization

problem. It is possible to mathematically demonstrate that design and operation in MES are coupled such that when solve sequentially or iteratively the results are not guaranteed to constitute a combined system optimum [9]. For this reason, contrarily to one-shot algorithms, traditional two-layer algorithms cannot ensure the global optimality of the solution. At the same time, most one-shot models tend to become very complex when applied to large MES and may result in non-acceptable computational times.

Beyond Montecarlo simulation methods, which today are of low interest, two-layer algorithms for MES design optimization are largely found in literature, mainly due to their ability to find good sub-optimal solution in reasonable computational times and to the possibility to easily implement realistic operational strategies in the lower layer. The most commons are Particle Swarm Optimization (PSO) and Genetic Algorithms (GA). PSO is one of the most used evolutionary algorithms to explore the upper level of MES design problem [10]. Evolutionary algorithms consist in metaheuristic procedures that imitate biological mechanism to efficiently explore the combinatorial space of design solutions [11]. PSO tends to guarantee a good exploration of the solution space without going too far from the global optimum point [12] [13], while one of its main drawbacks is that computational times may strongly vary from one instance to another, depending on the input tuning parameters. With respect to PSO methods, Genetic Algorithms (which still belong to the class of evolutionary algorithms) are typically adopted due to their ability to quickly improve the objective function, achieving convergence in a very short time [14]. On the other hand, they easily get stuck on local optima that can significantly differ from the global one [12], hence determining a low quality of the solution.

One-shot algorithm are based on models where design and operation variables are simultaneously optimized in a single problem. The optimization solver sees the whole simulation period and finds the best operation strategy as if it was possible to precisely predict the load behaviours and the RES production for the entire time horizon of the optimization. This is generally called *predictive design*, and can be both a limitation (it is based on an operation logic that is typically not embedded in the on-field control systems) and an opportunity (it is the only way properly design systems with long-term memory effects, such as seasonal energy storage). Predictive (one-shot) MES design is mainly performed with LP (Linear Programming), MILP (Mixed-Integer Linear Programming) and MINLP (Mixed-Integer Non Linear Programming) techniques. LP is the simplest one, only allowing for a linear objective function, linear constraints, and for continuous variables. MILP is an extension of LP, also allowing for integer and binary variables. This feature is particularly important for MES optimization both for the design (it allows to model the discrete size of the selected equipment) but also for the operation (it allows to precisely model the unit commitment, introducing startup costs, minimum up-time, piece-wise performance curves, etc.). Finally, MINLP also allows for non-linear objective and constraints but, on the other hand, determines a significant increase in the computational burden with respect to linear methods such as LP and MILP and in some cases it does not ensure the global optimality [15].

2. MILP optimization and MES design

2.1 MILP optimization: general concepts and software implementation

A *Mixed-Integer Linear Programming* problem (or *MILP* problem) is a linear optimization problem containing both discrete and continuous variables. The possibility to deal with integer and binary variables makes MILP optimization a very versatile tool nowadays adopted solve a wide number of real world problems [16] [17] [18] [19] [20] [21]. The most compact way to express a generic MILP problem is its *canonical form*:

$$\begin{aligned} \min/\max \quad & f = c^T x + d^T y \\ \text{s.t.} \quad & Ax + By \leq b \\ & x \geq 0, x \in \mathbb{R}^n \\ & y \in \mathbb{Z}^m \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^n$ is a vector of n continuous variables, $y \in Z^m$ is a vector of m integer variables, $c \in \mathbb{R}^n$ is a vector containing all the n objective coefficients of the continuous variables x, $d \in \mathbb{R}^m$ is a vector containing all the q objective coefficients of the integer variables y, $b \in \mathbb{R}^r$ is a vector of r constant terms, $A \in \mathbb{R}^r \times \mathbb{R}^n$ and $B \in \mathbb{R}^r \times \mathbb{R}^m$ are the so called *constraint matrices* containing the $r \times n$ and $r \times m$ coefficients multiplying the continuous and integer variables appearing in each one of the r linear constraints defining the problem. It is worth noting that both the objective function $f = c^T x + d^T y$ and the set of r constraints $Ax + By \leq b$ are expressed as linear combinations of the continuous and integer variables x and y.

Despite their simplicity of representation, MILP problems are very difficult to solve due to the combinatorial nature of the domain of y variables. In fact, most solution algorithms for MILP problems are based on enumerative techniques whose computational complexity increases exponentially with the number of discrete variables [4]. As a consequence, many advanced algorithms have been developed in literature aiming at reducing the computational time required to solve an instance of a MILP problem. The most widely adopted of these methods is

by far the *Branch and Bound (B&B) algorithm*, introduced in 1960 by Land and Doig [22] and nowadays used as a reference algorithm by all commercial solvers [23]. The basic idea behind the B&B algorithm is to adopt a search strategy based on a binary enumeration tree in which each node represents a LP problem characterized by a unique set of linear constraints. The initial node (also known as *root node*) is solved by assuming continuous y variables (we speak about a *fully relaxed* problem). Moving through the nodes, the set of initial constraints is progressively expanded with new ones by suitably bounding the y variables (*branching*). These additional "branching constraints" are specifically aimed at restoring the integer nature of y. Thanks to this progressive constraining process, a lower bound of the objective function f is always available during the search by solving relaxed [1] LP problems. If an integer solution is found at a certain point of the search, we call this an *incumbent* and its objective value a *cutoff value* $\tilde{f}$. Since $\tilde{f}$ is surely an upper bound for the global optimum f, we are then able to *fathom* all the nodes exhibiting a solution of the LP relaxation higher than $\tilde{f}$. Typically convergence is declared setting a tolerance on the so-called relative gap[2]. In this way, the algorithm is able to significantly reduce the size of the explored portion of the combinatorial solution space with respect to an exhaustive enumeration method.

Many advanced B&B techniques have been developed over the years to increase the computational efficiency of the algorithm without compromising the global optimality of the solution. Among these, the most relevant ones are undoubtedly *cutting planes*, *heuristics* and *decomposition methods*. Cutting planes (or simply *cuts*) are inequality constraints that tighten the solution space of the relaxed linear problem without cutting out any possible discrete solution and that are not included in the initial set of constraints defining the MILP problem. If a cut also excludes discrete solutions, then it is a special type of cut called *integer cut*. These additional constraints can be derived in various ways (before or during the search) and are so effective in speeding the convergence of the algorithm that are nowadays embedded in all commercial MILP softwares as a modified of the B&B algorithm called *Branch and Cut*. Heuristics are techniques aiming at generating an integer feasible solution in a short time. These techniques don't guarantee any optimality by themselves but, if integrated in a B&B algorithm, they may significantly speed up the convergence time by promoting the generation of good cutoff values without conducting all the necessary branching. Finally, decomposition methods rely on partitioning the original MILP problem into smaller MILP or LP problems that are easier to solve. These approaches can be very effective in reducing the computational time of a MILP algorithm but, deriving from the exploitation of specific features of the problem structure, they tend to be strongly model-dependent.

In practice, MILP models of real-life energy systems are solved by relying on two typologies of softwares: *modeling environments* [24] [25] [26] [1] [27] [28] [29] [30] [31] [32] and *numerical solvers* [33] [34] [35] [36] [37] [38] [39] [40] [41] [42] [43] [44]. The firsts are used to translate the *algebraic formulation*[3] of a specific model into its equivalent canonical form (1), which is taken as input by the latters, who are responsible for tuning and implementing the branch-and-cut algorithm to eventually find its optimal solution. MILP modeling environments are available both as open and commercial softwares for many programming languages (Python, C, C++, .NET, Java, MATLAB etc.) and are substantially equivalent to each other. A notable exception concerns YALMIP [1], one of the most widely adopted MILP modeling software for the MATLAB programming language, for which we have highlighted three peculiar limitations (the latter of which is ascribed to the MATLAB API of the solver and not to YALMIP itself):

1. Impossibility to define sets, and thus to adopt an indexed syntax for the definition of variables and parameters;
2. Incompatibility with some of the functionalities of the the MATLAB API of the MILP solver;
3. Absence of advanced functionalities in the MATLAB API of the MILP solver.

2.2 POLIMIP: a set-oriented MILP modeling environment for MATLAB

All these limitations are overcome thanks to the development of a novel multi-language modeling environment with MATLAB interface named *POLIMIP: a set-oriented MILP modeling environment for MATLAB*. As evident in figure 2, POLIMIP is obtained starting from a modified version of YALMIP by implementing the function `ndSparse` [45] and other custom functions. Moreover, in order to allow the integration of other solver APIs to overcome the third limitation reported above, a multi-language architecture following the scheme reported in figure 3 was implemented. In this Thesis, the POLIMIP modelling environment was used to implement the novel decomposition method proposed in section 3 and to set up the case-study analyzed in section 4.

[1] A *relaxed* problem in an unconstrained version of the original problem in which at least one integer variable assumes a fractional value

[2] The *relative gap* is the distance between the current cutoff value and the the highest lower bound available, divided by the cutoff

[3] The *algebiric formulation* is the physical description of the problem by means of physical parameters, variables and constraints

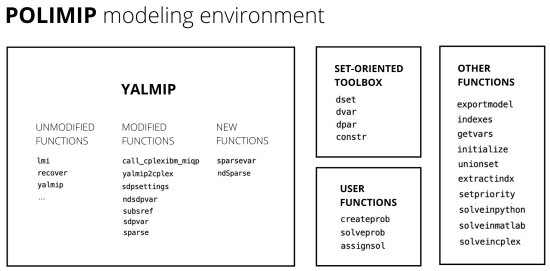


Figure 2. General organization and main modules of POLIMIP

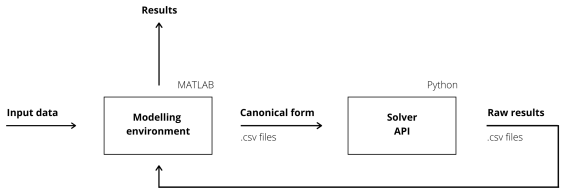


Figure 3. Multi-language architecture adopted to implement advanced solver functionalities not supported by the MATLAB API of the solver

2.3 Full-MILP algorithms for the optimal design of MES

At this point, before introducing MES decomposition methods, it is important to give a clear overview of the most widespread techniques adopted nowadays to reduce the complexity of MILP problems for MES design:

- *Model simplification*: it is possible to significantly reduce numerical complexity by directly acting on one or more of the modelling choices behind the mathematical description of the energy system (simplifying or even neglecting some of its variables and/or governing equations). Of course, this leads to a less accurate physical description of the system's features, hence resulting in a reduced soundness of the objective function;

- *Time aggregation*: an exponential increase of the size of the combinatorial solution space with the number of time steps is observed when binary or integer operation variables (typically used to model the on/off logic of the generators) are introduced in the MILP formulation of the problem. To mitigate this problem it is possible to increase the length of the time-step, hence reducing the number of interger/binary operation variables. Of course, the main drawback is a lower representativity of the system's dynamics;

- *Solution space reduction*: the size of the combinatorial solution space of a MILP problem actually explored by a branch-and-bound search algorithm can be significantly reduced by providing an initial upper bound associated with a sub-optimal integer feasible solution [46]. In this way, all the nodes leading to a local objective value higher than the initial cutoff value will be fathomed due to sub-optimality, thus reducing the number of explored combinations. As far as MES are concerned, an easy but effective way to find an initial incumbent is by guessing a reasonably oversized and/or redundant design and by solving the associated integer operation problem;

- *Decomposition methods*: decomposition methods are at the same time the best performing and the less intuitive among the four complexity reduction strategies presented in this section. In one line, they can be intended as *custom branch-and-bound search strategies exploiting the peculiarities of the problem's structure to reduce its combinatorial complexity* [47]. This is mainly done by breaking the original problem into smaller sub-problems that are easier to solve and that eventually provide integer cuts to reduce the size of the explored portion of the original search tree. The main price to be paid for this higher effectiveness is a marked problem-dependency. Moreover, their numerical implementation is quite a hard task since it necessarily requires an in-depth knowledge and control of the solution algorithm adopted by numerical solver. The novel decomposition algorithm proposed in the following chapter aims precisely at mitigating these two main limitations: (i) by broadening the class of treatable MES problems with respect to previous methods and (ii) by automatizing the numerical definition and implementation of their decomposed formulation on the widely adopted commercial solver IBM CPLEX.

3. A novel MILP decomposition algorithm allowing for mixed integer and continuous MES design

Mixed-Integer Linear Programming is undeniably one of the most promising approaches for design optimization of MES, since it combines high modelling flexibility with the capability of ensuring global optimality. The main issue associated to MILP optimal design problems is the computational time required by the search algorithm. In MILP problems, numerical complexity increases exponentially with the number of binary and integer variables due to the expansion of the branch and bound tree. In particular, in MES design the number of operation binary variables depends on the selected time horizon of the problem. So, when increasing the time horizon of unit commitment from some representative days to entire

weeks or months in order to enhance the representativeness of the solution, the solver may struggle in finding the optimal solution in an acceptable amount of time.

3.1 An introduction to MILP decomposition of MES problems: exploiting the internal hierarchy between design and operation

As previously stated, MILP decomposition methods are an effective way to reduce computational complexity without recurring to detrimental simplifications of the original model of the energy system. This is possible by effectively exploiting some peculiar characteristics of the problem's structure that typically make these approaches highly problem-dependent. This high problem-dependency is also one of the reasons why very few MES decomposition methods are actually found in literature. In fact, only few authors focused their efforts in developing original decomposition algorithms for MES, being this a task requiring a combination of in-depth knowledge coming both from energy modelling and from applied mathematics. Among these authors, Iyer and Grossmann [5] and Yokoyama [6] proposed two of the most cited and highly performing hierarchical decomposition methods for MES that inspired our novel approach. An important feature of both these approaches are their strict assumptions on the problem structure. To highlight these restriction and to better understand the main limits and potentialities coming from them, we will introduce a general MILP formulation of a MES design problem that we will compare with the two formulations required by the methods just introduced. A general MES design model can be formulated as the following MILP problem:

$$\begin{aligned} \min \quad & f\big(z_D, y_D, x_D, z_O, y_O, x_O\big) \\ \text{s.t.} \quad & l\big(z_D, y_D, x_D, z_O, y_O, x_O\big) \leq 0 \\ & z_D \in \{0,1\}^{n_z} \\ & y_D \in \mathbb{Z}^{n_y} \\ & x_D \in \mathbb{R}^{n_x} \\ & z_O \in \{0,1\}^{m_z} \\ & y_O \in \mathbb{Z}^{m_y} \\ & x_O \in \mathbb{R}^{m_x} \end{aligned} \tag{2}$$

where f is the objective function (a linear combination of all the problem's variables), z_D, y_D and x_D are the vectors containing the n_z binary, n_y integer and n_x continuous design variables, while z_O, y_O and x_O are the vectors containing the m_z binary, m_y integer and m_x continuous operation variables defining the problem. Effective formal decomposition for problem (2) are based on the observation that MES design problems are always characterized by an inherent hierarchy between design and operation variables: by fixing integer design variable (also known as *high-hierarchy* variables) (e.g. the investment decision on a given unit), we activate processes involving integer operation variables (also known as *low-hierarchy* variables) (e.g. on/off status variables of that unit).

To better understand these concepts, it is possible consider *branching priorities* as the simplest form of hierarchical decomposition. Applying branching priorities to integer design variables, the binary search tree starts exploring the general problem branching only high-hierarchy variables (*upper level*). Going deep into the tree we find a node where all the high-hierarchy variables are well constrained and assume an integer value, while low-hierarchy variables are still relaxed. We found an *entrance node* to the *lower level*. In the lower level, the branching procedure continues considering fixed values of the high-hierarchy variables, hence restoring the coherence between the two levels and providing a solutions respecting all the constraints of the original problem. Operating a simple ordering on the integer variables to be branched, branching priorities technique can be applied to the general formulation (2) without any assumption on the model, but it does not solve the problem of combinatorial nature of MILP variable domain.

Decomposition by Iyer and Grossmann With respect to this general formulation, the model by Iyer and Grossmann [5] introduces the following restricting assumptions:

- There are no integer design variables: $n_y = 0$;
- There are no integer operation variables: $m_y = 0$;
- Each binary design variable represents the investment variable of a potential unit;
- Each binary operation variable represents the on-off variable of a potential unit in a specific time-step;
- Each continuous design variable represents the capacity of an associated unit: $n_x = n_z := n$.

The resulting reference formulation is then:

$$\begin{aligned} \min \quad & f_D(z_D, x_D) + \sum f_{O_t}(z_{O_t}, x_{O_t}) \\ \text{s.t.} \quad & g_t\big(z_D, x_D, z_{O_t}, x_{O_t}\big) \leq 0 \quad \forall t \in \{1,\dots,T\} \\ & h\big(z_D, x_D, z_{O_1}, x_{O_1}, z_{O_2}, x_{O_2}, ..., z_{O_T}, x_{O_T}\big) \leq 0 \\ & z_D \in \{0,1\}^n \\ & x_D \in \mathbb{R}^n \\ & z_{O_t} \in \{0,1\}^n \qquad \forall t \in \{1,\dots,T\} \\ & x_{O_t} \in \mathbb{R}^n \qquad \forall t \in \{1,\dots,T\} \end{aligned} \tag{3}$$

where the problem constraints l have been divided into two main categories: the *uncoupled constraints* g_t (constraints containing only operation variables indexed on a single time-step, such as energy balances) and the *coupling constraints* h (constraints containing operation variables indexed on more than one time-step, such as minimum up-time requirements). It is worth noting that,

being f a linear combination of the problem's variables, it can always be expressed as a sum of terms depending on a smaller subset of variables.

Decomposition by Yokoyama Moving to the method by Yokoyama et al. [6], the restricting assumptions on the problem structure are the followings:

- There are no continuous design variables: $n_x = 0$;
- There are no coupling constraints h.

The resulting reference formulation is then:

$$\begin{aligned}
\min\ & f_D(z_D, y_D) + \sum f_{O_t}(z_{O_t}, y_{O_t}, x_{O_t}) \\
\text{s.t.}\ & g_t(z_D, y_D, z_{O_t}, y_{O_t}, x_{O_t}) \leq 0 \qquad \forall t \in \{1,\dots,T\} \\
& z_D \in \{0,1\}^{n_z} \\
& y_D \in \mathbb{Z}^{n_y} \\
& z_{O_t} \in \{0,1\}^{m_z} \qquad \forall t \in \{1,\dots,T\} \\
& y_{O_t} \in \mathbb{Z}^{m_y} \qquad \forall t \in \{1,\dots,T\} \\
& x_{O_t} \in \mathbb{R}^{m_x} \qquad \forall t \in \{1,\dots,T\}
\end{aligned} \tag{4}$$

The strong simplification of discarding both continuous variables and coupling constraints allowed the authors to divide the single operation subproblem into a series of T lighter and independent ones. Moreover, the integer design variables y_D are restored in the upper level, allowing the modelling of discrete equipment capacities. In the light of the above, it is possible to compare the decomposition methods by Iyer and Grossmann [5] and by Yokoyama et al. [6] on the basis of four main features related to the specific assumptions on the MES model structure.

Feature	*Iyer and Grossmann*	*Yokoyama et al.*
Continuous design variables	Yes	No
Coupling constraints	Yes	No
Discrete capacities	No	Yes
Solve complete operation	Always	Never

In brief:

- The method by Iyer and Grossman can be applied to MES models containing time-coupling constraints and/or continuous equipment capacities, but it is not suitable for models with discrete sizes of the units. Moreover, it always solves the complete operation worker problem (defined on the whole time horizon), hence not exploiting at all the multi-period structure found in most MES models to reduce the computational time of the solver;
- The method by Yokoyama et al. can be applied to MES models with discrete equipment capacities, but it is not suitable for models with time-coupling constraints and/or with continuous design variables. On the other hand, unlike the method by Iyer and Grossmann, the operation subproblem is never solved as a whole, hence effectively exploiting the multi-period structure of the MES model to reduce the combinatorial complexity of the search tree.

It is clear that the two methods are perfectly complementary. As a consequence, MES models exhibiting a "hybrid" set of modelling features, such as a catalogue of generating units characterized by both discrete and continuous capacities, cannot be solved by any of the decomposition methods previously introduced.

3.2 Mathematical formulation of the novel hierarchical decomposition method for MES design optimization

The novel decomposition method proposed in this Thesis work aims precisely at overcoming the limiting trade-off affecting the two approaches discussed above. In particular, its development has been specifically oriented at fulfilling the following three main requirements:

1. **Universality**: being applicable to any MES problem having the general form reported in eq. (2);
2. **Performance**: being faster than a general-purpose search algorithm in finding the global optimum;
3. **Usability**: being easy to implement without advanced programming skills.

To fulfill the universality condition, in our method we refer to the following form of the general formulation of a MES problem reported in eq. (2):

General problem

$$\begin{aligned}
\min\ & f_D(z_D, y_D) + f_{D^*}(x_D) + \sum f_{O_n}(z_{O_n}, y_{O_n}, x_{O_n}) \\
\text{s.t.}\ & g_n(z_D, y_D, x_D, z_{O_n}, y_{O_n}, x_{O_n}) \leq 0 \quad \forall n \in \{1,\dots,N\} \\
& h(z_D, y_D, x_D, z_{O_1}, y_{O_1}, x_{O_1}, ..., z_{O_N}, y_{O_N}, x_{O_N}) \leq 0 \\
& z_D \in \{0,1\}^{n_z} \\
& y_D \in \mathbb{Z}^{n_y} \\
& x_D \in \mathbb{R}^{n_x} \\
& z_{O_n} \in \{0,1\}^{m_z} \quad \forall n \in \{1,\dots,N\} \\
& y_{O_n} \in \mathbb{Z}^{m_y} \quad \forall n \in \{1,\dots,N\} \\
& x_{O_n} \in \mathbb{R}^{m_x} \quad \forall n \in \{1,\dots,N\}
\end{aligned} \tag{5}$$

where we assumed without loss of generality that the time horizon is composed of N typical periods of T time steps each[4] (e.g. 4 typical days of 24 hours each or 2

[4] Such indexing choice is particularly convenient to impose periodic constraints (e.g. the state of charge of a storage element at the end of each typical period must be higher than that at the beginning of the same period.)

typical weeks of 168 hours each) and we made explicit the coupling constraints h from the uncoupled constraints g_n. Moreover, we chose to rewrite the objective function f as the sum of $2+N$ terms, each containing only certain types of variables. It is worth noting that this can always be done due to the linearity of f with respect to the problem variables.

Starting from (5), the inherent hierarchy between design and operation variables can be exploited by defining a *master design problem* with relaxed operation variables and a *worker pseudo-operation problem* aimed at restoring the integer feasibility of z_{O_n} and y_{O_n} and at determining the value of x_D and x_{O_n} for a given combination $(\bar{z}_D, \bar{y}_D)$ of binary and integer design variables:

Master Problem (MP)

$$\begin{aligned}
\min \quad & f_D(z_D, y_D) + f_{D^*}(x_D) + \sum f_{O_n}(z_{O_n}, y_{O_n}, x_{O_n}) \\
\text{s.t.} \quad & g_n(z_D, y_D, x_D, z_{O_n}, y_{O_n}, x_{O_n}) \leq 0 \quad \forall n \in \{1,\dots,N\} \\
& h(z_D, y_D, x_D, z_{O_1}, y_{O_1}, x_{O_1}, ..., z_{O_N}, y_{O_N}, x_{O_N}) \leq 0 \\
& z_D \in \{0,1\}^{n_z} \\
& y_D \in \mathbb{Z}^{n_y} \\
& x_D \in \mathbb{R}^{n_x} \\
& z_{O_n} \in \mathbb{R}^{m_z} \quad \forall n \in \{1,\dots,N\} \\
& y_{O_n} \in \mathbb{R}^{m_y} \quad \forall n \in \{1,\dots,N\} \\
& x_{O_n} \in \mathbb{R}^{m_x} \quad \forall n \in \{1,\dots,N\}
\end{aligned} \tag{6}$$

Worker Problem (WP)

$$\begin{aligned}
\min \quad & f_D(z_D, y_D) + f_{D^*}(x_D) + \sum f_{O_n}(z_{O_n}, y_{O_n}, x_{O_n}) \\
\text{s.t.} \quad & g_n(z_D, y_D, x_D, z_{O_n}, y_{O_n}, x_{O_n}) \leq 0 \quad \forall n \in \{1,\dots,N\} \\
& h(z_D, y_D, x_D, z_{O_1}, y_{O_1}, x_{O_1}, ..., z_{O_N}, y_{O_N}, x_{O_N}) \leq 0 \\
& z_D \in \{0,1\}^{n_z} \\
& y_D \in \mathbb{Z}^{n_y} \\
& x_D \in \mathbb{R}^{n_x} \\
& z_{O_n} \in \{0,1\}^{m_z} \quad \forall n \in \{1,\dots,N\} \\
& y_{O_n} \in \mathbb{Z}^{m_y} \quad \forall n \in \{1,\dots,N\} \\
& x_{O_n} \in \mathbb{R}^{m_x} \quad \forall n \in \{1,\dots,N\} \\
& y_D = \bar{y}_D \\
& z_D = \bar{z}_D
\end{aligned} \tag{7}$$

with m_z, m_y and m_x all higher than or equal to T. It must be noticed that, due to the presence of coupling variables (x_D) and constraints (h), a subdivision of the worker problem in smaller subproblems (as done in [6]) is not possible.

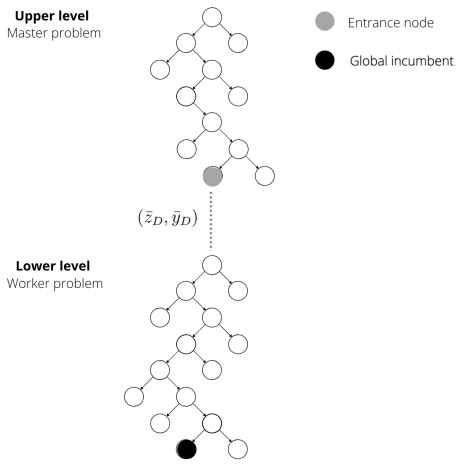


Figure 4. Graphical representation of concepts like "upper level", "lower level", "master problem", "worker problem", "integer design candidate", "entrance node" and "global incumbent"

By directly implementing the above decomposition without any further expedient we would obtain a behaviour like that depicted in figure 4.

The branching tree of the MP (6) (also known as *upper level*) is explored until an *integer design candidate* $(\bar{z}_D, \bar{y}_D)$ is found. When this happens, we say that we have found an *entrance node* to the lower level. At this point the information about $(\bar{z}_D, \bar{y}_D)$ is transferred to the WP (7), which is immediately set up and solved. The branching tree of problem (7) is also known as *lower level*. If the solution to problem (7) is both feasible and optimal, then it represents a *global incumbent* and the value of its objective function a *global cutoff*. Otherwise, the lower level is exited prematurely and the corresponding value of $(\bar{z}_D, \bar{y}_D)$ is discarded. In any case, the search process of a new integer design candidate is resumed in the upper level. These steps are repeated iteratively until a convergence criteria is met (typically, a specific value of the relative gap).

By following this approach we would only exploit the hierarchical relationship between the integer design and operation variables, without relying at all on the multi-period structure of the problem, exactly like branching priorities. In particular, by solving the WP we would obtain an information with the highest value possible: either we find that the solution is infeasible/suboptimal or we find a new global cutoff. Nonetheless, the computational effort required to extract such information would be the highest as well, since an extensive exploration of the subtree in the lower level is required each time

an integer design candidate is found in the upper level. This can become a non-negligible burden, in particular if the time horizon of the unit committment problem is significantly extended: the longer the time horizon, the more severe the impact on the computational time due to the exponential complexity of the subtree to be explored. To support this, it is worth recalling that the approach by Yokoyama and al. aims precisely at avoiding the complete resolution of problem (7) by giving up coupling variables and constraints in the model formulation. In the following we propose an alternative decomposition paradigm aiming at avoiding the resolution of the WP for each design candidate $(\bar{z}_D, \bar{y}_D)$ without adding any restricting assumption on the general problem structure reported in eq. (5), thus ensuring the universality of the method.

To this aim, the innovative concept of *local auxiliary problem* $\bar{P}'$ is introduced. In general, we define as *auxiliary problem* a modified version of an *original problem* P which is easier to solve and that provides a valid lower bound for the objective function of P (or for a part of it). Following this definition, an auxiliary problem differs from a WP in that it is completely optional (that is, it is not required to ensure the coherence between upper and lower level). In a decomposition method, an auxiliary problem is an *additional* subproblem that can be solved to provide useful cuts aimed at reducing the size of the combinatorial solution space, thus improving the overall computational performance of the algorithm. In the context of a hierarchical decomposition of a MES problem, a *local auxiliary problem* $\bar{P}'$ is defined as an auxiliary problem whose original problem $\bar{P}$ is a WP (7). On the other hand, a *global auxiliary problem* $\check{P}'$ is defined as an auxiliary problem whose original problem $\check{P}$ is the general problem (5). A global auxiliary problem differs from a local one in that its solution does provide a valid lower bound not only for the general problem, but also *all* the possible WPs. This is due to the fact that its original problem is an under-constrained version of the WP (7), from which it is obtained by removing the constraints $z_D = \bar{z}_D$ and $y_D = \bar{y}_D$). Conversely, a local auxiliary problem provides a lower bound which is only valid for its original WP.

An important property of local auxiliary problems is that, being derived by a over-constrained version of the general problem, their lower bounds are always higher than those provided by a corresponding[5] global auxiliary problem. This property is not exploited by any of the decomposition methods previously discussed, and represents the main original feature of the novel decomposition method here proposed. In fact, by relying on a more effective generation of lower bounds in the lower level (thanks to the definition of suitable local auxiliary problems), we are able to overcome the trade-off between universality and performance that makes the two algorithms analyzed before complementary, going towards a highly performing decomposition method which is also universally applicable. In our case, local auxiliary problems are simplified operation subproblems obtained from a corresponding WP by removing one or more constraints. In the proposed decomposition method, they are specifically used to effectively prove the sub-optimality or infeasibilty of a given integer design candidate $(\bar{z}_D, \bar{y}_D)$ before solving the associated complete WP (7).

Following the search scheme reported in figure 4, when the hierarchical B&B search algorithm reaches the entrance node, the integer design candidate $(\bar{z}_D, \bar{y}_D)$ (and so the value of $\bar{f}_D$) are fixed. At this point, instead of solving the corresponding WP on the full time horizon and compute the optimized value of the remaining part of f, it is possible to define smaller auxiliary problems to separately bound the contributions f_{D^*} and f_{O_n} given by the coupling variables x_D and by the MES operation in the different typical periods. Being additional and optional, auxiliary problems must have two main characteristics:

1. They must be very fast to solve;
2. They must provide valuable information to speed up the B&B search algorithm (e.g. by generating useful integer cuts for the MP).

The aim is to find the best trade-off between the computational time required to solve the additional subproblems and the reduction of combinatioral complexity obtained through the information extracted from each auxiliary problem.

In general, for a MES problem, we can classify auxiliary problems on the basis of four main *constraining features* here defined:

Fixed integer design *All the integer design variables of the MP are fixed to a certain integer value.* It allows to reduce the number of integer variables (directly but also indirectly, constraining corresponding operation variables of non-selected units) to branch and so the combinatorial complexity of the search tree. This is an inevitable condition for all the WPs defined in the lower level, where the design candidate is set at the entrance node and represents the actual difference between local and global auxiliary problems.

Coupling *The problem is solved on the whole time horizon.* This characteristics is required if we want

[5] By *corresponding* we mean problems optimizing the same part of objective function

to obtain significant information about the contribution given to the objective function by design variables, which need to assume a unique value on the whole time horizon to maintain a physical meaning. We can say they must respect a "coupling constraint". Nonetheless, both global and local auxiliary problems used to bound the objective function of the WPs may take advantage of the subdivision in typical periods to solve problems defined only on a single period, significantly reducing the number of integer variables of the problem and thus its complexity.

Integrality *The problem is solved maintaining all the integrality constraints active.* This characteristic allows to obtain a value of the objective function well-representing the one of the MP, but can be removed on certain variables of the WP to simplify the search tree and compute lower bounds of the objective function very quickly.

Objective coupling *The problem is solved optimizing the whole objective function.* Of course, in general all the different part of the objective function are inevitably interconnected, since design decisions influence the optimal operation strategy. In order to extract precise information about a specific term of f, auxiliary problems can be defined giving up the trade-off and optimizing only a single part of the objective function.

A local auxiliary problem can be properly defined starting from a WP by keeping the *fixed integer design* constraint feature and by dropping one or more between *coupling*, *integrality* and *objective coupling*. In fact, by also dropping the *fixed integer design* constraining feature we obtain a global auxiliary problem. It must be stressed out that the only way to obtain valid lower bounds for each term of f *separately* is by dropping the objective coupling constraining feature (that is, by defining local auxiliary problems in which each of the terms f_D, f_{D^*} and f_{O_n} is optimized on its own). This is a necessary requirement to effectively exploit the multi-period structure of the problem and eventually avoid the solution of the unit committment on the whole time horizon for a given integer design candidate.

The global auxiliary problems (GAP) set up to obtain a set of valid global lower bounds for f_D, f_{D^*} and f_{O_n} are defined starting from the general problem (5) by dropping the objective coupling and coupling constraining features and by neglecting the coupling constraints h. We have then:

GAP for f_D

$$\begin{aligned} \min\ & f_D\big(z_D, y_D\big) \\ \text{s.t.}\ & g_n\big(z_D, y_D, x_D, z_{O_n}, y_{O_n}, x_{O_n}\big) \leq 0 \\ & z_D \in \{0,1\}^{n_z} \\ & y_D \in \mathbb{Z}^{n_y} \\ & x_D \in \mathbb{R}^{n_x} \qquad \forall n \in \{1,...,N\} \\ & z_{O_n} \in \{0,1\}^{m_z} \\ & y_{O_n} \in \mathbb{Z}^{m_y} \\ & x_{O_n} \in \mathbb{R}^{m_x} \end{aligned} \tag{8}$$

GAP for f_{D^*}

$$\begin{aligned} \min\ & f_{D^*}\big(x_D\big) \\ \text{s.t.}\ & g_n\big(z_D, y_D, x_D, z_{O_n}, y_{O_n}, x_{O_n}\big) \leq 0 \\ & z_D \in \{0,1\}^{n_z} \\ & y_D \in \mathbb{Z}^{n_y} \\ & x_D \in \mathbb{R}^{n_x} \qquad \forall n \in \{1,...,N\} \\ & z_{O_n} \in \{0,1\}^{m_z} \\ & y_{O_n} \in \mathbb{Z}^{m_y} \\ & x_{O_n} \in \mathbb{R}^{m_x} \end{aligned} \tag{9}$$

GAP for f_{O_n}

$$\begin{aligned} \min\ & f_{O_n}\big(z_{O_n}, y_{O_n}, x_{O_n}\big) \\ \text{s.t.}\ & g_n\big(z_D, y_D, x_D, z_{O_n}, y_{O_n}, x_{O_n}\big) \leq 0 \\ & z_D \in \{0,1\}^{n_z} \\ & y_D \in \mathbb{Z}^{n_y} \\ & x_D \in \mathbb{R}^{n_x} \qquad \forall n \in \{1,...,N\} \\ & z_{O_n} \in \{0,1\}^{m_z} \\ & y_{O_n} \in \mathbb{Z}^{m_y} \\ & x_{O_n} \in \mathbb{R}^{m_x} \end{aligned} \tag{10}$$

Both the values of f_D and f_{D^*} are optimized N times (once for each typical period). Hence, since we are looking for a lower bound, only the highest among all the solutions to problems (8) and (9) are considered. Being completely independent of the specific values of z_D and y_D, all these problems can be solved before starting the actual B&B search. Moreover, since they are obtained from a relaxed version of the general problem (5), their solutions provide a set of global lower bounds $\check{\underset{\sim}{f}}_D$, $\check{\underset{\sim}{f}}_{D^*}$ and $\check{\underset{\sim}{f}}_{O_n}$ for f_D, f_{D^*} and f_{O_n} valid both in the upper and in the lower level that can be used to compute the following lower bounds for f and $\bar{f}$:

$$\underset{\sim}{f}^{\circ} := \check{\underset{\sim}{f}}_D + \check{\underset{\sim}{f}}_{D^*} + \sum \check{\underset{\sim}{f}}_{O_n} \leq f \tag{11}$$

$$\underset{\sim}{\bar{f}}^{\circ} := \bar{f}_D + \check{\underset{\sim}{f}}_{D^*} + \sum \check{\underset{\sim}{f}}_{O_n} \leq \bar{f} \tag{12}$$

where $\bar{f}$ and $\bar{f}_D$ are the values of f and f_D associated with a given integer design candidate $(\bar{z}_D, \bar{y}_D)$.

The local auxiliary problem (LAP) set up to obtain a valid local lower bound for the f_{D^*} term of the objective function associated with a certain integer design candidate $(\bar{z}_D, \bar{y}_D)$ is defined starting from the WP (7) by dropping the objective coupling and the integrality constraining features. We have then:

LAP for f_{D^*}

$$\begin{aligned}
\min\ & f_{D^*}(x_D) \\
\text{s.t.}\ & g_n(z_D, y_D, x_D, z_{O_n}, y_{O_n}, x_{O_n}) \leq 0 \quad \forall n \in \{1,\dots,N\} \\
& h(z_D, y_D, x_D, z_{O_1}, y_{O_1}, x_{O_1}, ..., z_{O_N}, y_{O_N}, x_{O_N}) \leq 0 \\
& z_D \in \{0,1\}^{n_z} \\
& y_D \in \mathbb{Z}^{n_y} \\
& x_D \in \mathbb{R}^{n_x} \\
& z_{O_n} \in \mathbb{R}^{m_z} \qquad \forall n \in \{1,\dots,N\} \\
& y_{O_n} \in \mathbb{R}^{m_y} \qquad \forall n \in \{1,\dots,N\} \\
& x_{O_n} \in \mathbb{R}^{m_x} \qquad \forall n \in \{1,\dots,N\} \\
& y_D = \bar{y}_D \\
& z_D = \bar{z}_D
\end{aligned} \tag{13}$$

Being dependent on the specific values of $\bar{z}_D$ and $\bar{y}_D$, these problems can only be solved in the lower level after each entrance node. Moreover, since they are obtained from a relaxed version of the WP (7), their solution provides a local lower bound $\underset{\sim}{\bar{f}}_{D^*}$ for f_{D^*} only valid in the subtree associated with the integer design candidate $(\bar{z}_D, \bar{y}_D)$.

It must be noticed that problem (13) is not a restricted version of the global auxiliary problem for f_{D^*}, since it contains both a restriction ($z_D = \bar{z}_D$, $y_D = \bar{y}_D$ and the coupling constraint h) and a relaxation ($z_{O_n} \in \mathbb{R}^{m_z}$, $y_{O_n} \in \mathbb{R}^{m_y}$ $\forall n \in \{1,\dots,N\}$). As a consequence, $\underset{\sim}{\bar{f}}_{D^*}$ may be lower or higher than $\underset{\sim}{\check{f}}_{D^*}$, depending whether the prevailing effect is that of the restriction or that of the relaxation. In any case, we can write that:

$$\max\left(\underset{\sim}{\check{f}}_{D^*}, \underset{\sim}{\bar{f}}_{D^*}\right) - \underset{\sim}{\check{f}}_{D^*} \geq 0 \tag{14}$$

Eq.(14) is trivial and always true, but useful to justify following steps. The local auxiliary problems (LAP) set up to obtain some valid local lower bounds for the f_{O_n} terms of the objective function associated with a certain integer design candidate $(\bar{z}_D, \bar{y}_D)$ are defined starting from the WP (7) by dropping the objective coupling and the coupling constraining features (that is, by defining one problem for each typical period) and by relaxing the coupling constraints h. We have then:

LAP for f_{O_n}

$$\begin{aligned}
\min\ & f_{O_n}(z_{O_n}, y_{O_n}, x_{O_n}) \\
\text{s.t.}\ & g_n(z_D, y_D, x_D, z_{O_n}, y_{O_n}, x_{O_n}) \leq 0 \\
& z_D \in \{0,1\}^{n_z} \\
& y_D \in \mathbb{Z}^{n_y} \\
& x_D \in \mathbb{R}^{n_x} \qquad \forall n \in \{1,\dots,N\} \\
& z_{O_n} \in \{0,1\}^{m_z} \\
& y_{O_n} \in \mathbb{Z}^{m_y} \\
& x_{O_n} \in \mathbb{R}^{m_x} \\
& y_D = \bar{y}_D \\
& z_D = \bar{z}_D
\end{aligned} \tag{15}$$

Being dependent on the specific values of z_D and y_D, these problems can only be solved in the lower level after each entrance node. Moreover, since they are obtained from a relaxed version of the WP (7), their solution provides a local lower bound $\underset{\sim}{\bar{f}}_{O_n}$ for f_{O_n} only valid in the subtree associated with the integer design candidate $(\bar{z}_D, \bar{y}_D)$.

It must be noticed that problem (13) is actually a restricted version of the global auxiliary problem for f_{D^*}, since it can be obtained from it by adding the additional constraints $z_D = \bar{z}_D$ and $y_D = \bar{y}_D$. As a consequence, $\underset{\sim}{\bar{f}}_{O_n}$ is always higher than $\underset{\sim}{\check{f}}_{O_n}$:

$$\underset{\sim}{\bar{f}}_{O_n} - \underset{\sim}{\check{f}}_{O_n} \geq 0 \quad \forall n \in \{1,\dots,N\} \tag{16}$$

At this point, thanks to the $N+1$ eqs. (14) and (16) and to eq. (12), we are able to write the following chain of inequalities providing $N+2$ local lower bounds for $\bar{f}$, in which the $N+1$ ones indicated as $\underset{\sim}{\bar{f}}^{(k)}$ (with $k \in \{1,...,N+1\}$) are those obtained after solving k among the $N+1$ local auxiliary problems defined before for a given integer design candidate:

$$\underset{\sim}{\bar{f}}^{\circ} \leq \underset{\sim}{\bar{f}}^{(1)} \leq \underset{\sim}{\bar{f}}^{(2)} \leq \dots \leq \underset{\sim}{\bar{f}}^{(N+1)} \leq \bar{f} \tag{17}$$

where $\underset{\sim}{\bar{f}}^{(k)}$ is equal to

$$\underset{\sim}{\bar{f}}^{(k)} := \bar{f}_D + \max\left(\underset{\sim}{\check{f}}_{D^*}, \underset{\sim}{\bar{f}}_{D^*}\right) + \sum_{n \in U} \underset{\sim}{\check{f}}_{O_n} + \sum_{n \in S} \underset{\sim}{\bar{f}}_{O_n} \tag{18}$$

or to

$$\underset{\sim}{\bar{f}}^{(k)} := \bar{f}_D + \underset{\sim}{\check{f}}_{D^*} + \sum_{n \in U} \underset{\sim}{\check{f}}_{O_n} + \sum_{n \in S} \underset{\sim}{\bar{f}}_{O_n} \tag{19}$$

depending on whether the local auxiliary problem for f_{D^*} is among the k local auxiliary problems already solved or not. In the above definitions, S is the set of indexes of the k (or $k-1$) local auxiliary problems for f_{O_n} already solved, while U is the set of indexes of the

$N-k$ (or $N-(k-1)$) local auxiliary problems for f_{O_n} still unsolved.

At this point we are ready to define a solution algorithm exploiting the valuable information provided by the global and local auxiliary problems previosuly defined to periodically generate valid local integer cuts[6]. These cuts are used to prune the subtree rooted at the node in which they are generated, thus significantly reducing the total number of explored nodes both in the upper and in the lower level. A local integer cut is enforced in the upper level every time a valid lower bound $\underset{\sim}{f}$ for f exceeds the current cutoff value $\tilde{f}$. A local integer cut is enforced in the lower level every time a valid lower bound $\underset{\sim}{\bar{f}}$ for $\bar{f}$ exceeds the current cutoff value $\tilde{f}$. For our purposes, an integer cut enforced at a node j is a linear constraint of the form:

$$f < (f^*)_j \qquad \text{(valid locally)} \tag{20}$$

where $(f^*)_j$ is the solution of the LP relaxation of node j. In this way, node j is immediately fathomed due to violation of constraint (20), and the corresponding subtree pruned. To better understand the usefulness of lower bounds in generating integer cuts it is possible to notice that the knowledge of a global lower bound $\underset{\sim}{f}$ in a node in the upper level may determine a premature pruning of the corresponding subtree, hence completely avoiding the generation of an integer design candidate and the exploration of the associated tree in the lower level.

Description of the solution algorithm The detailed algorithmic steps of the solution scheme are reported below (CPLEX is used as the reference solver):

1. The global auxiliary problems are solved to obtain and store the values of the global lower bounds $\check{\underset{\sim}{f}}_D$, $\check{\underset{\sim}{f}}_{D^*}$ and $\check{\underset{\sim}{f}}_{O_n}$;

2. A queue Q containing the indexes of the local auxiliary problems is initialized with an arbitrary order;

3. The global cutoff is initialized to a sufficiently high value and the search in the upper level is started;

4. A generic callback [48] is called at each node j in the upper level after solving the corresponding LP relaxation. If the elements of z_D and y_D contain at least one fractional value, then the following lower bound for f is computed and compared with the global cutoff value:

$$\underset{\sim}{f}_j = \max\left((f^*_D)_j, \check{\underset{\sim}{f}}_D\right) + \max\left((f^*_{D^*})_j, \check{\underset{\sim}{f}}_{D^*}\right) + \\ + \sum \max\left((f^*_{O_n})_j, \check{\underset{\sim}{f}}_{O_n}\right) \tag{21}$$

where $(f^*_D)_j$, $(f^*_{D^*})_j$ and $(f^*_{O_n})_j$ are the values of f_D, f_{D^*} and f_{O_n} evaluated for the solution of the LP relaxation of node j:

   - If $\underset{\sim}{f}_j < \tilde{f}$ then the search of the current tree branch is continued by branching on node j;
   - If $\underset{\sim}{f}_j \geq \tilde{f}$ then the search of the current tree branch is interrupted by fathoming node j through a suitable integer cut.

   If, on the other hand, all the elements of z_D and y_D have binary/integer values, then an integer design candidate $(\bar{z}_D, \bar{y}_D)$ has been found, and node j is labeled as an entrance node. The lower level is entered and the Q is re-ordered[7];

5. In the lower level, the values of $\bar{f}_D$ and $\underset{\sim}{\bar{f}}^\circ$ associated with the integer design candidate $(\bar{z}_D, \bar{y}_D)$ are immediately computed:

   - If $\bar{f}_D < \check{\underset{\sim}{f}}_D$ we are sure that the current integer design candidate will eventually lead to an unfeasible solution. This is due to the fact that $\check{\underset{\sim}{f}}_D$ is a globally valid lower bound for f_D. In this case, the lower level is immediately exited without solving any auxiliary/worker problem and the search of a new integer design candidate in the upper level is resumed;
   - If $\bar{f}_D \geq \check{\underset{\sim}{f}}_D$ then we can't conclude anything about the feasibility of $(\bar{z}_D, \bar{y}_D)$ and the search algorithm remains in the lower level.

   Moreover:

   - If $\underset{\sim}{\bar{f}}^\circ \geq \tilde{f}$ we can conclude that, since $\underset{\sim}{\bar{f}}^\circ \leq \bar{f}$, the current integer design candidate will surely lead to a sub-optimal solution: the lower level

[6] An integer cut is *valid* if it only cuts out sub-optimal or infeasible integer solutions. An integer cut is *local* if it is only enforced for the subtree rooted at the node where the cut is generated.

[7] Thanks to eq. (17) we are sure that each time a local auxiliary problem is solved in the lower level, it determines an increase $\underset{\sim}{\bar{f}}^{(k)} - \underset{\sim}{\bar{f}}^{(k-1)}$ or $\underset{\sim}{\bar{f}}^{(1)} - \underset{\sim}{\bar{f}}^\circ$ of the local lower bound for $\bar{f}$ which is equal to $\underset{\sim}{\bar{f}}_{O_n} - \check{\underset{\sim}{f}}_{O_n} \geq 0$ or to $\max\left(\check{\underset{\sim}{f}}_{D^*}, \underset{\sim}{\bar{f}}_{D^*}\right) - \check{\underset{\sim}{f}}_{D^*} \geq 0$, depending on the specific type of auxiliary problem. Similarly to what is done with the WPs of the algorithm by Yokoyama et al. [6], all these increases are stored in a list l_i indexed on the $N+1$ local auxiliary problems and are used to efficiently re-order the queue Q of local auxiliary problems each time the algorithm enters the lower level with a new integer design candidate. This is simply done by prioritizing the local auxiliary problems corresponding to the lists l_i characterised by the highest mean values of their stored increases. In this way, we are solving first those problems who will most likely provide the highest increase in $\bar{f}^{(k)}$, hence eventually leading to a faster proof of sub-optimality/infeasibility of the given integer design candidate.

is exited prematurely without solving any auxiliary/worker problem and the search of a new integer design candidate is resumed in the upper level;

- If $\underset{\sim}{\bar{f}}^{\circ} < \tilde{f}$ we can't conclude anything on the final value of $\underset{\sim}{\bar{f}}$: the algorithm sets up and solves the local auxiliary problem contained in the queue Q, starting from the first. The following valid upper bound is computed before solving each local auxiliary problem for f_{O_n}:

$$\tilde{f}_{O_n} = \tilde{f} - \left(\underset{\sim}{\bar{f}}^{(k-1)} - \underset{\sim}{\check{f}}_{O_n}\right) \tag{22}$$

6. The local auxiliary subproblem is solved and a feasibility check is carried out:
   - If the local auxiliary problem is infeasible, we can then conclude that, being it a under-constrained version of the worker problem, also the latter will be infeasible. In this case the lower level is exited prematurely without solving any further auxiliary/worker problem and the search of a new integer design candidate is resumed in the upper level;
   - If the local auxiliary problem is feasible, then the new local lower bound $\underset{\sim}{\bar{f}}^{(k)}$ is computed with the new $\underset{\sim}{\bar{f}}_{O_n}$ or $\max\left(\underset{\sim}{\bar{f}}_{D^*}, \underset{\sim}{\bar{f}}_{D^*}\right)$;
7. A sub-optimality check is done by comparing $\underset{\sim}{\bar{f}}^{(k)}$ with $\tilde{f}$:
   - If $\underset{\sim}{\bar{f}}^{(k)}$ is higher than the current cutoff value $\tilde{f}$, then the lower level is exited prematurely without solving any auxiliary/worker problem and the search of a new integer design candidate is resumed in the upper level;
   - If $\underset{\sim}{\bar{f}}^{(k)}$ is lower than the current cutoff value $\tilde{f}$, then the next auxiliary problem in the queue is set up and solved by the algorithm;
8. If all the $N+1$ local auxiliary problems are successfully solved and the final value $\underset{\sim}{\bar{f}}^{(N+1)}$ of the local lower bound is still lower than the current cutoff $\tilde{f}$, then the algorithm sets up and solves the worker problem to restore the coherence with the upper level. By means of a dedicated callback, the lower bound of this problem is continuously compared with the global cutoff at each node of the subtree: if it is higher, then the problem is aborted and the corresponding integer design candidate immediately discarded. A new cutoff value may be found or not depending on the feasibility and optimality of the solution of the worker problem. In any case, the algorithm resumes the search of a new integer design candidate in the upper level.

It is worth stressing that, in this solution scheme, the complete worker problem (step 8) is only solved if all the $N+1$ local auxiliary problems are feasible and contribute to a lower bound $\underset{\sim}{\bar{f}}^{(N+1)}$ which is lower than the global cutoff. Of course, a check of the relative gap is performed each time a generic callback is entered. The algorithm is stopped when the relative gap falls below a certain threshold.

It is important to underline that, each time the complete worker problem is solved (step 8), all the computational time spent in solving the local auxiliary problems becomes additional with respect to a reference case without any auxiliary problem (we are introducing an overhead). Hence, there exists a trade-off between the computational time saved when the algorithm exits the lower level before solving the worker problem and the overhead introduced when $\underset{\sim}{\bar{f}}^{(N+1)} < \tilde{f}$. Nevertheless, it is worth noting that the increase in computational time associated with a higher number of local auxiliary problems associated to a higher number of typical periods is approximatively *linear* (we are growing the number of subproblems without varying the number of integer operation variables in each one of them[8]). At the same time, by increasing N, the computational time required to solve the complete worker may potentially explode due to the combinatorial complexity of the problem (by linearly increasing the number of integer operation variables of the problem we are exponentially growing the number of possible discrete solutions). Considering this, we can reasonably expect that for problems defined over a sufficiently long time horizon, the time saved thanks to the additional integer cuts deriving from the local auxiliary problems will eventually prevail on the computational overhead introduced when $\underset{\sim}{\bar{f}}^{(N+1)} < \tilde{f}$, and that the highest the number of typical periods, the highest the benefit.

4. Assessment of the numerical performances of the novel decomposition algorithm on a real case study

In the last section, the numerical performances of the novel decomposition algorithm are tested on a real case study. The problem regards the optimal design of a multi-energy microgrid for the university campus of *Politecnico di Milano* called "Bovisa" in northern Italy. The MES must be designed to satisfy the demands of electricity, heat and cooling for all the buildings and the laboratories during the year. The system has been modelled to include both integer and continuous design variables, being therefore incompatible with both the methods by Iyer and Grossman and by Yokoyama et al.

[8] Of course, we are also increasing the number of continuous operation variables x_{O_n}, but this only slows the solution time of the LP relaxations, without affecting the size of the combinatorial solution space.

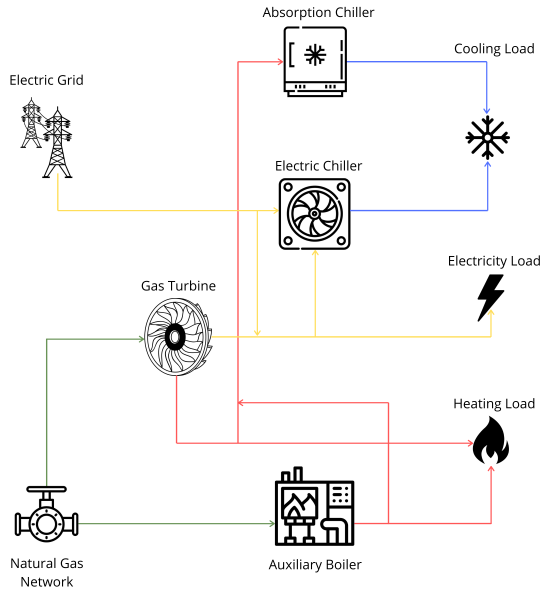


Figure 5. The microgrid architecture of the Bovisa case study

4.1 Description and modelling of the Bovisa case study

The architecture of the system is presented in fig.5. The microgrid includes cogenerative gas turbines for the combined generation of electricity and heat, auxiliary boilers and refrigerators including both absorption and electric chillers. Each technology is available in the form of a catalogue that contains different models characterised by various performance parameters, sizes and costs. For each technology is possible to select only a single model during the design phase. For each model we can install up to 4 units that can be operated separately. The catalogue of the available models is reported in tab.1. There are ten different available models of gas turbine and four models for each of the other component. It means that the design problem counts 88 potential machine units that might be installed to optimize the microgrid. It means that considering a time horizon of 14 days, the model will contain more than 29,500 binary variables only to model the on-off operation process of such units. The installation costs and the balance of plants associated to each technology are evaluated referring to [49]. We consider also a cost for the connections to the national electricity grid and to the gas network proportional to the maximum electric power and natural gas rate required by the system during the year. We assume a cost of the electricity connection equal to 50€/MW/y, while for the natural gas network 70€/MW/y (referred to the LHV of natural gas). The variables modelling the two connections introduce the need for introducing continuous design variables. The cost of the single MWh of electricity and natural gas we referred to historical real market prices, hence also considering typical hourly variations during the entire year. The objective function is defined minimizing the annuity associated to the microgrid. CAPEX are distributed as yearly costs assuming a life-time of the system of 15 years and a discount rate of 5%.

Observing the model reported in Appendix A, it is worth saying that, despite the relative simplicity of the Bovisa microgrid architecture, the accurate design model must include continuous design variable for modelling the maximum electric power and fuel consumption rate required to the external networks. In a hierachical decomposition method continuous design variables act as coupling variables between the UC representative periods, hindering the implementation of multiple worker problems, as done by Yokoyama et al. in [6]. This shows that, even in relatively simple MES, the need of a universal MES decomposition method is crucial to implement an efficient search algorithm without giving up modelling accuracy.

The design problem of the microgrid under analysis is solved in three different cases, gradually increasing the length of the time horizon to test the performances of the decomposition algorithm. The profiles for the demands of electricity, heating and cooling are always generated clustering the yearly data directly collected on field in a previous year. The hourly profile resolutions for the representative days are generated by means of a traditional k-means algorithm. The mathematical model has the structure of a typical catalogue based design problem and it is reported in Appendix A.

4.2 Analysis of the numerical performances

For each case, we also solved the problem by means of a conventional B&B algorithm as a benchmark. All the solver parameters were left untouched to the default values automatically set by CPLEX. The only exception concerns branching priority orders: a higher branching priority was given to integer and binary design variables in order to fairly compare the two solution approaches. In fact, a conventional CPLEX instance without branching priority orders would not exploit at all the hierarchical relationship between design and operation variables, hence performing worse than what possible for a MES design problem. For all the simulations we used a 4-core 2.9 GHz personal computer with a 16Gb RAM. The main purpose of our test is twofold:

1. Assessing the effectiveness of the integer cuts implemented both at the upper and at the lower level by the decomposition algorithm. This is necessary to understand if the global and local auxiliary problems have been properly defined and if they provide useful information to speed up the B&B

Gas Turbine										
Nominal Power [MW]	0.2	0.25	0.33	1	1.7	1.876	3.515	3.98	4.6	5.38
Heating Rate [MW]	0.295	0.376	0.45	1.299	3.233	3.945	8.92	8.80	9.56	10.91
Efficiency [-]	0.295	0.289	0.311	0.295	0.269	0.247	0.279	0.297	0.293	0.323
Cost [€/kW]	2001	1765	1716	1628	912	880	853	829	804	827
Auxiliary Boiler										
Nominal Power [MW]	0.7		1		1.4		2			
Efficiency [-]	0.92		0.92		0.92		0.92			
Cost [€/kW]	19.43		16.9		15.29		14.35			
Electric Chiller										
Nominal Power [MW]	0.564		0.704		0,844		1,056			
COP [-]	5		5		5		5			
Cost [€/kW]	115		115		115		115			
Absorption Chiller										
Nominal Power [MW]	0,692		1,036		1,382		1,728			
COP [-]	1.2		1.2		1.2		1.2			
Cost [€/kW]	240		240		240		240			

Table 1. Catalogue of the available models for the Bovisa case study

tree exploration;

2. Assessing the computational performance of the novel decomposition algorithm referring to the conventional B&B algorithm enforced with branching priorities as benchmark. In particular, we want to analyse the behaviour of the algorithm with increasing length of the time of horizon and so the complexity of the problem, when the conventional algorithm struggles for convergence.

To this aim, the design problem has been solved in three different cases, gradually extending the time horizon for the unit commitment from 3 days, to 7 and finally 14 days. The three design solutions are reported in tables 2, 3, 4. As we can notice, all the technologies have been installed, in general with a high number of units. Analysing the ED solution reported in fig.6 for three representative days, it is clear that the location is characterised by a strong weather seasonality and load variability (long academic vacations, weekends, etc.) that results in very different profiles and so different operation strategy depending on the considered typical period. The optimization procedure tends to select smaller units that cannot take advantage of economy of scale, but that can ensure to the system a high operation flexibility. This leads also to the fact that depending on the number of typical days introduced in the model and so on the representativity of the UC simulation, the design solution may be significantly different, resulting in an increase of the objective function of around 5% going from 3 to 14 typical days.

	Design Solution			
Selected Technology	GT3	AB1	EC4	AC1
Number of Units	4	1	3	3
Maximum El Power	2.95 MW			
Maximum NG Request	4.97 MW			
Objective Function	2,642,976 €/y			

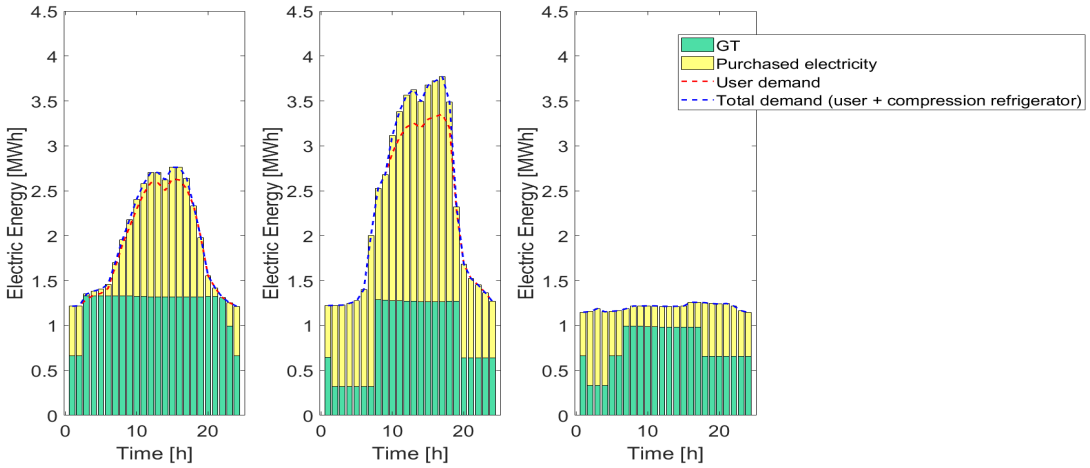
Table 2. Design solution of the 3-day design problem

	Design Solution			
Selected Technology	GT3	AB1	EC4	AC2
Number of Units	4	3	4	3
Maximum El Power	3.58 MW			
Maximum NG Request	5.64 MW			
Objective Function	2,716,181 €/y			

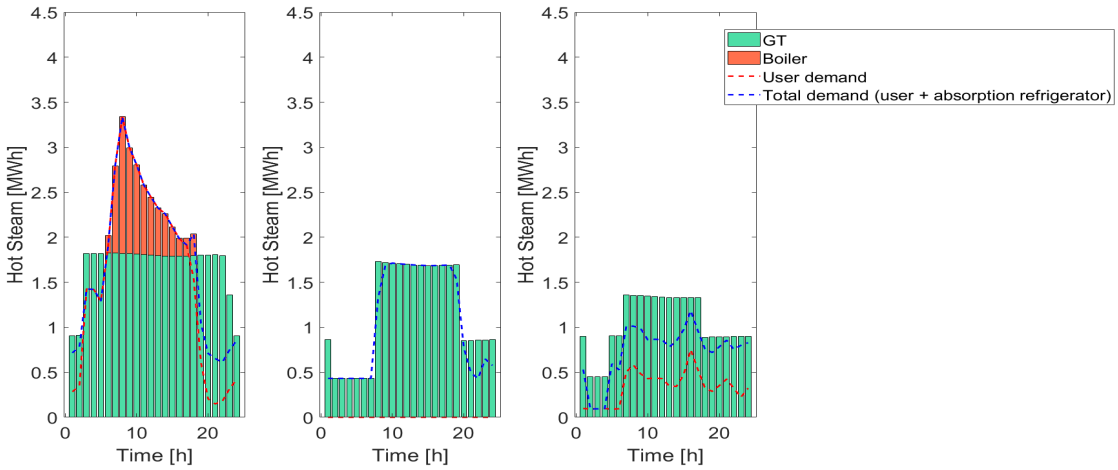
Table 3. Design solution of the 7-day design problem

	Design Solution			
Selected Technology	GT3	AB2	EC4	AC4
Number of Units	4	4	4	3
Maximum El Power	4.35 MW			
Maximum NG Request	6.33 MWh			
Objective Function	2,799,185 €/y			

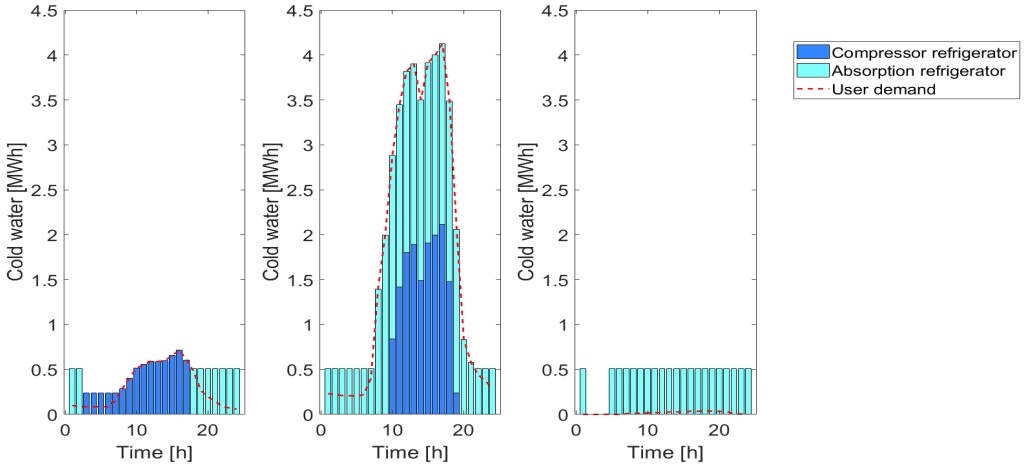
Table 4. Design solution of the 14-day design problem



(a) *Electricity UC of three out of the 14 typical days of the problem*



(b) *Heating UC of three out of the 14 typical days of the problem*



(c) *Cooling UC of three out of the 14 typical days of the problem*

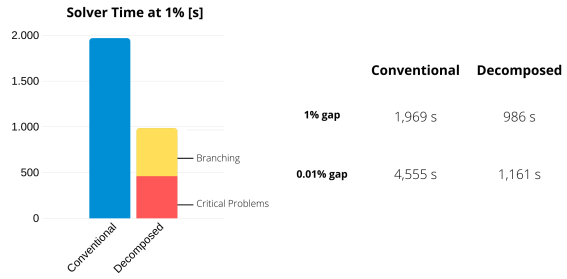
Figure 6. Economic dispatch for three representative days of the 14-day design problem

The really interesting results are extracted through the analysis of the computational performances. As reported in fig. 7a, the decomposition algorithm is competitive also in a limited time horizon of 3 days. The solution time is halved with respect to the traditional algorithm. Of course, this result seems not extremely remarkable, since the traditional algorithm takes a more than acceptable amount of time to get to 1% gap (around 2,000s). Yet, it is clear that with such microgrid architecture a time horizon of 3 days (72 operation hours) can be easily optimized by a traditional branch and bound algorithm, that can also take advantage of CPLEX advanced features like dynamic search and root node processing.

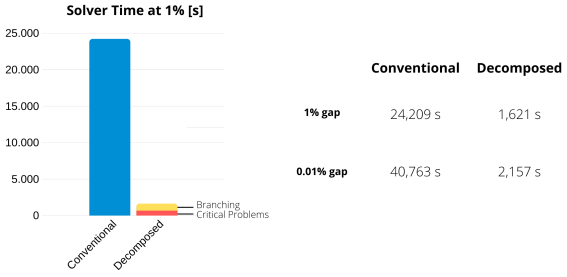
As expected, moving from 3 to 7 typical days, the complexity of the problem has exploded and the search tree for the conventional B%B algorithm has grown exponentially. CPLEX with default settings and branching priorities takes more than 24,200 s (6 hours and 40 minutes) to reach a relative gap of 1%, while the decomposition algorithm achieves the same gap in around 1,621 s (27 minutes), outperforming the conventional one and being 15 times faster (see fig. 7b). It is also interesting to highlight that the decomposition algorithm requires only additional 500 seconds to reach full convergence, while the conventional one still needs more than 20,000 s. A similar phenomenon occurs also in the 3-day design problem (see fig. 7a). Generally, the decomposition algorithm at 1% gap has already gathered a lot of information from the solution of worker and auxiliary problems. Integer cuts start being very effective and the full converge is achieved quite quickly.

The 14-day design problem is the one really demonstrating the effectiveness of the decomposition algorithm. As reported in fig. 7c, when increasing the time horizon from 7 to 14 representative days, the conventional B&B algorithm does not converge. After 50,000s the relative gap is still below 5% and the search procedure is stuck due to the enormous dimensions of the search tree. Introducing additional information that with a relatively small effort lead to prune extensive part of the tree at the upper level and to limit the exploration in the lower level is the only way to obtain the global optimum solution in a reasonable amount of time. The decomposition algorithm achieve 1% relative gap in less than 7,200 s (2 hours).

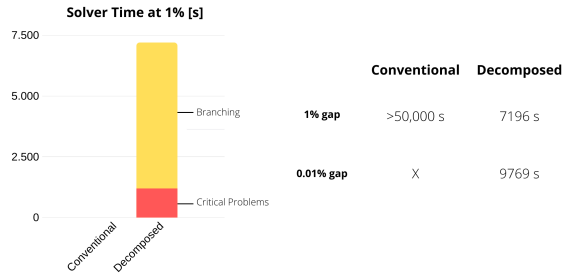
Such promising performances are confirmed by the analysis of the effectiveness of the introduced integer cuts of the decomposition algorithm. As extensively discussed, the role of local auxiliary problems is to evaluate lower bounds of the global objective function that allows to discard the corresponding integer design candidate before solving the complete worker problem in the lower



(a) *Computational performances of the novel decomposition algorithm in the 3-day design problem*



(b) *Computational performances of the novel decomposition algorithm in the 7-day design problem*



(c) *Computational performances of the novel decomposition algorithm in the 14-day design problem*

Figure 7. Computational performance comparisons between decomposed and conventional algorithms

level. Looking at fig. 8 referred to the 14-day design problem, we can state that the results are extremely promising. Considering 78 design candidates entering the lower level, only 7 complete worker problems are solved, while 71 are prematurely discarded thanks to local auxiliary problems (more than 90%). In addition to that, we can see that over 1136 potential local auxiliary problems, only 698 (around 60%) are actually solved in the lower level. This means that in general is sufficient to solve the auxiliary problems referred to 8 of the 14 typical periods to find the design candidate is infeasible or suboptimal. This percentage is higher during the first branching operations and tends to decrease as the number of design candidates entering the lower level increases. The reordering of the local auxiliary problems allows the algorithm to gradually learn which are the typical periods that favour infeasibility or suboptimality. As a result, the decomposition algorithm spends only 34.5% in the lower level, meaning the computational burden introduced by the solution of the auxiliary problems is acceptable if compared to the branching operations of the upper level. Finally, the results are convincing also for what global auxiliary problems are concerned. In spite they require around 15% of the overall solution time to be solved, they contribute to discard 342 design candidates even before entering the lower level (in addition to improving the effectiveness of the lower bounds in the lower level).

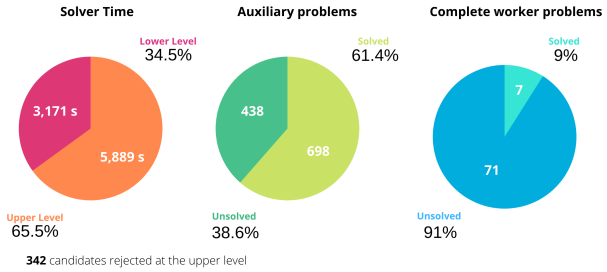


Figure 8. Analysis of the effectiveness of the introduced auxiliary problems and cuts in the 14-day design problem

The results obtained in the three cases are summarized in fig. 9. The presented graph finally enforces the theoretical discussion with numerical evidence and highlights the potentialities of the decomposition algorithm developed in this work. Conventional B&B shows the typical exponential growth in computational time when increasing the time horizon of the design problem, such that in the 14-day design problem is not able to converge to the optimal solution in an acceptable amount of time. Introducing the concept of auxiliary problems and exploiting the multi-period nature of MES design problems, the decomposition algorithm is able to effectively deal with an increasing number of integer operation variables, thanks to a more intelligent exploration of the lower level. Increasing the time horizon from 3 days to 14 days to obtain a more robust design solution, the algorithm moves from being the most efficient way to evaluate the optimum solution (15 times faster the conventional one) to being the only possible way to find the optimum design candidate in practical times.

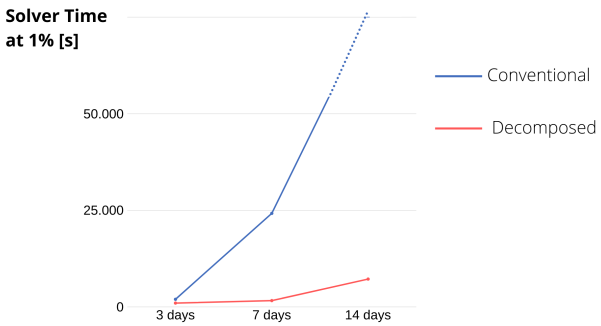


Figure 9. Computational performance comparison for different lengths of the time horizon

Conclusions

In this Thesis work, starting from the legacy of Iyer and Grossman [5] and Yokoyama [6], we introduced a novel MILP decomposition method exploiting the inherent hierarchy beween design and operation in MES design problems and we demonstrated its effectiveness

with respect to conventional approaches by means of a case study involving the design of a multi-energy microgrid for a university campus.

The main innovative elements of our proposed approach are both mathematical and numerical. From a mathematical standpoint, our method does not introduce any restricting assumption on the MES problem structure with respect to previous decomposition algorithms, hence ensuring the **universality** of the approach without sacrificing computational **performance**. Moreover, from a numerical perspective, an effort was made to guarantee a high **usability** of the decomposition algorithm for practical MES design purposes.

- **Universality**. Our method can be applied to MES problems not compatible with previous state-of-the-art decomposition paradigms. In particular, the MES model may contain both continuous and integer/binary design variables, as well as time-coupling constraints. This capability is particularly important in solving MES designs with complex architectures requiring a high modelling flexibility;

- **Performance**. Thanks to the introduction of the innovative concept of *local auxiliary problem* we were able to mantain a high generality of the method without giving up computational performances. This was demonstrated through a real case study in which the novel approach has been compared to the conventional MILP search strategy implemented in the commercial solver IBM CPLEX. Gradually increasing the time horizon and so the complexity of the design problem, conventional B&B showed an exponential growth in convergence time, while the decomposition algorithm demonstrated is capability of taking advantage of the multi-period nature of the problem to reduce complexity;

- **Usability**. The practical usability of the developed decomposition method is a feature indipendent from the previous two. Nonetheless, it is not less important. In fact, the implementation of MILP decomposition requires an in-depth knowledge of commercial solver API and of the corresponding programming language. Yet, MES desing is a field of interest for academics and professionals that do not necessarly have advanced programming skills. For this reason, we developed a dedicated code which automates the implementation of the decomposition algorithm in the POLIMIP, the MATLAB modelling environment developed in this Thesis work. Usability of the algorithm is crucial also beacause it favours future applications to new case studies and further developments.

Finally, we want to stress that the formal description of the novel decomposition method proposed in this work has been carried out mantaining a general profile that laid the ground for an actual *decomposition framework for Multi-Energy Systems*. In fact, the high-level concepts of "auxiliary problem" and "constraining feature" can be intended as mathematical tools that can be further expanded and/or modified to create new decomposition methods for specific types of MES problems. Some examples of possible future developments are:

- Assessment of the performances of the algorithm when applied to different MES architectures and operation models;

- Implementation of an heuristic operation logic in the lower level (instead of a predictive one);

- Application of the decomposition framework to a multi-year design problem for optimal investment planning considering yearly load variation scenarios.

Symbols appearing in chapter 3

f complete objective function;
g_t(or g_n) uncoupled constraints;
h coupling constraints;
z_D binary design variables;
y_D integer design variables;
x_D continuous design variables;
z_{O_t}(or z_{O_n}) binary operation variables;
y_{O_t}(or y_{O_n}) integer operation variables;
x_{O_t}(or x_{O_n}) continuous operation variables;
n_z number of binary design variables;
n_y number of integer design variables;
n_x number of continuous design variables;
m_z number of binary operation variables;
m_y number of integer operation variables;
m_x number of continuous operation variables;
T number of time steps in a typical period;
N number of typical periods;
f_D term of the objective function depending on the integer/binary design variables;
f_{D^*} term of the objective function depending on the continuous design variables;
f_{O_t}(or f_{O_n}) term of the objective function depending on the operation variables of period t (or n);
$^-$ (superscript) value related to an integer design candidate;
* (superscript) value related to the LP relaxation at the current node;
$_\sim$ (subscript) lower bound;
$^\sim$ (superscript) upper bound;
$^\circ$ (superscript) value computed before solving any local auxiliary problem;
$\check{\underset{\sim}{f}}_D$ global lower bound for f_D;
$\check{\underset{\sim}{f}}_{D^*}$ global lower bound for f_{D^*};
$\check{\underset{\sim}{f}}_{O_n}$ global lower bound for f_{O_n};
$\bar{\underset{\sim}{f}}_D$ local lower bound for f_D;
$\bar{\underset{\sim}{f}}_{D^*}$ local lower bound for f_{D^*};
$\bar{\underset{\sim}{f}}_{O_n}$ local lower bound for f_{O_n};
$\bar{\underset{\sim}{f}}^{(k)}$ best local lower bound available after solving k local auxiliary problems;

Appendix A

Sets

$I := \{\text{GT, AB, EC, AC}\}$ — Available technologies
$J_i := \{\text{M}_{1\text{i}}, \text{M}_{2\text{i}}..\text{M}_{\text{Ni}}\}$ — Models in the catalogue for each technology $i \in I$
$U := \{1,2,3,4\}$ — Potential installable units for each model $j \in J_i$
$TYP := \{1,2,3,4....\text{N}_{\text{typ}}\}$ — Typical days
$T := \{1,2,3,4...24\}$ — Hourly time step of each period

Variables

Binary/Integer Design Variables (z_D/y_D)

$\hat{\gamma}_{i,j}$:= Selection variable for the model j of the technology i
$\forall j \in J_i, i \in I$

$\hat{\varphi}_{i,j,u}$:= Investment variable for the unit u of the j$^{\text{th}}$ model
$\forall u \in U, j \in J_i, i \in I$

Binary/Integer Operation Variables (z_O/y_O)

$\hat{z}_{i,j,u,n,t}$:= On-off variable for u$^{\text{th}}$ unit at each time step t of the typical period n
$\forall t \in T, n \in TYP, u \in U, j \in J_i, i \in I$

$\hat{\delta}_{i,j,u,n,t}$:= Startup variable for u$^{\text{th}}$ unit at each time step t of the typical period n
$\forall t \in T, n \in TYP, u \in U, j \in J_i, i \in I$

Continuous Design Variables (x_D)

$\hat{el}^{max}$:= Maximum power required to the national electric grid [MW]

$\hat{fu}^{max}$:= Maximum fuel consumption rate required to the natural gas network [MW]

Continuous Operation Variables (x_O)

$\hat{y}_{i,j,u,n,t}$:= Primary energy output of the u$^{\text{th}}$ unit at each time step t of the typical period n
$\forall t \in T, n \in TYP, u \in U, j \in J_i, i \in I$

$\hat{y}^2_{GT,j,u,n,t}$:= Secondary energy output of the u$^{\text{th}}$ GT unit at each time step t of the typical period n
$\forall t \in T, n \in TYP, u \in U, j \in J_{GT}$

$\hat{x}^{ty}_{i,j,u,n,t}$:= Energy input of the u$^{\text{th}}$ unit at each time step t of the typical period n
$\forall t \in T, n \in TYP, u \in U, j \in J_i, i \in I$

$\hat{x}_{i,j,u,n,t}$:= Energy input of the u$^{\text{th}}$ unit accounting for start-up consumptions
$\forall t \in T, n \in TYP, u \in U, j \in J_i, i \in I$

$\hat{out}_{i,n,t}$:= Total primary energy output from the i$^{\text{th}}$ technology
$\forall t \in T, n \in TYP, i \in I$

$\hat{out}^2_{GT,n,t}$:= Total secondary energy output from gas turbines
$\forall t \in T, n \in TYP$

$\hat{in}_{i,n,t}$:= Total energy input for the i$^{\text{th}}$ technology
$\forall t \in T, n \in TYP, u \in U, j \in J_i, i \in I$

$\hat{el}_{n,t}$:= Purchased electricity from the grid

Objective Function

$\hat{f}_D$:= Part of the objective function depending on integer design variables

$\hat{f}_{D*}$:= Part of the objective function depending on continuous design variables

$\hat{f}_{O_n}$:= Part of the objective function depending on operation variables of the typical period n
$\forall n \in TYP$

Parameters

Energy demand and weather prediction

$d^{el}_{n,t} :=$ Electricity demand $\quad \forall t \in T, n \in TYP$

$d^{ht}_{n,t} :=$ Heating demand $\quad \forall t \in T, n \in TYP$

$d^{co}_{n,t} :=$ Cooling demand $\quad \forall t \in T, n \in TYP$

$\theta_{n,t} :=$ Temperature profile $\quad \forall t \in T, n \in TYP$

$H_n :=$ Yearly weight of the typical period n according to the clustering $\quad n \in TYP$

Machine operation characteristics

$m_{i,j} :=$ Performance slope $\quad \forall j \in J_i, i \in I$

$q_{i,j} :=$ Performance intercept $\quad \forall j \in J_i, i \in I$

$m^{T}_{i,j} :=$ Performance temprerature dependence $\quad \forall j \in J_i, i \in I$

$m^{2}_{GT,j} :=$ Performance slope for the GT seconday output $\quad \forall j \in J_{GT}$

$q^{2}_{GT,j} :=$ Performance intercept for the GT seconday output $\quad \forall j \in J_{GT}$

$m^{2T}_{GT,j} :=$ Performance temperature dependence for the GT seconday output $\quad \forall j \in J_{GT}$

$x^{l}_{i,j} :=$ Minimum input $\quad \forall j \in J_i, i \in I$

$x^{u}_{i,j} :=$ Maximum input $\quad \forall j \in J_i, i \in I$

$y^{u}_{i,j} :=$ Nominal output $\quad \forall j \in J_i, i \in I$

$ru_{i,j} :=$ Rump-up limit $\quad \forall j \in J_i, i \in I$

$rd_{i,j} :=$ Rump-down limit $\quad \forall j \in J_i, i \in I$

$su_{i,j} :=$ Start-up consumption $\quad \forall j \in J_i, i \in I$

Cost parameters

$IC_{i,j} :=$ Investment cost $\quad \forall j \in J_i, i \in I$

$CRF :=$ Capital Recovery Factor

$C^{el} :=$ Cost of connection to the national electric grid

$C^{fu} :=$ Cost of connection to the natura gas network

$c^{fu}_{i,n,t} :=$ Cost of the fuel per MWh $\quad \forall t \in T, n \in TYP, i \in FU$

$c^{el}_{n,t} :=$ Market price of the elctricit $\quad \forall t \in T, n \in TYP$

Constraints

Catalogue Selection and Investment Decision

$$\sum_i \hat{\gamma}_{i,j} \leq 1 \qquad \forall j \in J_i, i \in I$$

$$\hat{\varphi}_{i,j,1} = \hat{\gamma}_{i,j} \qquad \forall j \in J_i, i \in I$$

$$\sum_{n \in TYP} \sum_{t \in T} \hat{z}_{i,j,u,n,t} \leq \varphi_{i,j,u} \cdot \mathbb{N}_{ty} \cdot 24 \qquad \forall u \in U, j \in J_i, i \in I$$

Prioritization Constraints

$$\hat{\varphi}_{i,j,u} \leq \hat{\varphi}_{i,j,u-1} \qquad \forall u \in U \setminus \{1\}, j \in J_i, i \in I$$

$$\hat{z}_{i,j,u,n,t} \leq \hat{z}_{i,j,u-1,n,t} \qquad \forall t, \in T, n \in TYP, u \in U \setminus \{1\}, j \in J_i, i \in I$$

Input-output relationships

$$\hat{y}_{i,j,u,n,t} = m_{i,j}\hat{x}^{ty}_{i,j,u,n,t} + (q_{i,j} + m^T_{i,j}\theta_{n,t})\hat{z}_{i,j,u,n,t} \qquad \forall t \in T, n \in TYP, u \in U, j \in J_i, i \in I$$

$$x^l_{i,j}\hat{z}_{i,j,u,n,t} \leq x^{ty}_{i,j,u,n,t} \leq x^u_{i,j}\hat{z}_{i,j,u,n,t} \qquad \forall t \in T, n \in TYP, u \in U, j \in J_i, i \in I$$

$$\hat{out}_{i,n,t} = \sum_{j \in J_i}\sum_{u \in U} \hat{y}_{i,j,u,n,t} \qquad \forall t \in T, n \in TYP, i \in I$$

$$\hat{in}_{i,n,t} = \sum_{j \in J_i}\sum_{u \in U} \hat{x}_{i,j,u,n,t} \qquad \forall t \in T, n \in TYP, i \in I$$

$$\hat{out}^2_{GT,n,t} = \sum_{j \in J_i}\sum_{u \in U} \hat{y}^2_{GT,j,u,n,t} \qquad \forall t \in T, n \in TYP$$

Start-up, minimum up time and ramp limits

$$\hat{\delta}_{i,j,u,n,t} \geq \hat{z}_{i,j,u,n,t} - \hat{z}_{i,,ju,n,t-1} \qquad \forall t \in T, n \in TYP, u \in U \setminus \{1\}, j \in J_i, i \in I$$

$$\hat{x}_{i,j,u,n,t} = \hat{x}^{ty}_{i,j,u,n,t} + SU\hat{\delta}_{i,j,u,n,t} \qquad \forall t \in T, n \in TYP, u \in U \setminus \{1\}, j \in J_i, i \in I$$

$$\hat{y}_{i,j,u,n,t-1} - \hat{y}_{i,j,u,n,t} \leq ru_i + (1 - \hat{z}_{i,j,u,n,t})M \qquad \forall t \in T \setminus \{1\}, n \in TYP, u \in U, j \in J_i, i \in I$$

$$\hat{y}_{i,j,u,n,t} - \hat{y}_{i,j,u,n,t-1} \leq rd_i + (1 - \hat{z}_{i,j,u,n,t})M \qquad \forall t \in T \setminus \{1\}, n \in TYP, u \in U, j \in J_i, i \in I$$

$$\sum_{t-t^{min}+1}^{t} \hat{z}_{i,j,u,n,t} \geq t^{min}\hat{\delta}_{i,j,u,n,t-t^{min}+1} \qquad \forall t \in \{t^{min}, \ldots, T^{end}\}, n \in TYP, u \in U, j \in J_i, i \in I$$

Energy balances

$$\sum_{i \in OUT^{el}} \hat{out}_{i,n,t} - \sum_{i \in IN^{el}} \hat{in}_{i,n,t} \geq d^{el}_{n,t} - \hat{el}_{n,t} \qquad \forall t \in T, n \in TYP$$

$$\sum_{i \in OUT^{ht}} \hat{out}_{i,n,t} + \hat{out}^2_{GT,n,t} - \sum_{i \in IN^{ht}} \hat{in}_{i,n,t} \geq d^{ht}_{n,t} \qquad \forall t \in T, n \in TYP$$

$$\sum_{i \in OUT^{co}} \hat{out}_{i,n,t} - \sum_{i \in IN^{co}} \hat{in}_{i,n,t} \geq d^{co}_{n,t} \qquad \forall t \in T, n \in TYP$$

Maximum fuel consumpiton and power from the grid

$$\hat{el}_{n,t} \leq \hat{el}^{max} \qquad \forall t \in T, n \in TYP$$

$$\sum_{i \in FU} \hat{in}_{i,n,t} \leq fu^{max} \qquad \forall t \in T, n \in TYP$$

Objective function

$$\hat{f}_D = \sum_{i \in I}\sum_{j \in J_i}\sum_{u \in U} C_{i,j}\hat{\varphi}_{i,j,u}CRF$$

$$\hat{f}_{D*} = \hat{el}^{max}C^{el} + \hat{fu}^{max}C^{fu}$$

$$\hat{f}_{O_n} = \left(\sum_{i \in FU}\sum_{t \in T} \hat{in}_{i,n,t}c^{fu}_{n,t} + \sum_{t \in T} \hat{el}_{n,t}c^{el}_{n,t}\right)\frac{8760}{H_n} \qquad \forall n \in TYP$$

$$\min \hat{f} = \hat{f}_D + \hat{f}_{D*} + \sum_{n \in TYP} \hat{f}_{O_n}$$

References

[1] *YALMIP*. URL: https://yalmip.github.io (cit. on pp. 1, 4).

[2] European Commission. *Tools and technologies for coordination and integration of the European energy system*. 2017. URL: https://ec.europa.eu/info/funding-tenders/opportunities/portal/screen/opportunities/topic-details/lce-05-2017 (cit. on p. 1).

[3] Peter Asmus. "Microgrids, Virtual Power Plants and Our Distributed Energy Future". In: *Electricity Journal* 23.10 (2010), pp. 72–82. ISSN: 10406190. DOI: 10.1016/j.tej.2010.11.001. URL: http://dx.doi.org/10.1016/j.tej.2010.11.001 (cit. on pp. 1, 2).

[4] George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization*. New York, NY, USA: Wiley-Interscience, 1988. ISBN: 0-471-82819-X (cit. on pp. 2, 3).

[5] R.R. Iyer and Ignacio Grossmann. "Synthesis and Operational Planning of Utility Systems for Multiperiod Operation". In: *Computers Chemical Engineering* 22 (July 1998), pp. 979–993. DOI: 10.1016/S0098-1354(97)00270-6 (cit. on pp. 2, 6, 7, 18).

[6] Ryohei Yokoyama et al. "Optimization of energy supply systems by MILP branch and bound method in consideration of hierarchical relationship between design and operation". In: *Energy Conversion and Management* 92 (2015), pp. 92–104. ISSN: 01968904. DOI: 10.1016/j.enconman.2014.12.020. URL: http://dx.doi.org/10.1016/j.enconman.2014.12.020 (cit. on pp. 2, 6–8, 12, 14, 18).

[7] IRENA International Renewable Energy Agency. *Innovation Outlook Mini-Grids*. 2016 (cit. on p. 2).

[8] L Moretti et al. "Assessing the impact of a two-layers predictive dispatch algorithm on design and operation of off-grid hybrid microgrids". In: ().

[9] H. K. Fathy et al. "On the coupling between the plant and controller optimization problems". In: *Proceedings of the 2001 American Control Conference. (Cat. No.01CH37148)*. Vol. 3. June 2001, 1864–1869 vol.3 (cit. on p. 3).

[10] Davide Fioriti et al. "Stochastic sizing of isolated rural mini-grids , including effects of fuel procurement and operational strategies". In: *Electric Power Systems Research* 160 (2018), pp. 419–428. ISSN: 0378-7796. DOI: 10.1016/j.epsr.2018.03.020. URL: https://doi.org/10.1016/j.epsr.2018.03.020 (cit. on p. 3).

[11] J. Kennedy and R. Eberhart. "Particle swarm optimization". In: *Proceedings of ICNN'95 - International Conference on Neural Networks*. Vol. 4. Nov. 1995, 1942–1948 vol.4. DOI: 10.1109/ICNN.1995.488968 (cit. on p. 3).

[12] Rémy Rigo-mariani et al. "Comparison of optimization frameworks for the design of a multi-energy microgrid". In: *Applied Energy* 257.July 2019 (2020), p. 113982. ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2019.113982. URL: https://doi.org/10.1016/j.apenergy.2019.113982 (cit. on p. 3).

[13] Davide Fioriti et al. "Comparison among deterministic methods to design rural mini-grids : effect of operating strategies". In: *2019 IEEE Milan PowerTech* (), pp. 1–6 (cit. on p. 3).

[14] Bei Li, Robin Roche, and Abdellatif Miraoui. "Microgrid sizing with combined evolutionary algorithm and MILP unit commitment". In: *Applied Energy* 188 (2017), pp. 547–562. ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2016.12.038. URL: http://dx.doi.org/10.1016/j.apenergy.2016.12.038 (cit. on p. 3).

[15] Julia Sachs and Oliver Sawodny. "Multi-objective three stage design optimization for island microgrids". In: *Applied Energy* 165 (2016), pp. 789–800. ISSN: 03062619. DOI: 10.1016/j.apenergy.2015.12.059. URL: http://dx.doi.org/10.1016/j.apenergy.2015.12.059.

[16] K. Heidenberger. "Dynamic project selection and funding under risk: A decision tree based MILP approach". In: *European Journal of Operational Research* 95.2 (1996). cited By 33, pp. 284–298. DOI: 10.1016/0377-2217(95)00259-6 (cit. on p. 3).

[17] T. Öncan and M. Cağirici. "MILP formulations for the order batching problem in low-level picker-to-part warehouse systems". In: *IFAC Proceedings Volumes (IFAC-PapersOnline)* 46.9 (2013). cited By 1, pp. 471–476. DOI: 10.3182/20130619-3-RU-3018.00372 (cit. on p. 3).

[18] S. Belil, S. Kemmoé-Tchomté, and N. Tchernev. "MILP-based approach to mid-term production planning of batch manufacturing environment producing bulk products". In: *IFAC-PapersOnLine* 51.11 (2018). cited By 0, pp. 1689–1694. DOI: 10.1016/j.ifacol.2018.08.213 (cit. on p. 3).

[19] C.A. Méndez et al. "State-of-the-art review of optimization methods for short-term scheduling of batch processes". In: *Computers and Chemical Engineering* 30.6-7 (2006). cited By 581, pp. 913–946. DOI: 10.1016/j.compchemeng.2006.02.008 (cit. on p. 3).

[20] M.G. Earl and R. D'Andrea. "Iterative MILP methods for vehicle-control problems". In: *IEEE Transactions on Robotics* 21.6 (2005). cited By 98, pp. 1158–1167. DOI: 10.1109/TRO.2005.853499 (cit. on p. 3).

[21] A. Bischi et al. "A rolling-horizon optimization algorithm for the long term operational scheduling of cogeneration systems". In: *Energy* 184 (2019). cited By 8, pp. 73–90. DOI: 10.1016/j.energy.2017.12.022. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85042304353&doi=10.1016%2fj.energy.2017.12.022&partnerID=40&md5=eb41d74c817192bdb2d87b8a8186800d (cit. on p. 3).

[22] A. H. Land and A. G. Doig. "An Automatic Method of Solving Discrete Programming Problems". In: *Econometrica* 28.3 (1960), pp. 497–520. ISSN: 00129682, 14680262. URL: http://www.jstor.org/stable/1910129 (cit. on p. 4).

[23] *Mixed-Integer Programming (MIP) – A Primer on the Basics*. URL: https://www.gurobi.com/resource/mip-basics/ (cit. on p. 4).

[24] *GAMS modeling software*. URL: https://www.gams.com/ (cit. on p. 4).

[25] *AMPL modeling software*. URL: https://ampl.com/ (cit. on p. 4).

[26] *PYOMO*. URL: http://www.pyomo.org (cit. on p. 4).

[27] *JuMP*. URL: https://github.com/JuliaOpt/JuMP.jl (cit. on p. 4).

[28] *PuLP*. URL: https://pythonhosted.org/PuLP (cit. on p. 4).

[29] *ZIMPL*. URL: https://zimpl.zib.de (cit. on p. 4).

[30] *CVX*. URL: http://cvxr.com/cvx/ (cit. on p. 4).

[31] *CVXPY*. URL: https://github.com/cvxgrp/cvxpy (cit. on p. 4).

[32] *TOMLAB*. URL: http://tomopt.com/tomlab/ (cit. on p. 4).

[33] *CPLEX*. URL: https://www.ibm.com/analytics/cplex-optimizer (cit. on p. 4).

[34] *Gurobi*. URL: https://www.gurobi.com (cit. on p. 4).

[35] *BARON*. URL: https://minlp.com/baron (cit. on p. 4).

[36] *Mosek*. URL: https://www.mosek.com/ (cit. on p. 4).

[37] *LINDO*. URL: https://www.lindo.com (cit. on p. 4).

[38] *GLPK*. URL: https://www.gnu.org/software/glpk/ (cit. on p. 4).

[39] *LPsolver*. URL: http://lpsolve.sourceforge.net/5.5/ (cit. on p. 4).

[40] *BLIS*. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.410.8877&rep=rep1&type=pdf (cit. on p. 4).

[41] *CBC*. URL: https://github.com/coin-or/Cbc (cit. on p. 4).

[42] *MINTO*. URL: https://neos-server.org/neos/solvers/milp:MINTO/AMPL.html (cit. on p. 4).

[43] *SCIP*. URL: https://scip.zib.de (cit. on p. 4).

[44] *SYMPHONY*. URL: https://www.coin-or.org/SYMPHONY/index.htm (cit. on p. 4).

[45] *ndSparse class*. URL: https://www.mathworks.com/matlabcentral/fileexchange/29832-n-dimensional-sparse-arrays#feedbacks (cit. on p. 4).

[46] Ed Klotz and Alexandra Newman. "Practical guidelines for solving difficult mixed integer linear programs". In: *Surveys in Operations Research and Management Science* 18 (Oct. 2013), pp. 18–32. DOI: 10.1016/j.sorms.2012.12.001 (cit. on p. 5).

[47] Ted Ralphs and Matthew Galati. "Decomposition Methods for Integer Programming". In: Feb. 2011. DOI: 10.1002/9780470400531.eorms0233 (cit. on p. 5).

[48] *Generic callbacks*. URL: https://www.ibm.com/support/knowledgecenter/de/SSSA5P_12.8.0/ilog.odms.cplex.help/CPLEX/UsrMan/topics/progr_adv/callbacks/introCallbacks.html (cit. on p. 12).

[49] Davis Langdon Engineering Services, ed. *Spoon's Mechanical and Electrical Services Price Book*. Spoon Press, 2011.

Chapter 1

Multi-Energy Systems (MES): an overview

1.1 MES design: a complex task

> *"The increasing share of variable renewable energy sources and the 2020 and 2030 targets for the reduction of greenhouse gas emission in the EU are calling for important changes in our energy system: more flexibility, more active involvement of all stakeholders and more collaboration. If no actions are taken, the power system will face several risks such as poor quality of the electricity supply, congestion, lack of stability, excessive levels or curtailments, impossibility to cope with electro mobility demand, etc. The challenge is therefore to create and deploy common tools for planning, integration and operation across the energy system and its actors."* [1]
>
> European Commission

From researchers to policy makers, from companies to international organizations, many efforts are being made nowadays to encourage the penetration of renewable energy sources into existing energy-mixes and to favour the emergence of new electrification plans already geared towards sustainability in countries that are currently developing their own power systems [4]. Nevertheless, meeting **challenging environmental targets** and **guaranteeing secure and affordable energy** to

future generations are two equally important missions of the energy sector that may risk hindering each other.

The quotation at the beginning of this chapter is taken from the H2020 call "Tools and technologies for coordination and integration of the European energy system". European commission has inevitably come to admit that, while new energy generation technologies are continuously evolving, our energy distribution system is still tied to obsolete paradigms. Energy resources are becoming more and more unpredictable. Energy services are increasing and require more flexibility. In such a context, the way we intend and design the energy system cannot do without two key factors: **coordination** and **integration**. The "classic" energy system where generation units, as well as loads, networks and energy vectors are treated "separately" is giving way to a model whereby electricity, heat, cooling, fuels, and transport optimally interact with each other at various levels, from buildings to districts, cities or regions. The capability of properly designing and operating so-called **Multi-Energy Systems** (MES) is decisive to fully exploit the potential of new technologies that are now trying to emerge in the energy sector.

1.1.1 MES: their role and applications in the energy sector

A comprehensive definition of MES can be found in [2], where the fundamental concept of multi-energy system is summed up as follows:

> *"..an integrated energy system consisting of distributed energy resources and multiple energy loads operating as a single, autonomous grid either in parallel to or "islanded" from the existing utility grid"*

MES are often associated to the innovative concept of Smart Grid and the idea of optimally integrating distributed renewable electricity resources in the national power system by means of electric storage and load demand aggregation. Nonetheless, this only covers a part of the MES framework. Multi-energy systems try to practice integration strategies addressing **all energy sectors**, and not only

electricity, from both the operational and the design viewpoints [3]. Nowadays, generation and consumption of the various energy vectors are strictly interconnected. For instance, heating and cooling continuously interact with the electric system in many distributed technologies such as CHP (combined heat and power), electric heat pumps, air condition and refrigeration systems. Electricity is intensifying its presence also in transport and fuel-chain, with the emergence of bio-fuels, electric vehicles and hydrogen-based transports [5].

The cited "integration" of MES occurs at three different levels:

Multi-service A multi-energy system is generally capable of generating different useful outputs and multiple services to the external market. This comes with the possibility of turning otherwise wasted conversion by-products into useful output, improving both techno-economic and environmental performances. This is one of the reason why many MES are based on multi-generation units, such as CHP and CCHP (Combined Cooling, Heat and Power) systems. A co-generative engine for heating and power of a building may represent the smallest "block" in a MES. Of course, the generation of many energy vectors requires a connection with many different external energy networks, such as electricity, DH (district heating), gas, hydrogen, etc [7].

Multi-input The many interactions with the external market are not only limited to the multiple outputs, but also and primarily to the inputs. Apart from the more classic connections with electricity and gas networks, MES may receive also "unconventional" goods as input, such as waste to be converted into bio-fuel to produce heat and electricity [8], or water to feed an electrolizer for hydrogen generation. Such "goods" are seen from the MES perspective as energy carriers, to be converted into useful outputs for the market. MES possible inputs include also renewable energy sources (RES), such as solar radiation or kinetic wind power. RES have a great potential both from the economic and environmental perspective, but they bring about the challenge of being non-programmable and difficult to predict.

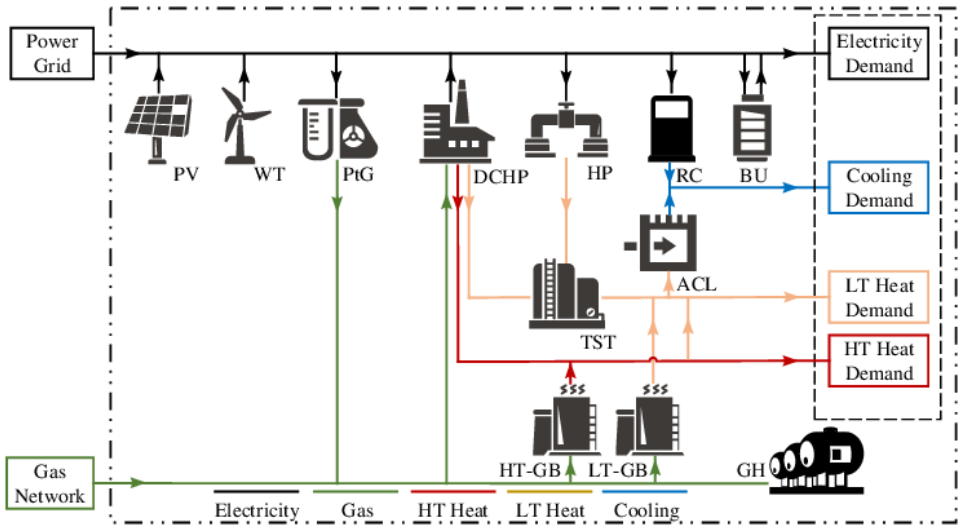


Figure 1.1: Typical architecture of a multi-energy microgrid

Multi-process The last level of MES interconnectivity is within the processes. MES are not simply systems of energy conversion, with a classic generation-consumption balance, but may include different processes, such as transport and storage. As electric and thermal storage technologies are becoming more competitive [9], the inclusion of energy storage systems in MES design offers many economic advantages and allows to better exploit non-programmable sources and, at the same time, to offer a higher flexibility of the operation.

The MES definition and framework can be related to many different innovative systems in the energy sector, from virtual power plants [10] to smart grids [11]. However, the most concrete application is surely in the field of **microgrids**. Microgrids are low or medium voltage distribution systems controlling and coordinating many distributed energy resources, mainly distributed generation, storage and controllable loads [7]. Although initially designed for the production of electricity, today multi-energy microgrids (see fig.1.1) are becoming the most promising model of distributed generation, for both on-grid and off-grid applications [9]. The main advantages are related to:

- Optimal utilization of primary energy sources for multiple services and, consequentially, **high conversion efficiency**;

- High energy system **flexibility** and optimal market interaction;

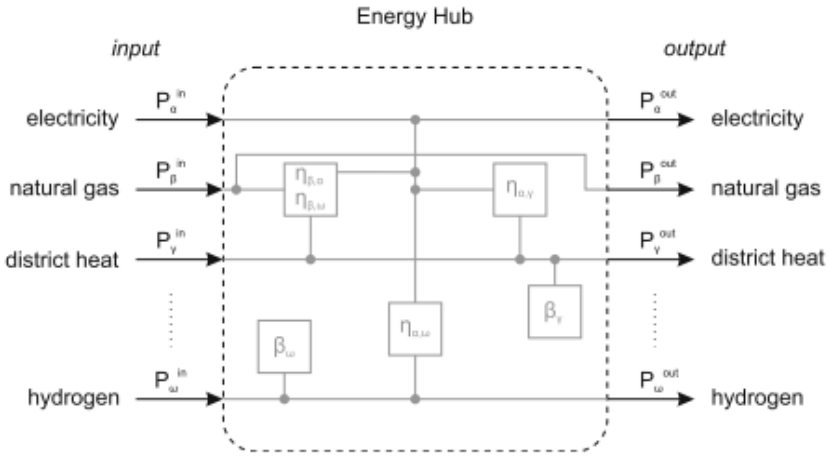


Figure 1.2: Example of a energy hub containing converters η and storage β [14]

- Potentially high **penetration of RES** without compromising the quality and the availability of the offered services.

All these benefits come at non-negligible costs: **increased complexity** and **high unpredictability**. At this point, a spontaneous question arises: *how is it possible to properly design and operate these systems in order to effectively exploit all their potentialities?*

1.1.2 Mathematical modelling of MES: main challenges

When speaking of MES, design phase and operation phase should always be considered together, since strongly interdependent. This unavoidable interdependence actually represents one of the main inherent complexities in developing mathematical models that allow to effectively design these kind of systems.

When designing a simple energy system, with single independent units, independent processes and one or two energy vectors to be converted, the solution space is quite limited. In this case, many design variables (e.g. the size of the components) can be limited to a narrow range or reasonable values defining some nominal operating conditions. Hence, when moving to the detailed techno-economic optimization through the simulation of the operation phase, the values of many variables has already been set. Moreover, optimal operation strategies are often trivial, based on

very simple trade-off, and produce very similar results regardless the simulation scenario.

On the other hand, in MES design all the units are interconnected and design choices regarding a single variable may dramatically change the optimal value of all the others. Multiple components, multiple processes, multiple scenarios. The solution space is so broad that even the most trivial design choice may result impossible to make without a detailed and comprehensive simulation of the operation phase of the system. Many internal interconnections result in many possible operation strategies that inevitably affect the optimality of the design solution. Many external interconnections force the behaviour of individual components and influence the behaviour of the entire system. Hence, a suitable model for the design of MES needs to incorporate an accurate and realistic mathematical description of its complex operation, resulting in a single huge problem.

MES operation modelling

In light of the above, the first step to model the design of a MES is to find a way to properly describe its operation. To this end, many researchers had the idea of providing synthetic information and optimization basis of complex systems through input-output relationships. They have set aside the traditional operation models used in electrical systems to give space to a new framework: the **energy hub** [13] [12]. The basics elements of an energy hub are energy conversion units, energy storage units and input-output connections [14] (see fig.1.2).

The energy hub concept is nowadays a common approach for steady state modelling and optimization of MES dispatch and can represent energy system operation at various scales and resolution, from optimal dispatch [15] to optimal power flow modelling, topological optimization and reliability assessment [16]. As a matter of fact, it is surely a way to schematically model an interconnected system made of many different components such as a MES. Nonetheless, a proper MES

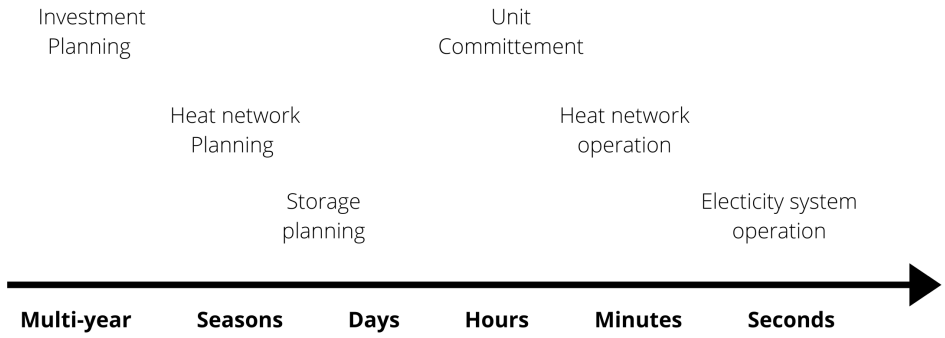


Figure 1.3: Temporal scales in MES

operation model should address many modelling issues that inevitably carry a high level of complexity.

Multiple time scales All MES dispatch models are steady-state model, assuming that fast phenomena have reached equilibrium in the time-step interval. Nonetheless, MES cover a wide range of time scales, from microseconds for the electric system operation to months for the seasonal influences of RES (see fig.1.3). The use of temporal-aggregated data allows to simplify the model but may lead to neglect fine-scale phenomena deviating the results [17]. Generally, during the design phase the aggregation is performed at the scale of **Unit Commitment** (UC) (from 1 hour to 15 minutes).

Non-linear relations In spite the fact that many MES relations can be expressed as linear equations (e.g. energy balances), the input-output behaviour of several units shows significant non-linearities. For instance, off-design performance curves of thermal generators like gas turbine are often represented through quadratic or cubic curves. Heat exchangers or chemical reactors conversion laws are strongly non-linear, too. Even without considering fast-dynamic phenomena, MES operation models need to deal with non-linear equations, or through the use of non-linear models (high complexity and accuracy), or through linearisation procedures (trade-off between complexity and accuracy).

Binary operation logics Proper models for UC require binary variables that represent the status of the machines at each time interval. In this way, the on-off logics of the energy hub unit operation are modelled, along with many dynamic phenomena of energy generators, such as start-up costs, minimum up time and ramp limits. Without binary status variables, the optimization algorithm generally proposes solution that are not realistic nor feasible. On the other hand, the integration of binary operation variables is typically computationally expensive.

Predictive control strategies For MES operation planning, the concept of Economic Dispatch (ED) is central. The ED is the optimization of the UC in order to cover the energy demands at a minimum cost or to maximize the energy production revenues, at the same time satisfying all the physical constraints [18]. Since ED requires short-term planning strategies to optimize the memory effects of MES components (e.g. energy storage), Model-based Predictive Control techniques are being applied, based on receding prediction horizon and feedback mechanisms [19]. Predictive control strategies result in further complications of the UC model, that needs to account for the stochastic nature of the problem, caused b y uncertainty over demand and RES prediction.

From operation to design modelling

As previously said, the MES design problem needs to embed the simulation of the operation to produce reliable results, so all the criticalities of dispatch modelling will be present during the design phase as well. In addition to that, we need to consider the challenges related to the design optimization itself.

Discreteness of components MES design models include many candidate components which may potentially integrated into the final architecture. The investment decision on each potential component is modelled through binary variables. Moreover, one of the crucial decision to make is the size of the components. The simplest way to model it is by means of a simple continuous variable. Yet, this may produce very inaccurate results. If we consider for instance a Gas Turbine (GT), the selection of the GT nominal power affects the performances and the consumptions, too. In

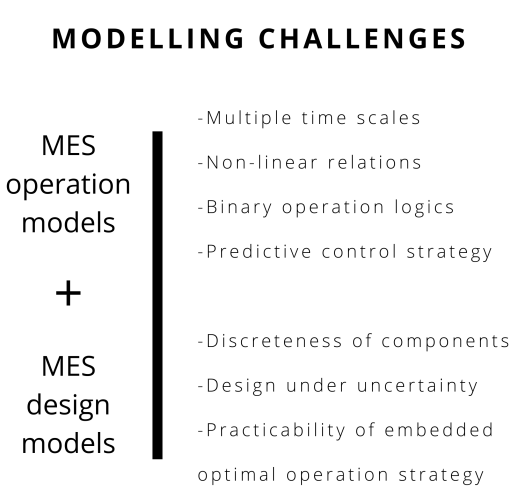


Figure 1.4: Main modelling challenges for MES design and operation

that case it is necessary to introduce binary variables that change the performance curve when a certain size threshold is exceeded. And the complexity is further increased in the event that the catalogue of the available unit model is limited and a continuous design is not representative. Catalogue-based design models require investment binary variables for each available model and for each architectural technology, or sometimes also for each potential unit of each available model of each architectural technology [20]. For this reason, the number of binary design variables is prone to increase very fast with the size of the components' catalogue.

Design under uncertainty MES design models need to simulate the operation phase of potential architecture configurations to evaluate their performance, and such simulations are based on predicted profiles of load demands and RES production. Of course, the quality of the solution strongly depends on the capability of predicting load behaviour and weather conditions to create representative profiles. This is why MES design models may include machine learning algorithm to increase prediction accuracy [21] as well as stochastic design algorithms to account for multiple possible scenarios [22].

Practicability of embedded optimal operation strategy Design models embed operation simulation that are generally performed by optimizing long-term

periods (weeks, months or even a full year) to exploit energy storage and dispatchable loads at best. In this way, the design model embeds a kind of omniscience of the control system that can never be realized in reality, and this could compromise the optimisation of the solution found. One of the greatest challenges of MES modelling lies in taking into account the control strategies implemented in field while assessing the operation performance of possible design solutions.

1.2 Main approaches to MES design optimization

The aim of this section is to present the main methodologies historically applied to solve MES design optimization problems and to highlight their main potentialities and limitations.

1.2.1 Design optimization methods: a review

As already mentioned in the previous sec. 1.1, the optimization of multi-energy architectures is a problem where the operation of the system is considered from the design phase. To this end, the optimization framework may be based on two different paradigms:

Two-layer (TL) The problem is solved by means on two nested algorithms. The outer loop iteratively generates potential design solution, while the inner loop optimize the UC and estimates the operating expenses (OPEX) for the corresponding system configuration.

One-shot (OS) One-shot optimization frameworks make use of a single model containing both design and operation variables. System sizing and dispatch are solved simultaneously in a single optimization problem.

It is possible to mathematically demonstrate that design and operation in MES are coupled such that when solve sequentially or iteratively the results are not guaranteed to constitute a combined system optimum [24]. For this reason, traditional two-layer algorithms cannot ensure the global optimality of the solution and this is

the unfair advantage of OS optimization. At the same time, accurate OS models tend to become very complex when applied to large multi-energy systems and may result in non-acceptable computational times. Beyond such general considerations, each optimization strategy is characterised by specific pros and cons, which is correct to know before approaching the resolution of a generic MES design problem. Historically, TL algorithms were the first to significantly attract the attention of researchers, showing important improvements over the year.

Montecarlo simulation [TL] In the outer loop Monte Carlo-based methods perform random generation of component sizes that are then investigated solving the ED problem in the inner loop. In order to increase the effectiveness of Monte Carlo exploration, it is possible to combine random simulation with analytical procedures as proposed in [25]. This method was one of the first to be proposed in the field of MES design, but to date it remains one of the least interesting. The main reason is that for advanced multi-energy architecture the design solution space is so broad that random research would take an extremely long time to explore it effectively. In [23], a case in which a Monte Carlo-based algorithm with Latin Hypercube Sampling consisting in 10,000 random samples takes 10 hours to get to results very similar to the ones obtained through evolutionary algorithms in less than 1 hour.

Partical Swarm Optimization [TL] Particle Swarm Optimization (PSO) is one of the most used evolutionary algorithms to explore the upper level of MES design problem. Evolutionary algorithms consist in metaheuristic procedures that imitate biological mechanism to select and generate the family of the solutions to investigate [26]. In spite being one of the first to be employed in this field, PSO is still one of the most promising alternative for TL optimization. When working with evolutionary algorithm, the main issue is to find a proper tuning of the parameters (number of particles, iterations, etc.) and to obtain the best trade-off between:

1. Controlling the maximum computational time for the optimal sizing;

2. Getting sufficiently close to the global optimum of the problem.

Even if tuning methodologies are present in literature [28], defining a generalized approach is still an open challenge.

Evolutionary algorithms are stochastic methods and their exploration of the solution space may vary significantly even starting from the same population. One of the main advantage of PSO is that it tends to guarantee a good exploration of the solution space without going too far from the global optimum point [23], [29]. The price is a relatively long convergence time, which inevitably results in a very large number of operation simulation. If the operation problem is complex and requires more than few tenths of a second to be solved (e.g. high number of units, sophisticated rolling-horizon control strategies) the convergence time will probably result unacceptable.

Genetic Algorithms [TL] Genetic Algorithms (GA) have been recently introduced as innovative solution space exploration method for two-layer MES design optimization. GA are evolutionary algorithms as PSO, but recently they have attracted a lot of attention thanks to their ability to quickly improve the objective function, achieving convergence in a very short time [30]. Nonetheless, GA algorithm are often characterised by very undesired behaviour: after reaching an interesting area of the solution space very quickly, they get stuck on that without guaranteeing a good exploration of the problem. As a consequence, the proposed solution may very a lot from run to run (even starting from the same population) and is often far from the global optimum of the problem [23]. This is why many researchers are still sceptical about their actual usefulness.

TL methods have for many years been the only way to achieve sufficiently robust results for the design of multi-energy systems, in particular in the field of multi-energy system. One of the main reason is that, since in TL algorithms design solution search and UC optimization are de-coupled, once a design solution is found it is possible to directly simulate the operation strategy that will be actually implemented on field [31]. This allows to obtain a good coherence between the

simulation results and the actual performance on field.

OS algorithm are based on models where design variables are related to operation variables and simultaneously optimized in a single problem. The optimization solver sees the whole simulation period and finds the best operation strategy as if it was possible to precisely predict the load behaviours and the RES production for the entire time horizon of the optimization. This is generally called *predictive design* and is the reason why OS approaches suffer for the modelling issue reported in sec. 1.1 as "Practicability of embedded optimal operation strategy". Nevertheless, in recent years predictive design has become a need more than a burden, since it is the only way properly design systems with long-term memory effects, such as seasonal energy storage. Different strategies have been studied to deal with the stochastic nature of the problem [32] and sophisticated control algorithm have been developed to integrate the optimal operation strategy obtained through predictive solutions in on-field Energy Management Systems (EMS) [20]. While many TL methods are implementing predictive optimization in the inner loop as well, OS methods are attracting more and more attention, in particular for their capability of always ensuring the globality of the optimum they find.

Linear Programming [OS] Linear Programming (LP) is surely the simplest way to perform a one-shot design optimization of a MES [33]. Unfortunately, it only admits purely linear equations containing continuous variables in the model. The impossibility to model integer and binary variables does not allow to describe in detail both the operation dynamics of the system and the discreteness of the design catalogue. Moreover, also non-linear input-output relations need to be modelled with very simplified linear fitting. LP cannot be applied to solve detailed design problems, with broad catalogues of hourly simulation of unit commitment. Yet, the main advantage is surely related to the extremely low computational time required for the optimization. They are mainly applied to obtain qualitative information or to solve optimization problem for which all the other OS methods would require an excessive amount of time (e.g. multi-year design optimization).

Mixed-Integer Linear Programming [OS] When speaking of predictive design of MES and microgrids, Mixed-Integer Linear Programming (MILP) is without doubts the reference optimization approach. Thanks to the adoption of integer and binary variables in the model it allows to precisely describe the on-off logic of the UC. Non-linear equations of typical MES units can be approximated through piece-wise relations using binary variables without impacting the quality of the results [34]. Catalogue-based design can be properly model using binary investment variables [35]. MILP optimization ensures global optimality while modelling the behaviour of the system in detail, and it is nowadays one of the most promising approach to MES design problems. The non-negligible drawback of MILP is its numerical complexity. In fact, integer programming is NP-complete, which essentially means that its computational time tends to explode with increasing number of discrete variables. In particular, since the number of operation binary variables depends on the time horizon and time step of the problem, when increasing the time horizon of unit commitment (from some representative days to entire weeks or months) the solver may struggle in finding the optimal solution in an acceptable amount of time. This is why some researchers nowadays are focused on clustering techniques to limit the time horizon without compromising its representativity [36]. Reducing the computational complexity of MILP algorithms is still an open challenge addressed by many researchers.

Mixed-Integer Non-linear Programming [OS] Mixed-Integer Non-linear Programming (MINLP) methods come with all the advantages of MILP plus the possibility of modelling also non-linear relations. It is worth to say that the vast majority of MES non-linear modelling relations can be easily linearised without compromising the accuracy of the results. Yet, there are some specific cases in which strongly non-linear phenomenon need to be considered for a detailed description of the system, such as power electronic relations [37]. Of course, the solution of complex MES design problems with MINLP requires a tremendous computational power and time, so non-linear programming is generally avoided unless strictly necessary.

Chapter 2

MILP optimization and MES design

2.1 MILP optimization: general concepts and software implementation

In this section we will provide some general insights about MILP optimization, including both theoretical concepts and practical considerations concerning its numerical implementation in complex real-world MES design problems.

2.1.1 Introduction to the Branch and Bound algorithm

A *Mixed-Integer Linear Programming* problem (or *MILP* problem) is a linear optimization problem containing both discrete and continuous variables. The possibility to deal with integer and binary variables makes MILP optimization a very versatile tool nowadays adopted solve a wide number of real world problems in the fields of finance [38], logistics [39], production planning [40], chemical engineering [41], control engineering [42], energy engineering [43] and many others. The most

compact way to express a generic MILP problem is its *canonical form*:

$$\begin{aligned} \min/\max \quad & f = c^T x + d^T y \\ \text{subjected to} \quad & Ax + By \leq b \\ & x \geq 0, x \in \mathbb{R}^n \\ & y \in \mathbb{Z}^m \end{aligned} \tag{2.1}$$

where $x \in \mathbb{R}^n$ is a vector of n continuous variables, $y \in Z^m$ is a vector of m integer variables, $c \in \mathbb{R}^n$ is a vector containing all the n objective coefficients of the continuous variables x, $d \in \mathbb{R}^m$ is a vector containing all the q objective coefficients of the integer variables y, $b \in \mathbb{R}^r$ is a vector of r constant terms, $A \in \mathbb{R}^r \times \mathbb{R}^n$ and $B \in \mathbb{R}^r \times \mathbb{R}^m$ are the so called *constraint matrices* containing the $r \times n$ and $r \times m$ coefficients multiplying the continuous and integer variables appearing in each one of the r linear constraints defining the problem. It is worth noting that both the objective function $f = c^T x + d^T y$ and the set of r constraints $Ax + By \leq b$ are expressed as linear combinations of the continuous and integer variables x and y.

Unfortunately, despite their simplicity of representation, MILP problems are very difficult to solve due to the combinatorial nature of the domain of y variables. In fact, most solution algorithms for MILP problems are based on enumerative techniques whose computational complexity increases exponentially with the number of discrete variables [44]. To have an idea of the exponential complexity of these kind of techniques we can refer to the simplest enumeration technique possible, consisting in an exhaustive search of the discrete solution space where a linear-programming problem containing all and only the x variables of the original MILP is solved by means of a traditional simplex method for each possible combination of the y variables. This trivial solution strategy is so inefficient that, even by assuming a solving time of $10^{-4}s$ for each LP subproblem, it would take more than $2^{48} \cdot 10^{-4}s = 892,55$ years to solve a MILP problem with 48 binary variables (e.g. a hourly unit commitment problem of a single unit on a 2-day time horizon). As a consequence, many advanced algorithms have been developed in literature aiming at reducing the computational time required to solve an instance of a MILP problem.

Brief description of the Branch and Bound scheme

The most widely adopted of these methods is by far the *Branch and Bound (B&B) algorithm*, introduced in 1960 by Land and Doig [45] and nowadays used as a reference algorithm by all commercial solvers [46]. In the following we will refer to a minimization problem without loss of generality. The basic idea behind the B&B algorithm is to adopt a search strategy based on a binary enumeration tree in which each node represents a LP problem characterized by a unique set of linear constraints. More in detail, a *full relaxation* of the MILP problem (that is, a LP version of the problem in which all the y variables are treated as continuous ones) is initially solved in the *root node*. Then, the enumeration procedure is started by taking a variable y_i assuming a fractional value y_i^* and by defining two new distinct LPs (the *child nodes*) only differing from their *parent node* due to the constraints

$$y_i \geq \lceil y_i^* \rceil \tag{2.2}$$

and

$$y_i \leq \lfloor y_i^* \rfloor \tag{2.3}$$

respectively. Of course, if the variable y_i is supposed to be binary, the *branching* of the parent node is simply done by putting $y_i = 0$ and $y_i = 1$. This procedure is repeated recursively, selecting a different *branching variable* at each node among the ones contained in the y vector. In this way, a complete enumeration of the discrete solution space is possible by properly tightening the continuous space with constraints like those reported above (we talk about *implicit enumeration*). A graphical example summarizing the branching procedure of a B&B algorithm is reported in figure 2.1. The root node 0 becomes the parent node of 1 and 2 after branching on the integer variable y_1. 5 and 6 are child nodes of 4, from which are derived after branching of the binary variable y_3.

It is clear that branching alone is just a particular enumeration technique. The real advantage of the B&B method over an exhaustive enumeration method lies instead in the *bounding* procedure which, combined with node *fathoming*, allows to exploit the convexity of the linear problem to reduce the number of explored

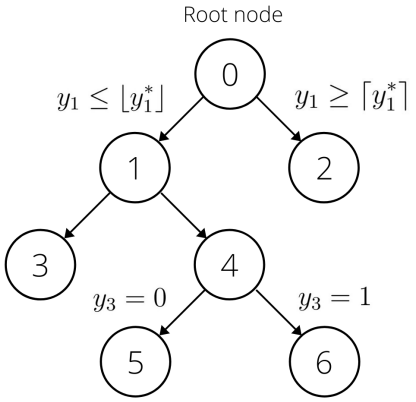


Figure 2.1: The branching procedure of the B&B algorithm

nodes on the enumeration tree. When, during the branching process, all the y variables of an LP solution in certain node take integer values, then a feasible solution of the original MILP problem is found. Such a solution is called *incumbent* and the value of the associated objective function represents an useful upper bound for the objective function of the original MILP problem. In each moment of the optimization process, the lowest among all the upper bounds found during the search is known as the *cutoff value*. A convenient lower bound is also available in each instant by simply taking the highest objective function value among all the feasible nodes explored in the search tree. This value is also known as the *best bound*. While the best bound is only used to tighten the solution space of the various LP subproblems, the cutoff value can be repeatedly compared with the value of the objective function of each explored node to test its sub-optimality. If the objective value of a certain node is higher than the cutoff value, then we conclude that any further branching would lead to child nodes with a worse value of f. Hence, the node is *fathomed* and the corresponding tree branch *pruned*. The same is done if the node is *infeasible* (that is, if the set of linear constraints that define it make the associated LP problem infeasible) or *integer feasible* (that is, if all the y variables have taken integer values so that any further branching is useless).

The B&B algorithm terminates when all the unfathomed nodes of the enumeration

tree are explored. At this point, being the B&B an implicit enumeration algorithm (but still an enumeration algorithm), we are sure that the cutoff value coincides with the global minimum value of f for the MILP problem under consideration. Hence, we can state that the B&B algorithm ensures global optimality. Another interesting feature of the B&B algorithm is that, being both an upper and a lower bound available at each node, it is possible to always estimate the maximum relative error that we would accept by taking the best incumbent as the optimal solution of the problem. Such a parameter is called the *relative gap*, and is defined as the ratio of the difference between the cutoff value and the best bound and the cutoff value.

Advanced Branch and Bound techniques

By increasing the number of pruning opportunities, or by pruning at a higher level of the enumeration tree, it is possible to significantly reduce the number of nodes explored in a B&B instance. To this end, many advanced Branch and Bound techniques have been developed over the years to reduce the computational times without compromising the global optimality of the solution found by the algorithm. A detailed and exhaustive treatment of such techniques is out of the scope of this work. Nevertheless, being the novel algorithm presentend in Chapter 3 based on some of these advanced techniques, an introduction of some selected basic concepts becomes necessary.

Cutting planes A *cutting plane* (or simply a *cut*) is an inequality constraint that tighthens the solution space of the relaxed linear problem without cutting out any possible discrete solution and that is not included in the initial set of constraints defining the MILP problem. If a cut also excludes discrete solutions, then it is a special type of cut called *integer cut*. These additional constraints can be derived in various ways. For instance, they can be generated before the beginning of the branching procedure by directly manipulating the initial set of constraints or during the search algorithm by solving specific instances at each node. In general, as evident in figure 2.2, the tightened solution space generated by a cutting plane always contains the discrete solution space of the original MILP problem (also known as

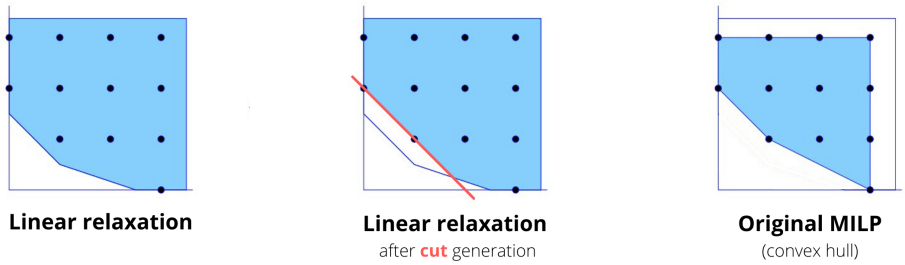


Figure 2.2: Effect of the introduction of a cutting plane on the tightness of the relaxed solution space

convex hull), thus keeping the possibility to achieve a global optimum. Cutting plane generation is an extensive research area in operations research [47], and is so effective in tightening the relaxation of the solution space that, already in 1958 (two years before the pubblication of the B&B algorithm), R. Gomory developed an early algorithm for the solution of MILP problems completely based on recursive cut generation [48], today known as the *Cutting Plane method*. Nevertheless, the most common implementation of cutting planes in modern MILP solvers is an integration with the traditional B&B scheme also known as *Branch and Cut algorithm* [49] [50]. According to this scheme, cuts are continuously generated during the exploration of the search tree to reduce the number of branching operations by cutting out fractional solutions after tightening the relaxed solution space of the nodes.

Heuristics A MILP *heuristic* is a technique aiming at generating an integer feasible solution in a short time. Heuristic techniques don't guarantee any optimality by themselves but, if integrated in a B&B algorithm, they may significantly speed up the convergence time by promoting the generation of good cutoff values without conducting all the necessary branching. Of course, this only happens if the time saved due to ligther branching is higher than the additional time spent to solve the set of heuristics (which, as reported in [51], is typically the case). A very simple but illustrative example is the so called *rounding heuristic* [52], which basically consists in converting the fractional solution of a node in an integer one by rounding all the fractional values. The direction of the rounding procedure (upward or downward)

for each variable is decided so that all the involved constraints are still respected.

Decomposition methods A *decomposition method* consists in partitioning the original MILP problem into smaller MILP or LP problems that are easier to solve. In this way, the original enumeration tree is broken down into several lighter trees that can be possibly discarded all at once by relying on information not available during conventional branching of the original problem. Decomposition methods can be of different types (e.g. column generation, row generation) and, unlike heuristic methods, they can only be applied to very specific kind of problems characterized by well defined structures. Famous MILP decomposition methods are the Benders decomposition [54] (also implented in many commercial solvers [53]) and the Dantzig-Wolfe decomposition [55].

For the sake of completeness, after having mentioned alternative approaches to the B&B algorithm for solving MILP problems (e.g. the Cutting Plane method), it is worth mentioning the paper by Raman and Grossman [56], where a *Logic-Based method* is applied to solve MILP problems pertaining to the field of chemical engineering.

2.1.2 MILP solvers and modeling environments

The numerical solution of a MILP instance can be found by relying on a number commercial or open MILP solvers like CPLEX [57], Gurobi [58], BARON [59], Mosek [60], LINDO [61] (commercial), GLPK [62] and LP_solve [63], BLIS [64], CBC [65], MINTO [66], SCIP [67], SYMPHONY [68] (open). All these softwares take as input a numerical representation of the canonical form reported in eq. (2.1) and provide as output the value of the objective function, together with a solution vector in which the optimal value of each variable of the problem is listed in the corresponding position. As an example, the lines of code required to run a MILP instance in the CPLEX Python API are:

```
prob = cplex.Cplex()
prob.objective.set_sense(prob.objective.sense.minimize)
prob.linear_constraints.add(rhs=b, senses=s)
```

```
prob.variables.add(obj=f, lb=l, ub=u, types=c)
prob.linear_constraints.set_coefficients(zip(rows, cols, vals))
prob.solve()
```

where `b` is a $1 \times r$ list containing the r constant terms appearing in the right-hand side of the linear constraints, `f` is a $1 \times (n+m)$ list containing the objective coefficients appearing in vectors c and d, `l` and `u` are two $1 \times (m+n)$ lists containing the lower and upper bounds for each variable of the problem, `c` is a $1 \times (n+m)$ string in which each character represents the typology of the corresponding variable (real, integer or binary), while each i-th element of the tuple (`rows[i]`, `cols[i]`, `vals[i]`) individuates the coefficient `vals[i]` of the variable `cols[i]` in the constraint `rows[i]` of the A matrix.

The canonical form is undoubtedly the most compact and efficient way to represent a generic MILP problem, which makes it the default input logic by most solvers on the market. Nevertheless, such representation is not informative at all, since it is extremely unpractical to extract any information about the mathematical problem from which it is generated. The *formulation* of a MILP problem is therefore distinguished from its *implementation* in a solver. The first is tipycally done by following an *algebraic description* written in a set-oriented notation in which each variable may be indexed on one or more sets. Similarly, the constraints that bind these variables together (as well as the objective function) are expressed according to the so-called *parameters* of the model, which are also indexed on appropriate sets defined in advance. The basic blocks of a generic algebraic formulation of a MILP problem are then: (i) sets, (ii) parameters, (iii) variables, (iv) constraints, (v) objective function, as clearly seen in the following illustrative case study taken from [69], which also shows the set-oriented indexing logic of the problem's variables and parameters:

Sets

$P :=$ {Seattle, San Diego} *Plants*

$M :=$ {New-York, Chicago, Topeka} *Markets*

Parameters

$a_i :=$ supply of commodity of plant i $\forall i \in P$

$b_j :=$ demand of commodity at market j $\forall j \in M$

$c_{ij} :=$ shipping cost per unit shipment between plant i and market j $\forall i \in P, j \in M$

Variables

$\hat{x}_{ij} :=$ amount of commodity to ship from plant i to market j $\forall i \in P, j \in M$

Constraints

$$\sum_j \hat{x}_{ij} \leq a_i \qquad \forall i \in P$$

$$\sum_i \hat{x}_{ij} \geq b_j \qquad \forall j \in M$$

Objective

$$\text{minimize } f = \sum_i \sum_j c_{ij} \hat{x}_{ij}$$

By using the numeric data reported in [69] for the parameters a_i, b_j and c_{ij}, the canonical form of the above problems writes:

$$\begin{aligned} \min \quad & \begin{bmatrix} 2.5 & 1.7 & 1.8 & 2.5 & 1.8 & 1.4 \end{bmatrix} \begin{bmatrix} x_{13} \\ x_{14} \\ x_{15} \\ x_{23} \\ x_{24} \\ x_{25} \end{bmatrix} \\ \text{subjected to} \quad & \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{13} \\ x_{14} \\ x_{15} \\ x_{23} \\ x_{24} \\ x_{25} \end{bmatrix} \leq \begin{bmatrix} 350 \\ 600 \\ 325 \\ 300 \\ 275 \end{bmatrix} \\ & x_{ij} \geq 0, x_{ij} \in \mathbb{R} \quad \forall i \in P, j \in M \end{aligned} \tag{2.4}$$

It is apparent that, for problems even slightly more complex than the one reported above, the size of the constraint matrix A (whose columns and rows represent the

variables and the constraints, respectively) may explode, making the conversion from the algebraic form to the canonical form impossible to perform by hand. This is why such task is typically automatically run by a class of dedicated software programs also known as *MILP modeling environments*. They take as input a formulation of the problem as similar as possible to the algebraic form and, subsequently, convert it into the canonical form accepted by the low level solver. Some widely adopted MILP modeling environments are: GAMS [70], AMPL [71], PYOMO [72], YALMIP [73], JuMP [74], PuLP [75], ZIMPL [76], CVX [77], CVXPY [78], TOMLAB [79]. Almost all the above mentioned environments are based on a set-oriented synthax aiming at closely emulating the algebraic formulation. For instance, the GAMS formulation of the problem example reported above is the following:

```
Sets
   i   canning plants   / Seattle, San-Diego /
   j   markets          / New-York, Chicago, Topeka / ;

Parameters
   a(i)  capacity of plant i in cases
      /    Seattle       350
           San-Diego     600  /
   b(j)  demand at market j in cases
      /    New-York      325
            Chicago      300
            Topeka       275  / ;

Table  d(i,j)  distance in thousands of miles
                          New-York       Chicago      Topeka
          Seattle           2.5            1.7         1.8
          San-Diego         2.5            1.8         1.4  ;

Scalar f  freight in dollars per case per thousand miles  /90/ ;

Parameter
      c(i,j)  transport cost in thousands of dollars per case ;
c(i,j) = f * d(i,j) / 1000 ;

Variables
       x(i,j)  shipment quantities in cases
       f       total transportation costs in thousands of dollars ;
Positive variables x ;
```

```
Equations
       cost             define objective function
       supply(i)        observe supply limit at plant i
       demand(j)        satisfy demand at market j ;
cost ..        f  =e=  sum((i,j), c(i,j)*x(i,j)) ;
supply(i) ..   sum(j, x(i,j))  =l=  a(i) ;
demand(j) ..   sum(i, x(i,j))  =g=  b(j) ;
Model transport /all/ ;
Solve transport using LP minimizing f ;
```

A peculiar exception among MILP modeling environments is "YALMIP: A toolbox for modeling and optimization in MATLAB" [80]. In fact, unlike other well-known packages such as Pyomo, AMPL and GAMS, YALMIP does not support the definition of sets and the consequent indexing of variables and parameters. This missing functionality represents a big barrier to the use of MATLAB for the resolution of MILP optimization problems, especially considering that:

- To the best of the authors' knowledge, there aren't other open-source alternatives to YALMIP for high-level modeling of MILP problems in MATLAB;
- MATLAB is a programming language widely adopted in the industrial/engineering environment;
- A MILP problem can represent a constitutive element of a "master" MATLAB code with which it interacts dynamically. Not everyone who needs to solve a MILP problem (e.g. industrial engineers) has the expertise to develop their own code in environments other than MATLAB (e.g. in Python through Pyomo), while data exchange between MATLAB and softwares like GAMS and AMPL is rather slow and cumbersome.

2.2 POLIMIP: a set-oriented MILP modeling environment for MATLAB

It has already been underlined how one of the biggest limitations of YALMIP as a MILP modelling environment is the impossibility to adopt a "GAMS-like" indexed syntax for the definition of variables and parameters. However, this isn't the only

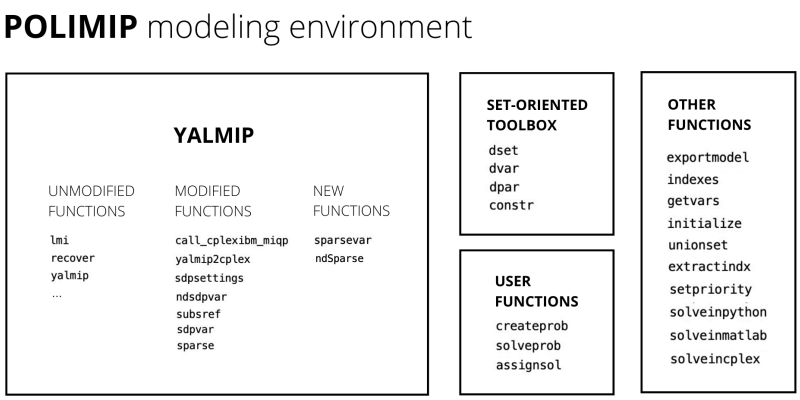


Figure 2.3: General organization and main modules of POLIMIP

limitation that differentiates YALMIP from other similar softwares like Pyomo. YALMIP, in fact, is not compatible with many advanced commercial solver features such as the definition of branching priorities in the branch and bound algorithm or the use of callback functions during the solver search. Depending on the particular functionality, these limitations may be attributable to two distinct reasons: (i) a lack of compatibility between YALMIP and the MATLAB API of the low level solver, (ii) the absence of the advanced functionality in the MATLAB API of the low level solver.

"POLIMIP: a set-oriented MILP modeling environment for MATLAB" is a novel modeling toolbox developed to overcome the three main limits of YALMIP in modeling and solving MILP problems in MATLAB, namely: (i) absence of a set-oriented synthax, (ii) absence of full compatibility with the MATLAB API of the MILP solver and (iii) absence of advanced functionalities in the MATLAB API of the MILP solver.

2.2.1 Problem I: absence of a set-oriented synthax.

A comparison between the YALMIP and AMPL synthaxes in defining a real variable $x_{ij}\ \forall i \in S_1, j \in S_2$ with $n(S_1) = 2$ and $n(S_2) = 3$ is reported here:

YALMIP

```
x = sdpvar(2,3)
```

AMPL

```
var x {set1,set2}
```

From this very simple example, it is already possible to highlight the total absence of the mathematical concept of "set" in the YALMIP environment, which adopts a "matrix-oriented style" . Moving to the definition of a parameter $p_i \ \forall i \in S_3$ with $n(S_3) = 4$ we notice that in YALMIP we simply rely on the definition of a MATLAB array, while in AMPL we still keep a set-logic:

YALMIP

```
p = [1 2 3; 4 5 6]
```

AMPL

```
param p {set3}
```

Finally, examining the constraints formulation, the algebraic description $x_{ij} < p_{ij} \quad \forall i \in S_1, j \in S_2$ becomes:

YALMIP

```
for i = 1:2
  for j = 1:3
    for k = 1:2
      Constraints = [ Constraints, "x(i,j) <= p(i,k)"]:
    end
  end
end
```

AMPL

```
subject to constraint {i in set1, j in set2, k in set3}: x[i,j] <= p[i,k]
```

The previous examples suggests us to model mathematical sets in MATLAB as row arrays of length equal to their cardinality, that we will call *set-arrays*. Apparently, in order to allow for set operations as set union and intersection, each element of the array must indicate one and only one of the elements of the associated mathematical set, hence being its one-to-one image. For this purpose, each element of each set-array must be unique (that is, it cannot be present in other set-arrays of

the same MILP problem). The most elegant and immediate way to implement this feature is through a one-to-one correspondence of the elements of each set-array with a subset of the natural numbers. More formally, the generic set-array with N discrete elements is defined in MATLAB as:

```
set = i:j
```

where `i` and `j` are two natural numbers (with `N = i - j + 1`) belonging exclusively to the set-array they are defining. With reference to the above example (opting a mapping starting from 1 without "holes") we have then:

```
set1 = 1:2
set2 = 3:5
set3 = 6:7
```

This logic is particularly convenient, as it allows each image in a set-array to be directly used as an index of the set element it represents. In practical terms, the image relative to the third element of `set2` (that is, 5) is obtainable as `set2(3)`, and this formal writing can be extended to represent also the associated index. Following this logic, the element of the parameter `p` relative to the first element of `set2` and to the second element of `set3` will be characterized by the writing `p(set2(1),set3(2))`, while the exmaple of constraint reported above becomes:

```
for i = set1
  for j = set2
    for k = set3
      Constraints = [ Constraints, "x(i,j) <= p(i,k)"]:
    end
  end
end
```

which closely resembles the algebraic notation. The price to pay for the elegance of such a notation lies in the sparsity of the matrix `p`, which will contain zeros in the positions from `(1,1)` to `(2,7)`. The same applies to the `sdpvar` variables.

Efficiency in numerical representation of sparse matrices and vectors is a key requirement for MILP modeling softwares that implement the set-array indexing

logic described above. In fact, in the presence of an inefficient numerical representation, such indexing logic could be incompatible with MILP problems characterized by a high number of variables and/or parameters due to RAM memory saturation. As an illustrative example, let's consider a problem consisting of four sets containing 3000 elements each:

```
set1 = 1:3000
set2 = 3001:6000
set3 = 6001:9000
set4 = 9001:12000
```

Let's then consider two parameters `p1` and `p2` defined on `set1` and `set2` and on `set1` and `set4`, respectively. These parameters have the same number of elements (i.e. 9000000) but, due to the indexing logic adopted, the `p1` array has a size of 6000×6000 and memory occupation of 288Mb, while the `p2` parameter has a size of 12000×12000 and a memory occupation of 1.15Gb of memory ($4\times$). Moreover, the ratio between these two memory sizes increases quadratically with the number of sets on which `p1` and `p2` are indexed. It follows that, considering problems with larger sets and/or with multiple variables/parameters (possibly indexed on more than two sets), this inefficiency in memory usage, as well as negatively affecting computational performance, may lead to the numerical unrepresentability of the problem. To relieve this issue, the MATLAB `sparse` function allows to efficiently define sparse vectors and 2D matrices by significantly cutting the memory overhead due to the presence of zero elements. For instance, with reference to the above example, the commands `p1 = sparse(p1)` and `p2 = sparse(p2)` reduce the memory size of both `p1` and `p2` to 144Mb from 288Mb and 1.15Gb, respectively. However, this built-in function cannot be extended to N-dimensional matrices with $N > 2$. This limit is unacceptable for our purposes, as it does not allow to define sparse parameters indexed on more than two sets (that are those objects on which the representation overhead introduced by the proposed indexing scheme impacts most). The same applies to variables (i.e. YALMIP's `sdpvar` objects), as openly stated by its developer Johan Lofberg [81]. It is then reasonable to conclude that the reason behind the absence of a set-oriented indexing logic in YALMIP lies in the limitations of the MATLAB function `sparse`, which can only be applied

to one-dimensional and two-dimensional matrices, hence not allowing an efficient representation of variables and parameters indexed on more than two sets.

This initial barrier can be overcome thanks to the `ndSparse` function by M. Jakobson and M. Volker [82], which is basically an extension of the built-in `sparse` function that also applies to multi-dimensional arrays. In addition to directly solving the problem of N-dimensional sparse parameters, this function was also used to develop a modified version of YALMIP containing the new function `sparsevar`, which introduced the possibility to define multi-dimensional sparse `sdpvar` objects. The synthax of the function call is illustrated in the following example, in which a sparse object `test_var` containing two `sdpvar` objects (one in position (20, 30, 40) and another in position (200, 300, 400)) is defined:

```
test_var = sparsevar([20, 30, 40; 200, 300, 400],ndspvar(1,2))
```

The new custom version of YALMIP (embedding `sparsevar` for the definition of N-dimensional sparse variables and `ndSparse` for the definition of N-dimensional sparse `double` arrays) finally opened the way to an efficient implementation of the set-oriented modeling synthax built around the concept of *set-array* previously introduced. This implementation was done by developing a dedicated MATLAB toolbox in which the four core elements are represented by the functions `dset`, `dpar`, `dvar` and `constr`[1] for the GAMS-like definition of sets, parameters, variables and constraints, respectively. To easily compare the synthax of this novel set-oriented modeling environment with the algebraic and GAMS formulations previously reported, we still refer to the simple transport problem introduced in section **??**:

`sets.m`

```
% Definition of set "I" with 2 elements
dset('I',2)

% Definition of set "J" with 3 elements
dset('J',3)
```

[1] A detailed description of these functions is available in the POLIMIP Quick User Guide reported in **??**

data.m

```
% Data for parameter ‘a’
a = [350, 600];

% Data for parameter ‘b’
b = [325, 300, 275];

% Data for parameter ‘c’
d = [2.5, 1.7, 1.8; 2.5, 1.8, 1.4];
f = 90;
c = f * d / 1000;
```

parameters_v1.m

```
% Definition of parameter ‘a’
dpar(‘a’,"I")

% Definition of parameter ‘ b’
dpar(‘ b’,"J")

% Definition of parameter ‘ c’
dpar(‘c’,"I","J")
```

parameters_v2.m

```
% Definition of parameter ‘a’
for i = s.I
  dpar(‘a’,i)
end

% Definition of parameter ‘b’
for j = s.J
  dpar(‘b’,j)
end

% Definition of parameter ‘c’
for i = s.I
  for j = s.J
    dpar(‘c’,i,j)
  end
end
```

variables_v1.m

```
% Definition of variable x
dvar(‘x’,‘real’,"I","J")
```

variables_v2.m

```
% Definition of variable x
for i = s.I
  for j = s.J
    dvar(‘x’,‘real’,i,j)
  end
end
```

objective.m

```
% Definition of the objective function
Objective = sum(sum(p.c(s.I,s.J).*v.x(s.I,s.J)));
```

constraints.m

```
% Definition of the first constraint
for i = s.I
  constr(‘sum(v.x(i,s.J)) <= p.a(i)’)
end

% Definition of the second constraint
for j = s.J
  constr(‘sum(v.x(s.I,j)) >= p.b(j)’)
end
```

where the scripts ending with `v1.m` and `v2.m` indicate alternative notations for the definition of the problem variables and parameters. A strong resemblance between the algebraic description of the problem and the POLIMIP programming synthax can be noticed. We hope that this will help make optimization modeling in MATLAB more accessible to academics and professionals from many different fields.

2.2.2 Problem II: incompatibilities between the modelling environment and the MATLAB API of the MILP solver.

For many classes of MILP problems, it is possible to significantly speed up the solver time by properly tuning the B&B search algorithm based on our categorical

knowledge of the problem. As an example, CPLEX gives the possibility to freely choose which integer/binary variables to branch first during the binary tree search. As reported in [83], this may lead to better performances in problems characterised by an internal hierarchy between integer variables: "Problems that use integer variables to represent different types of decisions should assign higher priority to those that must be decided first. For example, if some variables in a model activate processes, and others use those activated processes, then the first group of variables should be assigned higher priority than the second group. In that way, you can use priority to achieve better solutions". This is the exact situation encountered in typical MES design problems: by fixing integer and binary design variables we activate processes involving integer and binary operation variables. Unfortunately, at the moment of writing, YALMIP still doesn't allow to set branching priorities for `sdpvar` objects, even though the MATLAB API of CPLEX does accept this kind of annotation [84]. There is therefore an incompatibility issue between the modeling environment (YALMIP) and the solver (CPLEX MATLAB API). In POLIMIP we have introduced this functionality by suitably modifying the YALMIP-CPLEX interface code and by including some dedicated functions for variable annotation (such as `setpriority`) in the proposed modeling toolbox.

It has to be noticed that the case of *branching priority orders* is just one of several examples of incompatibility between the high-level modeling software and the low-level solver API. However, the implementation described above can be easily repeated for every tuning parameter of each supported solver, thus allowing a full compatibility between POLIMIP and the functionalities of the MATLAB API of most commercial MILP solvers on the market.

2.2.3 Problem III: absence of advanced functionalities in the MATLAB API of the MILP solver.

Most commercial solvers are distributed in the form of APIs for different coding languages. For instance, CPLEX and Gurobi APIs are available for C, C++, Java, .NET, Python, R and MATLAB. Nonetheless, not all solver functionalities are

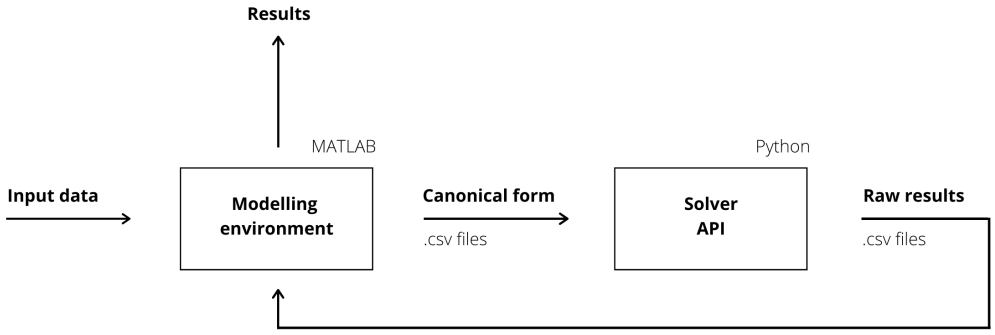


Figure 2.4: Multi-language architecture adopted to implement advanced solver functionalities not supported by the MATLAB API of the solver

available in each API. For instance, as reported in [85], the MATLAB API of CPLEX does not support *control callbacks*[2], and the same applies to Gurobi. The impossibility to use control callbacks in the MATLAB environment introduces a significant limitation to the possibility of implementing advanced solution strategies based on modifications of the default branching logic of the solver. However this time, unlike what seen before with reference to branching priority orders, the incompatibility is at the solver level and, therefore, it cannot be solved by simply upgrading the code of the modeling environment (in our specific case, YALMIP).

The only way to overcome this restriction is by means of a multi-language architecture characterised by an interface module between the modelling environment (written in MATLAB) and a fully-compatible solver API (written in Python, C, C++, .NET or in Java). The proposed architecture is reported graphically in figure 2.4.

The problem data are directly provided by the user to MATLAB through dedicated scripts involving the POLIMIP functions `dset`, `dvar`, `dpar` and `constr` (as reported in a previous illustrative example). Other possible inputs are the solver options and the variable annotations for setting the branching priority orders. Once

[2] A *control callback* is a special function that allow user code to be executed regularly during the tree search.

all the raw information required to solve the optimization problem is stored in the MATLAB working environment, the POLIMP toolbox starts the generation of the canonical form of the problem, which is eventually saved on the current working folder as a list of solver-ready .csv files. These files are then passed to a dedicated program written in one of the coding languages mentioned above, whose task is to call the solver API with full compatibility and eventually find the solution to the problem. The latter is thus stored in a further .csv file that is immediately recovered in MATLAB by the POLIMIP toolbox, hence allowing for a convenient post-processing of the results.

The API Python was preferred over the other available coding languages due to its higher understandability and adoption. Moreover, since .py modules can be directly called and executed in the MATLAB environment [86], this choice opens to the possibility of implementing a single-platform and integrated toolbox, in which the flow of information between MATLAB and Python is flawlessly managed by tools such as the MATLAB Engine API for Python [87]. In this way, the decoupling between the modeling environment and the solver program is completely transparent to the user, who then only interacts with the multi-language architecture through traditional MATLAB scripts.

2.3 Full-MILP algorithms for the optimal design of MES

As already reported in section 1.2.1, MILP algorithms are just one of many possible approaches to MES design optimization. Their undiscussed advantage lies in the ability to find a global optimum, which is a very important feature for particularly complex problems (such as MES design and operation problems) whose solution space is characterised by a very high number of local optima. On the other hand, when dealing with full-MILP algorithms, the price to be paid is a significant computational complexity linked to the exponential increase of the potentially explored tree nodes with the number of integer and binary vari-

ables. To mitigate this issue, a common approach is to perform a sort of *hybrid decomposition* of the optimization problem into a design and an operation layer. In this hybrid architecture, the upper design layer is usually solved by means of an evolutionary algorithm, while the lower operation level is solved as a MILP problem with fixed design [88] [31]. This approach aims at lowering the computational time of the algorithm by renouncing to the global optimality of the solution.

In this long-running challenge between different solution approaches and paradigms, the theorem proved in [90] paves the way to some interesting observations: "*there is no polynomial time algorithm*[3] *for solving the synthesis problem of decentralized energy systems unless* $P = NP$[4] ". The general validity of this statement (it is true for *any* possible algorithm) somehow legitimates the adoption and development of advanced *full-MILP algorithms* for the solution of MES design problems by stating that an exponential increase of the computational time with the problem size should be expected by non-MILP algorithms as well.

In the following paragraph, a critical review of the state-of-the-art strategies adopted to reduce the computational complexity of full-MILP algorithms for the optimization of MES design is carried out to better understand the main advantages introduced with the novel decomposition method presented in the next chapter.

2.3.1 Complexity reduction strategies for full-MILP MES design methods

As evident in figure 2.5, the attention of the scientific community towards full-MILP methods for the design of MES gained relevance in recent years[5].

[3] A polynomial-time algorithm is an algorithm whose execution time is either given by a polynomial on the size of the input, or can be bounded by such a polynomial.

[4] The P versus NP problem is a major unsolved problem in computer science. It is one of the seven Millennium Prize Problems selected by the Clay Mathematics Institute, and it is a widely spread belief that its solution is $P \neq NP$ [91].

[5] The research has been conducted on Clarivate Analytics' Web of Science [92] by comparing the two entries *((milp OR (mixed integer linear programming) OR (mixed-integer linear programming)) AND ((energy syst*) OR microgrid OR micro-grid) AND design)* and *(((energy syst*) OR*

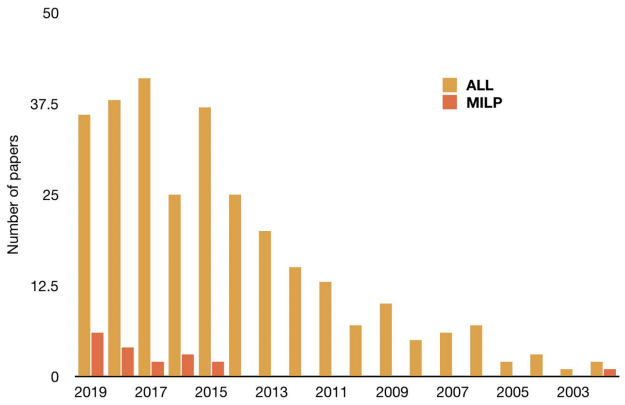


Figure 2.5: Number of papers concerning MILP methods for the design of microgrids and multi-energy systems vs. number of papers concerning all kinds of methods for the design of microgrids and multi-energy systems (trend).

One of the main reasons behind this increasing trend is undoubtedly the development of advanced solution methods allowing for a reduction of the computational effort required to solve the MILP formulation of the problem, thus allowing an accurate optimization of complex real-world microgrid and multi-energy systems in a reasonable solver time. These methods can be divided into four (possibly overlapping) categories.

1. Model simplification One of the easiest ways to reduce numerical complexity is by directly acting on one or more of the modelling choices behind the mathematical description of the energy system, simplifying or even neglecting some of its variables and/or governing equations. Of course, a simplified mathematical model is always associated with a less accurate physical description of the system's features, hence resulting in a reduced soundness of the objective function. Some model simplification strategies commonly found in MES design problems are: adoption of linear performance curves for the generating units (no piecewise approximation) [93], continuous capacities [95] and absence of dynamic features such as ramp-up limits, minimum up/down time and start-up/shut-down penalties [94].

microgrid OR micro-grid) AND design).

2. Time aggregation An exponential increase of the size of the combinatorial solution space with the number of time steps is observed when binary or integer operation variables (typically used to model the on/off logic of the generators) are introduced in the MILP formulation of the problem. As a direct consequence, the number of time steps that can be included in the problem is strongly influenced by the increase in its computational complexity. The modelist is then faced with the choice of mantaining the initial time horizon by increasing the time-step (time aggregation) or, alternatively, of reducing the length of the time horizon. Of course, both choices are not always viable or acceptable. For instance, for energy systems containing a seasonal storage (e.g. an hydrogen tank coupled with a reversible PEM fuel cell), an excessive shrinking of the time horizon would lead to a limited exploitation of the storage element. In such cases, where capturing fast dynamics is less relevant than considering a sufficiently large time horizon, time aggregation is the preferred simplifying solution. A typical time-step for MES design problems is 1 hour [20] [96], but time aggregations up to 2 [97] and 4 hours [98] are also found in literature.

3. Solution space reduction The size of the combinatorial solution space of a MILP problem actually explored by a branch-and-bound search algorithm can be significantly reduced by providing an initial upper bound associated with a sub-optimal integer feasible solution [99]. In this way, all the nodes leading to a local objective value higher than the initial cutoff value will be pruned due to sub-optimality, thus reducing the number of explored combinations. In a commercial MILP software, this can be done by implementing an user-defined heuristic in the pre-processing phase providing a so called "MIP start" to the solver [100]. As far as MES are concerned, an easy but effective way to find an initial incumbent is by guessing a reasonably oversized and/or redundant design and by solving the associated integer operation problem. Regardless of the quality of the design candidate, this procedure may still lead to better incumbents than the ones found by the solver during the default root node processing phase, since it relies on problem information that may seem trivial to the user, but not to a general-purpose solver.

4. Decomposition methods Decomposition methods are at the same time the best performing and the less intuitive among the four complexity reduction strategies presented in this section. In one line, they can be intended as *custom branch-and-bound search strategies exploiting the peculiarities of the problem's structure to reduce its combinatorial complexity* [101]. This is mainly done by breaking the original problem into smaller sub-problems that are easier to solve and that eventually provide integer cuts to reduce the size of the explored portion of the original search tree. The main price to be paid for this higher effectiveness is a marked problem-dependency. Moreover, since all these methods are based on the iterative solution of one or more user-defined sub-problems in some specific points of the branch-and-bound search algorithm, their numerical implementation is quite a hard task since it necessarily requires an in-depth knowledge and control of the solution strategy of the numerical solver. The novel decomposition algorithm proposed in the following chapter aims precisely at mitigating these two main limitations: (i) by broadening the class of treatable MES problems with respect to previous methods and (ii) by automatizing the numerical definition and implementation of their decomposed formulation on a widely adopted commercial solver.

Chapter 3

A novel MILP decomposition algorithm allowing for mixed integer and continuous MES design

As mentioned in sec.1.2 and extensively discussed in sec.2.3, MILP is undeniably one of the most promising approaches for design optimization of multi-energy systems since it combines high modelling flexibility with the capability of ensuring global optimality. The main issue associated to MILP optimal design problems is the computational time required by the search algorithm. In MILP problems, numerical complexity increases exponentially with the number of binary and integer variables due to the expansion of the branch and bound tree (see subsec.2.1.1). In particular, in MES design the number of operation binary variables depends on the selected time horizon of the problem. So, when increasing the time horizon of unit commitment from some representative days to entire weeks or months in order to enhance the representatively of the solution, the solver may struggle in finding the optimal solution in an acceptable amount of time. In this chapter, after a brief review of the current state-of-the-art MILP decomposition algorithms for MES design, we introduce an innovative general framework that may be applied to decompose the branch and bound procedure of a MES design problem in order to

increase computational performance and make the solution of complex multi-period models viable while mantaining the guarantee of global optimality ensured by a traditional branch-and-bound scheme.

3.1 An introduction to MILP decomposition of MES problems: exploiting the internal hierarchy between design and operation

A generic *decomposition method* consists in partitioning the original MILP problem into smaller MILP or LP problems that are easier to solve. In this way, the original enumeration tree is broken down into several lighter trees that can be possibly discarded all at once by relying on information not available during conventional branching of the original problem. The main subproblem is also called *master problem*, while the others subproblems solved during the B&B to restore all the original constraints are also reported as *worker problems* [103]. As suggested by the name itself, **hierarchical decomposition** distinguish between master problem and worker problems on the basis of an inherent hierarchy between different integer variables of the model. Very similarly to the case of branching priorities (see subsec.2.2.2), a high-hierarchy variable is the one that activate the process that other variables use. There is no clearer hierarchy than that which exists between the design and operation variables of a MES design problem: by fixing integer design variable (e.g. the investment decision on a given unit) we activate processes involving integer operation variables (e.g. on/off status variables of that unit).

3.1.1 Implementing branching priority orders in MES problems by means of an equivalent decomposition method

When applying a hierarchical decomposition the B&B tree of the master problem is divided between *upper level* and *lower level*. The upper level is the part of the tree that is explored normally branching the high-hierarchy integer variables. The

lower level is the part of the tree where worker problems are solved separately from the master. We can define *high-hierarchy* variables the integer variables that are branched in the upper level, while we indicate as *low-hierarchy* variables the integer variables that are still relaxed when entering the lower level. To better understand how hierarchical MES decomposition algorithms affect the search path of a traditional branch-and-bound tree we can rely on a simple but illustrative excercise that consists in implementing the *branching priorities* for the integer and binary design variables of a MES problem by means of an equivalent decomposition method.

The behaviour of the resulting B&B algorithm is presented in fig.3.1. The binary search tree starts exploring the master problem branching only high-hierarchy variables. We are in the upper level. Going deep into the tree we find a node where all the high-hierarchy variables are well constrained and assume an integer value, while low-hierarchy variables are still relaxed. We found an *entrance node* to the lower level. In the lower level, the worker problem is solved with fixed values of the high-hierarchy variables, hence restoring the coherence between the two levels and providing a solutions respecting all the constraints of the original problem.

To define equivalent decomposition in a formal way, let us first consider the following mathematical formulation of a generic MES problem:

$$\begin{aligned} \min \quad & f\Big(z_D, y_D, x_D, z_O, y_O, x_O\Big) \\ \text{subjected to} \quad & l\Big(z_D, y_D, x_D, z_O, y_O, x_O\Big) \leq 0 \\ & z_D \in \{0,1\}^{n_z} \\ & y_D \in \mathbb{Z}^{n_y} \\ & x_D \in \mathbb{R}^{n_x} \\ & z_O \in \{0,1\}^{m_z} \\ & y_O \in \mathbb{Z}^{m_y} \\ & x_O \in \mathbb{R}^{m_x} \end{aligned} \tag{3.1}$$

where f is the cost function (a linear combination of all the problem's variables), z_D, y_D and x_D are the vectors containing the n_z binary, n_y integer and n_x continuous

design variables, while z_O, y_O and x_O are the vectors containing the m_z binary, m_y integer and m_x continuous operation variables defining the problem. Let's also consider the following modified version of problem 3.1:

$$\begin{aligned} \min \quad & f\Big(z_D, y_D, x_D, z_O, y_O, x_O\Big) \\ \text{subjected to} \quad & l\Big(z_D, y_D, x_D, z_O, y_O, x_O\Big) \leq 0 \\ & z_D \in \{0,1\}^{n_z} \\ & y_D \in \mathbb{Z}^{n_y} \\ & x_D \in \mathbb{R}^{n_x} \\ & z_O \in \mathbb{R}^{m_z} \\ & y_O \in \mathbb{R}^{m_y} \\ & x_O \in \mathbb{R}^{m_x} \end{aligned} \tag{3.2}$$

By solving this problem (which is nothing more than the original problem but with a relaxed operation) with a conventional solver, we end up finding an integer combination $(\bar{y}_D, \bar{z}_D)$ in correspondence of a so-called *entrance node*. This new *integer design candidate* is then used to define the following worker problem, aiming at restoring the integrity of the operation variables for the specific integer design candidate found at the entrance node:

$$\begin{aligned} \min \quad & f\Big(z_D, y_D, x_D, z_O, y_O, x_O\Big) \\ \text{subjected to} \quad & l\Big(z_D, y_D, x_D, z_O, y_O, x_O\Big) \leq 0 \\ & z_D \in \{0,1\}^{n_z} \\ & y_D \in \mathbb{Z}^{n_y} \\ & x_D \in \mathbb{R}^{n_x} \\ & z_O \in \{0,1\}^{m_z} \\ & y_O \in \mathbb{Z}^{m_y} \\ & x_O \in \mathbb{R}^{m_x} \\ & y_D = \bar{y}_D \\ & z_D = \bar{z}_D \end{aligned} \tag{3.3}$$

This problem is solved with a conventional B&B algorithm in which five scenarios can be encountered after solving the relaxed LP problem at each node:

1. The LP problem is infeasible: the node is fathomed and the branching operation is continued;

2. The LP problem is feasible but sub-optimal: the node is fathomed and the branching operation is continued;

3. The LP problem is feasible and its objective is below the global cutoff value (see subsec.2.1.1): the branching operation is continued;

4. The LP problem is integer feasible and its objective is below the global cutoff value: the global cutoff value is updated and the solution of the worker problem is saved as the new *global incumbent*;

5. The LP problem is integer feasible and its objective is above the global cutoff value: the integer solution is rejected due to sub-optimality;

In any case, the algorithm goes back to the upper level (problem 3.2) and starts again branching the high-hierarchy design variables until a new entrance node (that is, a new incumbent) is found or a convergence criterion is met, typically a specific value of the relative gap.

Of course, when speaking of branching priorities the distinctions between upper level and lower level, master problem and worker problems are purely conceptual. What happens in reality is a simple B&B search algorithm in which the branching operations of the low-hierarchy variables start only when all high-hierarchy variables assume integer/binary values. Nonetheless, what has been presented as a possible schematization of a prioritized B&B is exactly what happens in a hierarchical decomposition of a MES optimization problem. Due to the clear hierarchical relationship between design and operation in MES model, we consider *high-hierarchy variables* of a MES design problem all the integer design variables (e.g. investment decisions, number of units, discrete capacities, etc.). On the other hand, integer operation variables are considered *low-hierarchy variables* and are always relaxed in the upper level. One of the most important characteristics of MES models is that low-hierarchy variables are typically organized in different independent sub-groups. This occurs

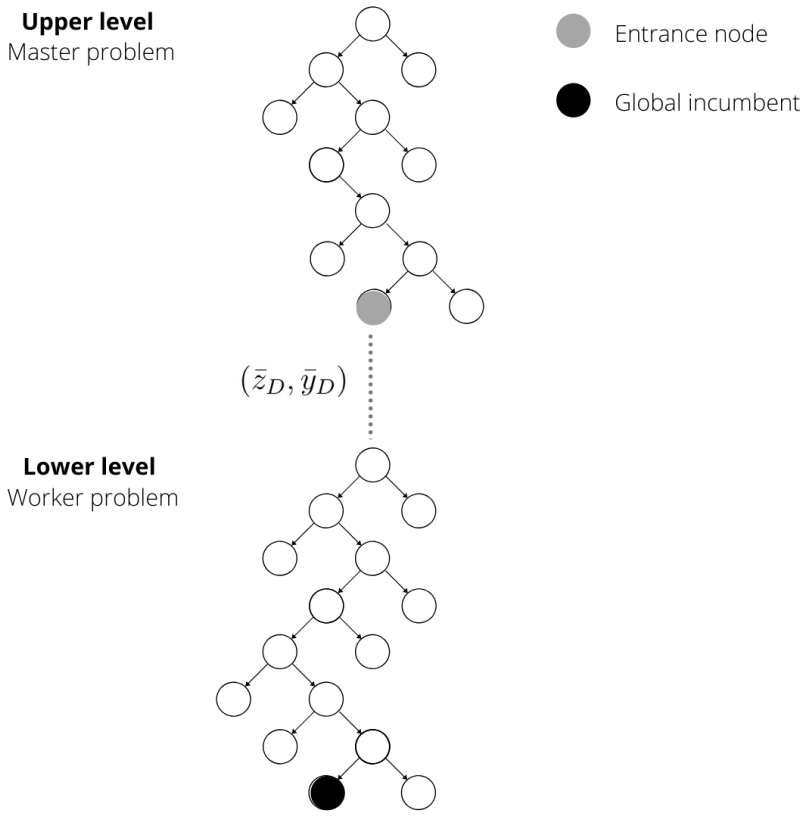


Figure 3.1: Branching priorities interpreted as hierarchical decomposition

thanks to the subdivision of the UC horizon in multiple typical representative time periods (typical days, weeks, etc.). As we will see, typical periods are determining in improving the performance of a hierarchical decomposition algorithm.

3.1.2 Critical review of two state-of-the-art hierarchical decomposition methods for MES design: the Iyer/Grossmann and the Yokoyama decompositions

In this section, two highly performing and widely known decomposition approaches for MES design are critically analyzed to highlight their main limits and potentialities and to put into perspective the novel decomposition method proposed in the following section. The two methods here analyzed are those developed by Iyer and Grossmann[1] (1998) [102] and by Yokoyama[2] (2015) [104]. Both methods are based on some restricting assumptions on the problem's structure that can be clearly highlighted starting from a direct comparison with the general formulation reported in eq. (3.1).

The Iyer and Grossmann decomposition

The additional restricting assumptions on the problem structure advanced by Iyer and Grossmann [102] with respect to the general formulation reported in (3.1) are the followings:

- There are no integer design variables: $n_y = 0$;
- There are no integer operation variables: $m_y = 0$;
- Each binary design variable represents the investment variable of a potential unit;
- Each binary operation variable represents the on-off variable of a potential unit in a specific time-step;

[1] 98 times cited at the time of writing.
[2] 34 times cited at the time of writing.

- Each continuous design variable represents the capacity of an associated unit: $n_x = n_z := n$.

The resulting reference formulation is then:

$$\begin{aligned}
\min \quad & f_D(z_D, x_D) + \sum f_{O_t}(z_{O_t}, x_{O_t}) \\
\text{subjected to} \quad & g_t\Big(z_D, x_D, z_{O_t}, x_{O_t}\Big) \leq 0 \qquad \forall t \in \{1, \dots, T\} \\
& h\Big(z_D, x_D, z_{O_1}, x_{O_1}, z_{O_2}, x_{O_2}, ..., z_{O_T}, x_{O_T}\Big) \leq 0 \\
& z_D \in \{0,1\}^n \\
& x_D \in \mathbb{R}^n \\
& z_{O_t} \in \{0,1\}^n \qquad \forall t \in \{1, \dots, T\} \\
& x_{O_t} \in \mathbb{R}^n \qquad \forall t \in \{1, \dots, T\}
\end{aligned} \tag{3.4}$$

where the problem constraints l have been divided into two main categories: the *uncoupled constraints* g_t (constraints containing only operation variables indexed on a single time-step) and the *coupling constraints* h (constraints containing operation variables indexed on more than one time-step). It is worth noting that, being f a linear combination of the problem's variables, it can always be expressed as a sum of terms depending on a smaller subset of variables, as done in (3.4).

The decomposition proposed by the authors consists of a master (design) problem providing a tentative combination $\bar{z}_D$ of the binary design variables, which is passed to a single worker (pseudo-operation) problem with the task of determining the corresponding feasible values of x_D, x_O and z_O. The design problem (upper level) is a modified version of the original problem reported in eq. (3.4) in which all the constraints containing the binary operation variables z_{O_t} are eliminated and all the corresponding objective function coefficients put equal to zero:

$$\begin{aligned}
\min \quad & f_D(z_D, x_D) + \sum f'_{O_t}(x_{O_t}) \\
\text{subjected to} \quad & g'_t\Big(z_D, x_D, x_{O_t}\Big) \leq 0 \quad \forall t \in \{1, \ldots, T\} \\
& h'\Big(z_D, x_D, x_{O_1}, x_{O_2}, ..., x_{O_T}\Big) \leq 0 \\
& z_D \in \{0,1\}^n \\
& x_D \in \mathbb{R}^n \\
& x_{O_t} \in \mathbb{R}^n \qquad \forall t \in \{1, \ldots, T\}
\end{aligned} \tag{3.5}$$

The pseudo-operation subproblem (lower level) is instead identical to the original one, but with the binary design variables fixed by the design problem:

$$\begin{aligned}
\min \quad & f_D(z_D, x_D) + \sum f_{O_t}(z_{O_t}, x_{O_t}) \\
\text{subjected to} \quad & g_t\Big(z_D, x_D, z_{O_t}, x_{O_t}\Big) \leq 0 \quad \forall t \in \{1, \ldots, T\} \\
& h\Big(z_D, x_D, z_{O_1}, x_{O_1}, z_{O_2}, x_{O_2}, ..., z_{O_T}, x_{O_T}\Big) \leq 0 \\
& z_D \in \{0,1\}^n \\
& x_D \in \mathbb{R}^n \\
& z_{O_t} \in \{0,1\}^n \qquad \forall t \in \{1, \ldots, T\} \\
& x_{O_t} \in \mathbb{R}^n \qquad \forall t \in \{1, \ldots, T\} \\
& z_D = \bar{z}_D
\end{aligned} \tag{3.6}$$

From an algorithmic standpoint, the implementation of the decomposition is done by iteratively solving the sequence of design and pseudo-operation problems and, every time an integer feasible solution is found in the lower level, by adding a special set of integer and design cuts avoiding the multiple generation of the same design candidate $\bar{z}_D$ and enforcing the coherence between upper and lower level (see figure 3.2). The algorithm is implemented and tested on a real case-study involving a CHP plant with a superstructure of more than 20 potential units, outperforming the computational performance of the traditional branch-and-cut solver of one order of magnitude.

It is important to point out that, as the authors state in their paper, "*the (operation) problem, however, must be solved simultaneously due to the linking constraints h*" that, in the case study reported in the paper, are used to model the maximum

up-time of the units due to scheduled maintenance. As we will see in the in the next paragraph, this limitation is not encountered in the method proposed by Yokoyama et al. [104], in which the problem structure does not allow any kind of coupling constraint, hence unlocking the possibility to slice the operation subproblem in many smaller ones.

The Yokoyama decomposition

The additional restricting assumptions on the problem structure advanced by Yokoyama et al. [104] with respect to the general formulation reported in (3.1) are the followings:

- There are no continuous design variables: $n_x = 0$;

- There are no coupling constraints h.

The resulting reference formulation is then:

$$\begin{aligned}
\min \quad & f_D\Big(z_D, y_D\Big) + \sum f_{O_t}\Big(z_{O_t}, y_{O_t}, x_{O_t}\Big) \\
\text{subjected to} \quad & g_t\Big(z_D, y_D, z_{O_t}, y_{O_t}, x_{O_t}\Big) \leq 0 \qquad \forall t \in \{1, \dots, T\} \\
& z_D \in \{0,1\}^{n_z} \\
& y_D \in \mathbb{Z}^{n_y} \\
& z_{O_t} \in \{0,1\}^{m_z} \qquad \forall t \in \{1, \dots, T\} \\
& y_{O_t} \in \mathbb{Z}^{m_y} \qquad \forall t \in \{1, \dots, T\} \\
& x_{O_t} \in \mathbb{R}^{m_x} \qquad \forall t \in \{1, \dots, T\}
\end{aligned} \tag{3.7}$$

As already anticipated, the strong simplification of discarding both continuous variables and coupling constraints allowed the authors to divide the single operation subproblem into a series of T lighter and independent ones. Moreover, the integer design variables y_D are restored in the upper level, allowing the modelling of discrete equipment capacities. By following a hierarchical logic very similar to that illustrated in section 3.1.1, the master problem of the decomposed formulation is

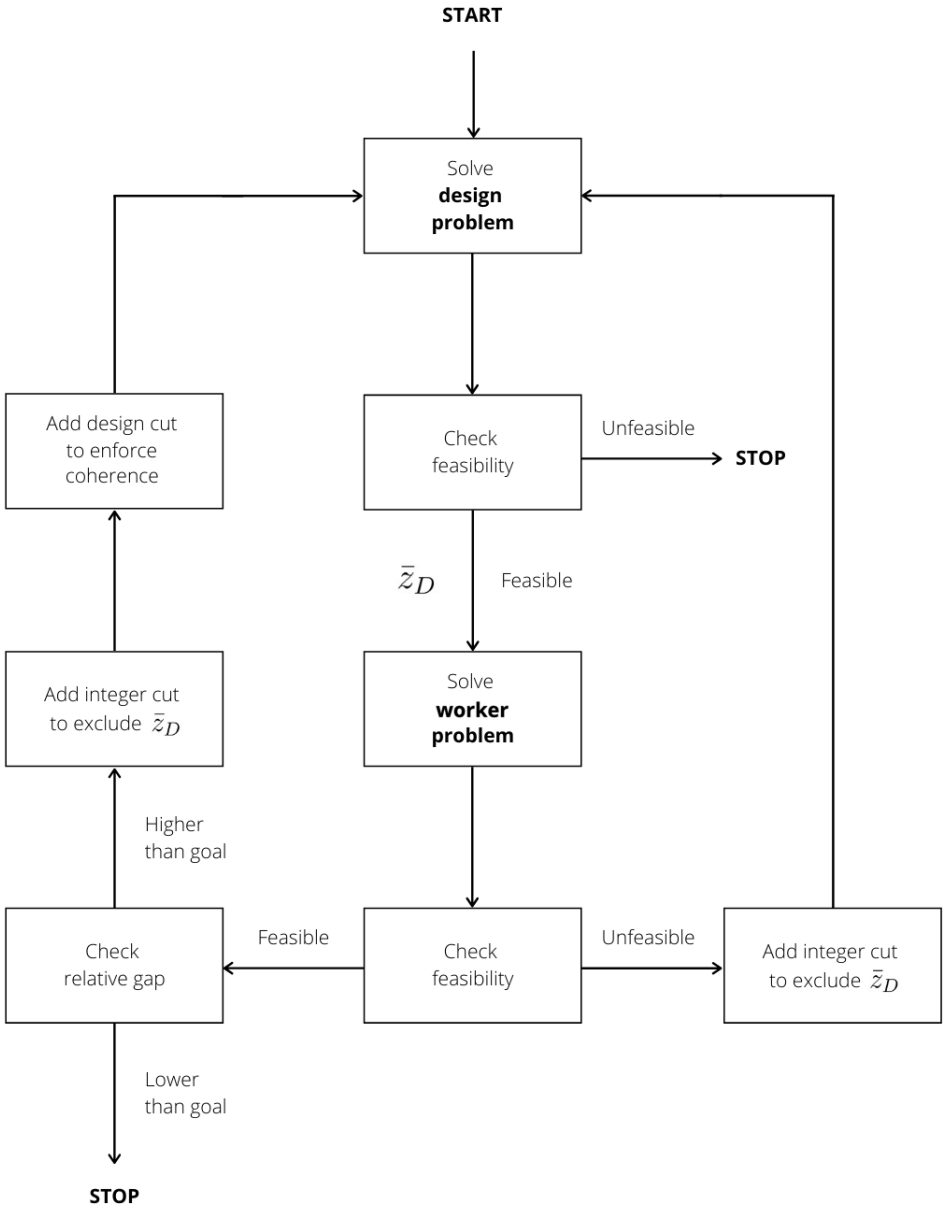


Figure 3.2: The decomposition algorithm by Iyer and Grossmann [102]

simply obtained from the original problem by relaxing all the operation variables:

$$\begin{aligned}
\min \quad & f_D\Big(z_D, y_D\Big) + \sum f_{O_t}\Big(z_{O_t}, y_{O_t}, x_{O_t}\Big) \\
\text{subjected to} \quad & g_t\Big(z_D, y_D, z_{O_t}, y_{O_t}, x_{O_t}\Big) \leq 0 \qquad \forall t \in \{1, \dots, T\} \\
& z_D \in \{0,1\}^{n_z} \\
& y_D \in \mathbb{Z}^{n_y} \\
& z_{O_t} \in \{0,1\}^{m_z} \qquad \forall t \in \{1, \dots, T\} \\
& y_{O_t} \in \mathbb{Z}^{m_y} \qquad \forall t \in \{1, \dots, T\} \\
& x_{O_t} \in \mathbb{R}^{m_x} \qquad \forall t \in \{1, \dots, T\}
\end{aligned} \tag{3.8}$$

while the k-th worker problem is defined as:

$$\begin{aligned}
\min \quad & f_D\Big(z_D, y_D\Big) + f_{O_k}\Big(z_{O_k}, y_{O_k}, x_{O_k}\Big) \\
\text{subjected to} \quad & g_k\Big(z_D, y_D, z_{O_k}, y_{O_k}, x_{O_k}\Big) \leq 0 \\
& z_D \in \{0,1\}^{n_z} \\
& y_D \in \mathbb{Z}^{n_y} \\
& z_{O_k} \in \{0,1\}^{m_z} \qquad \forall k \in \{1, \dots, T\} \\
& y_{O_k} \in \mathbb{Z}^{m_y} \\
& x_{O_k} \in \mathbb{R}^{m_x} \\
& z_D = \bar{z}_D \\
& y_D = \bar{y}_D
\end{aligned} \tag{3.9}$$

where $(\bar{z}_D, \bar{y}_D)$ is the integer design candidate found at the entrance node in the upper level. It is worth stressing that the possibility to consider only the k-th constraint g_k and objective term f_{O_k} among all the T possible ones in the definition of each worker problem wouldn't be possible in presence of a coupling constraint h or of a coupling variable x_D. Nevertheless, by giving up coupling variables and constraints, the authors opened to a more efficient definition of the subtrees in the lower level with respect to the approach by Iyer and Grossmann, by avoiding the definition of a complete operation subproblem over the whole time horizon.

Moreover, this choice allows to generate an initial set of valid lower bounds for f_D and for all the f_{O_t} by solving some preliminary *critical problems* before starting

with the actual decomposition algorithm. These additional problems are defined starting from problem 3.9 by removing the constraints $z_D = \bar{z}_D$ and $y_D = \bar{y}_D$ and by optimizing each single term of the objective function on its own:

$$\begin{aligned} \min \quad & f_D\Big(z_D, y_D\Big) \\ \text{subjected to} \quad & g_k\Big(z_D, y_D, z_{O_k}, y_{O_k}, x_{O_k}\Big) \leq 0 \\ & z_D \in \{0,1\}^{n_z} \\ & y_D \in \mathbb{Z}^{n_y} \qquad\qquad \forall k \in \{1,\dots,T\} \\ & z_{O_k} \in \{0,1\}^{m_z} \\ & y_{O_k} \in \mathbb{Z}^{m_y} \\ & x_{O_k} \in \mathbb{R}^{m_x} \end{aligned} \tag{3.10}$$

$$\begin{aligned} \min \quad & f_{O_k}\Big(z_{O_k}, y_{O_k}, x_{O_k}\Big) \\ \text{subjected to} \quad & g_k\Big(z_D, y_D, z_{O_k}, y_{O_k}, x_{O_k}\Big) \leq 0 \\ & z_D \in \{0,1\}^{n_z} \\ & y_D \in \mathbb{Z}^{n_y} \qquad\qquad \forall k \in \{1,\dots,T\} \\ & z_{O_k} \in \{0,1\}^{m_z} \\ & y_{O_k} \in \mathbb{Z}^{m_y} \\ & x_{O_k} \in \mathbb{R}^{m_x} \end{aligned} \tag{3.11}$$

All the T solutions $\underset{\sim}{f}^{CO}_{O_t}$ of the *critical operation problems* 3.11 are stored and used as valid lower bounds for f_{O_t} while, as far as the *critical design problems* 3.10 are concerned, only the highest among the T solutions $\underset{\sim}{f}^{CD}_{D}$ is stored as a valid lower bound for f_D.

The price to be paid with respect to the decomposition by Iyer and Grossmann is uncompatibility of the method with MES models characterised by continuous unit capacities or involving an inherent coupling between operation variables across different time-steps, such as problems with seasonal storage elements and/or with generating units affected by non-negligible memory effects (e.g. ramp-up limits, minimum down time and maximum up time, start-up and shut-down penalties etc.).

However, the numerical implementation of the problem is more elegant than that proposed by Iyer and Grossmann in 1998, mainly due to the significant advancements in the capabilities of commercial solvers in managing a fine tuning of the search algorithm by means of advanced software functionalities such as the *callback functions*[3] [105]. Unlike Iyer and Grossmann, who simply re-start the same MILP problem by iteratively adding integer cuts after each subproblem in order to exclude all the integer design candidates already explored from being generated again, Yokoyama and al. take advantage of the *legacy callbacks* of the CPLEX solver to reject all the incumbents found in the upper level by problem 3.8. In this way, the upper bound of the master problem is never actually updated in CPLEX and the search procedure in the upper level never interrupted. Of course, keeping track of the actual cutoff value becomes a task of the software program written to implement the decomposition algorithm.

A more detailed description of the decomposition algorithm proposed in [104] (see fig. 3.3) is reported below (CPLEX is used as the reference solver):

1. The critical design and operation problems are solved to obtain the lower bounds $\underset{\sim}{f}_D^{CD}$ and $\underset{\sim}{f}_{O_t}^{CO}$;

2. The cutoff value $\tilde{f}$ is initialized to a sufficiently high value. The order in which the T worker problems are solved in the lower level (here defined the *worker problems' queue*) is arbitrarily set and the B&B binary tree search of problem 3.8 is started;

3. At each node of the upper level, the *user cut callback* [110] is called and the following lower bound is built starting from the information coming from both the critical problems and the LP relaxation of the master problem at the current node:

$$\underset{\sim}{f} = \max\left(\underset{\sim}{f}_D^*, \underset{\sim}{f}_D^{CD},\right) + \sum \max\left(\underset{\sim}{f}_{O_t}^*, \underset{\sim}{f}_{O_t}^{CO}\right) \tag{3.12}$$

[3]From the IBM website: "callbacks allow you to monitor closely and to guide the behavior of CPLEX optimizers. In particular, callbacks allow user code to be executed regularly during an optimization or during a tuning session."

where $\underset{\sim}{f}^*_D$ and $\underset{\sim}{f}^*_{O_t}$ are the value of f_D and f_{O_t} from the solution of the LP relaxation at the current node. This lower bound is compared with the current cutoff $\tilde{f}$: if it is higher, then the node is fathomed by adding a suitable local user cut, otherwise the algorithm exits from the callback and resumes the B&B search in the upper level;

4. Each time an incumbent[4] $(\bar{z}_D, \bar{y}_D)$ is found in the upper level (that is, each time an *entrance node* is reached in the upper tree), it is rejected by means of the *incumbent callback* [108]. The *lazy constraint callback* is called immediately after, thus entering the lower level;

5. When in the lower level, the objective function associated with the solution of the LP problem of the entrance node is used together with the solutions of the critical problems to build a valid lower bound $\underset{\sim}{f}$ for the objective value f of the original problem:

$$\underset{\sim}{f} = f_D(\bar{z}_D, \bar{y}_D) + \sum \max\left(\underset{\sim}{\bar{f}}^*_{O_t}, \underset{\sim}{f}^{CO}_{O_t}\right) \tag{3.13}$$

where $\underset{\sim}{\bar{f}}^*_{O_t}$ is the value of f_{O_t} provided by the LP relaxation of the master problem at the entrance node. This lower bound is compared with the current cutoff $\tilde{f}$: if it is higher, then the lazy constraint callback is exited and the tree exploration in the upper level resumed, otherwise the solution algorithm sets up and solves the first worker problem in the queue;

6. After solving each worker problem, the lower bound provided by eq. (3.13) is updated as follows:

$$\underset{\sim}{f} = f_D(\bar{z}_D, \bar{y}_D) + \sum_{\text{solved}} \bar{f}_{O_t} + \sum_{\text{unsolved}} \max\left(\underset{\sim}{\bar{f}}^*_{O_t}, \underset{\sim}{f}^{CO}_{O_t}\right) \tag{3.14}$$

where the second term is the summation of the solutions $\bar{f}_{O_t}$ of the worker problems already solved in the current lower level, while the other summation is the same as in eq. (3.13) but limited to the indexes of the worker problems still unsolved. This lower bound is compared with the current cutoff $\tilde{f}$: if it is

[4]It is worth noting that an incumbent of 3.8 is an integer combination $(\bar{z}_D, \bar{y}_D)$ of the design variables. For this reason, this is also called a *design candidate*.

higher, then the algorithm immediately exits from the lower level, otherwise the solution algorithm sets up and solves the next worker problem in the queue;

7. When all the worker problems in the queue are succesfully solved, then an integer feasible solution of the original problem has been found. As a consequence, eq. (3.14) is used to compute the value of the associated objective function, which is then compared with the current cutoff: if it is lower, then it is used to update the problem's cutoff by means of a suitable lazy constraint, otherwise the algorithm immediately exits the lower level. In any case, the B&B search in the upper level is resumed from where it left off.

The above search algorithm stops when the relative gap falls below the user-defined value[5]. In order to provide a comprehensive overview of the method reported in [104], we still need to specify how the queue of worker problems is updated during the search. In brief, this is done by computing the difference $\bar{f}_{O_t} - \max\left(\underset{\sim}{\bar{f}}^*_{O_t}, \underset{\sim}{f}^{CO}_{O_t}\right)$ after solving each t-th worker problem and by storing it in a list Δ_t dedicated to that specific subproblem. Every time the algorithm enters the lower level, the mean value of each list Δ_t is computed: this value represents the average increase in $\underset{\sim}{f}$ related to the solution of the t-th worker problem. As such, this can be intended as the expected increase of $\underset{\sim}{f}$ found by solving the upcoming t-th worker problem. As a consequence, the queue of worker problems is updated by prioritizing those problems related to an index t corresponding to the highest mean value of Δ_t, since these are those who will most likely provide the highest increase of $\underset{\sim}{f}$, thus eventually leading to a faster proof of suboptimality.

[5]It is worth mentioning that, since all the incumbents found in the upper level are rejected by the algorithm, the relative gap must be manually computed in the user cut callback called at each node of the upper level by comparing the current cutoff with the current best bound provided by CPLEX.

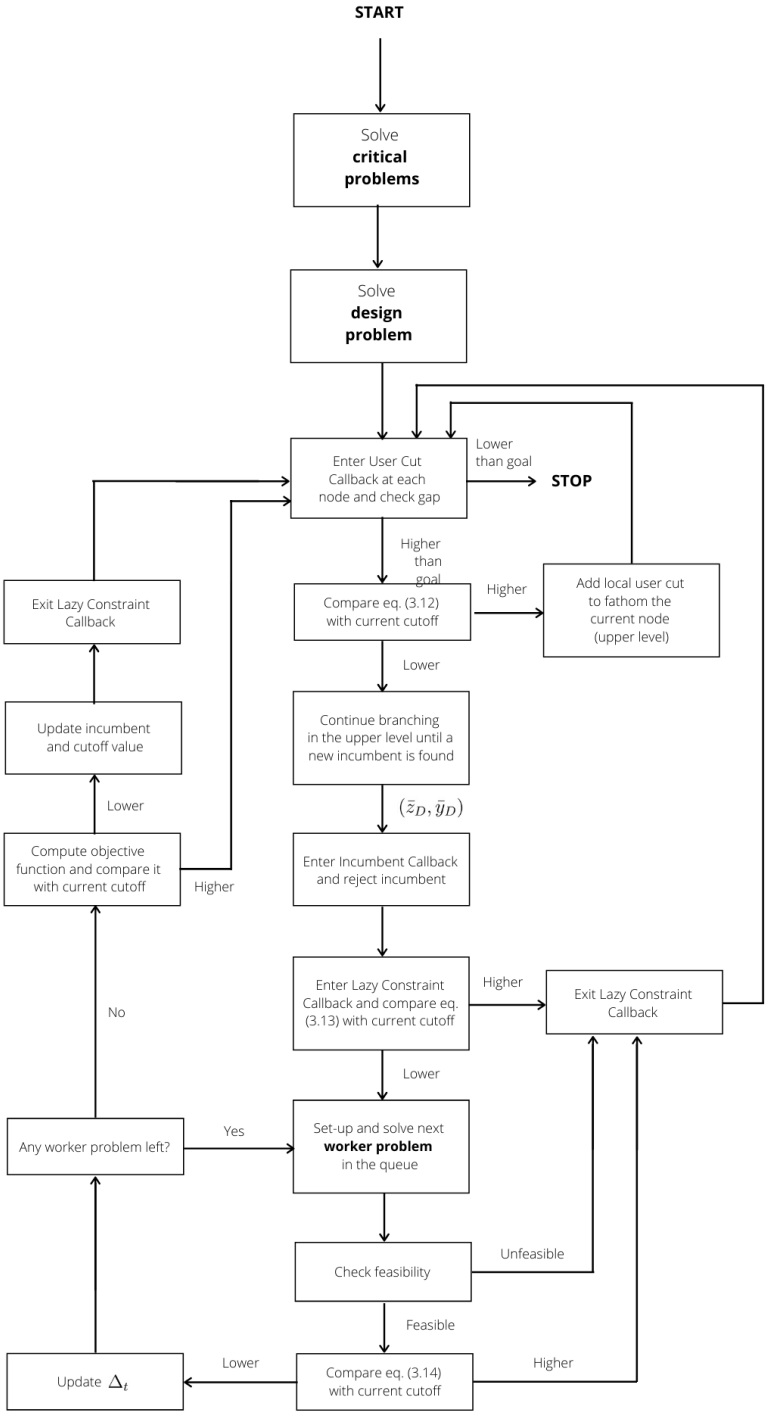


Figure 3.3: The decomposition algorithm by Yokoyama et al. [104]

How decomposition affects MES modelling: a comparison between the two methods

In the light of the information given in the previous sections, it is possible to compare the decomposition methods by Iyer and Grossmann [102] and by Yokoyama et al. [104] on the basis of four main features related to the specific assumptions on the MES model structure.

Feature	*Iyer and Grossmann*	*Yokoyama et al.*
Continuous design variables	Yes	No
Coupling constraints	Yes	No
Discrete capacities	No	Yes
Solve complete operation	Always	Never

In brief:

- The method by Iyer and Grossman can be applied to MES models containing time-coupling constraints and/or continuous equipment capacities, but it is not suitable for models with discrete sizes of the units. Moreover, it always solves the complete operation worker problem (defined on the whole time horizon), hence not exploiting at all the multi-period structure found in most MES models to reduce the computational time of the solver;

- The method by Yokoyama et al. can be applied to MES models with discrete equipment capacities, but it is not suitable for models with time-coupling constraints and/or with continuous design variables. On the other hand, unlike the method by Iyer and Grossmann, the operation subproblem is never solved as a whole, hence effectively exploiting the multi-period structure of the MES model to reduce the combinatorial complexity of the search tree.

It is clear that the two methods are perfectly complementary. As a consequence, MES models exhibiting a "hybrid" set of modelling features, such as a catalogue of generating units characterized by both discrete and continuous capacities, cannot be solved by any of the decomposition methods previously introduced.

3.2 Mathematical formulation of the novel hierarchical decomposition method for MES design optimization

In this section the mathematical formulation of the novel hierarchical decomposition method for the predictive design of complex MES architectures is presented. With respect to previous methods, the purpose of the proposed framework is threefold:

1. **Universality**: being applicable to any MES problem having the general form reported in eq. (3.1);

2. **Performance**: being faster than a general-purpose search algorithm in finding the global optimum;

3. **Usability**: being easy to implement without advanced programming skills.

3.2.1 Definition of the general, master and worker problems

As Iyer and Grossmann clearly state in their paper [102], in presence of time-coupling constraints and/or variables, the solution of a worker problem defined on the whole time horizon of the original problem is necessary each time a design candidate is found at the upper level. This is required to maintain a coherence between the two levels. As a consequence, in order to mantain a high level of generality of the algorithm, the general master and worker problems adopted in the proposed method are those defined in section 3.1.1 and below reported for the

reader's convenience:

$$\textbf{General problem} \quad \begin{aligned} \min \quad & f\Big(z_D, y_D, x_D, z_O, y_O, x_O\Big) \\ \text{subjected to} \quad & l\Big(z_D, y_D, x_D, z_O, y_O, x_O\Big) \leq 0 \\ & z_D \in \{0,1\}^{n_z} \\ & y_D \in \mathbb{Z}^{n_y} \\ & x_D \in \mathbb{R}^{n_x} \\ & z_O \in \{0,1\}^{m_z} \\ & y_O \in \mathbb{Z}^{m_y} \\ & x_O \in \mathbb{R}^{m_x} \end{aligned} \tag{3.15}$$

$$\textbf{Master problem} \quad \begin{aligned} \min \quad & f\Big(z_D, y_D, x_D, z_O, y_O, x_O\Big) \\ \text{subjected to} \quad & l\Big(z_D, y_D, x_D, z_O, y_O, x_O\Big) \leq 0 \\ & z_D \in \{0,1\}^{n_z} \\ & y_D \in \mathbb{Z}^{n_y} \\ & x_D \in \mathbb{R}^{n_x} \\ & z_O \in \mathbb{R}^{m_z} \\ & y_O \in \mathbb{R}^{m_y} \\ & x_O \in \mathbb{R}^{m_x} \end{aligned} \tag{3.16}$$

$$\textbf{Worker problem} \quad \begin{aligned} \min \quad & f\Big(z_D, y_D, x_D, z_O, y_O, x_O\Big) \\ \text{subjected to} \quad & l\Big(z_D, y_D, x_D, z_O, y_O, x_O\Big) \leq 0 \\ & z_D \in \{0,1\}^{n_z} \\ & y_D \in \mathbb{Z}^{n_y} \\ & x_D \in \mathbb{R}^{n_x} \\ & z_O \in \{0,1\}^{m_z} \\ & y_O \in \mathbb{Z}^{m_y} \\ & x_O \in \mathbb{R}^{m_x} \\ & y_D = \bar{y}_D \\ & z_D = \bar{z}_D \end{aligned} \tag{3.17}$$

where, as always, $(\bar{z}_D, \bar{y}_D)$ represents the integer design candidate associated with a feasible solution (an incumbent) of the master problem (3.16). Without loss of generality, we assume that the time horizon is composed of N typical periods of T time steps each[6] (e.g. 4 typical days of 24 hours each or 2 typical weeks of 168 hours each) and we make explicit the coupling constraints h from the uncoupled constraints g_n. Moreover, we choose to rewrite the objective function f as the sum of $2 + N$ terms, each containing only certain types of variables. It is worth noting that this can always be done due to the linearity of f with respect to the problem variables. The resulting general, master and worker problems are:

General problem

$$
\begin{aligned}
\min \quad & f_D\Big(z_D, y_D\Big) + f_{D^*}\Big(x_D\Big) + \sum f_{O_n}\Big(z_{O_n}, y_{O_n}, x_{O_n}\Big) \\
\text{subjected to} \quad & g_n\Big(z_D, y_D, x_D, z_{O_n}, y_{O_n}, x_{O_n}\Big) \leq 0 \quad \forall n \in \{1, \ldots, N\} \\
& h\Big(z_D, y_D, x_D, z_{O_1}, y_{O_1}, x_{O_1}, ..., z_{O_N}, y_{O_N}, x_{O_N}\Big) \leq 0 \\
& z_D \in \{0,1\}^{n_z} \\
& y_D \in \mathbb{Z}^{n_y} \\
& x_D \in \mathbb{R}^{n_x} \\
& z_{O_n} \in \{0,1\}^{m_z} \quad \forall n \in \{1, \ldots, N\} \\
& y_{O_n} \in \mathbb{Z}^{m_y} \quad \forall n \in \{1, \ldots, N\} \\
& x_{O_n} \in \mathbb{R}^{m_x} \quad \forall n \in \{1, \ldots, N\}
\end{aligned}
\tag{3.18}
$$

[6]Such indexing choice is particularly convenient to impose periodic constraints (e.g. the state of charge of a storage element at the end of each typical period must be higher than that at the beginning of the same period.)

Master problem

$$
\begin{aligned}
\min \quad & f_D\Big(z_D, y_D\Big) + f_{D^*}\Big(x_D\Big) + \sum f_{O_n}\Big(z_{O_n}, y_{O_n}, x_{O_n}\Big) \\
\text{subjected to} \quad & g_n\Big(z_D, y_D, x_D, z_{O_n}, y_{O_n}, x_{O_n}\Big) \leq 0 \quad \forall n \in \{1, \ldots, N\} \\
& h\Big(z_D, y_D, x_D, z_{O_1}, y_{O_1}, x_{O_1}, ..., z_{O_N}, y_{O_N}, x_{O_N}\Big) \leq 0 \\
& z_D \in \{0,1\}^{n_z} \\
& y_D \in \mathbb{Z}^{n_y} \\
& x_D \in \mathbb{R}^{n_x} \\
& z_{O_n} \in \mathbb{R}^{m_z} \quad \forall n \in \{1, \ldots, N\} \\
& y_{O_n} \in \mathbb{R}^{m_y} \quad \forall n \in \{1, \ldots, N\} \\
& x_{O_n} \in \mathbb{R}^{m_x} \quad \forall n \in \{1, \ldots, N\}
\end{aligned}
\tag{3.19}
$$

Worker problem

$$
\begin{aligned}
\min \quad & f_D\Big(z_D, y_D\Big) + f_{D^*}\Big(x_D\Big) + \sum f_{O_n}\Big(z_{O_n}, y_{O_n}, x_{O_n}\Big) \\
\text{subjected to} \quad & g_n\Big(z_D, y_D, x_D, z_{O_n}, y_{O_n}, x_{O_n}\Big) \leq 0 \quad \forall n \in \{1, \ldots, N\} \\
& h\Big(z_D, y_D, x_D, z_{O_1}, y_{O_1}, x_{O_1}, ..., z_{O_N}, y_{O_N}, x_{O_N}\Big) \leq 0 \\
& z_D \in \{0,1\}^{n_z} \\
& y_D \in \mathbb{Z}^{n_y} \\
& x_D \in \mathbb{R}^{n_x} \\
& z_{O_n} \in \{0,1\}^{m_z} \quad \forall n \in \{1, \ldots, N\} \\
& y_{O_n} \in \mathbb{Z}^{m_y} \quad \forall n \in \{1, \ldots, N\} \\
& x_{O_n} \in \mathbb{R}^{m_x} \quad \forall n \in \{1, \ldots, N\} \\
& y_D = \bar{y}_D \\
& z_D = \bar{z}_D
\end{aligned}
\tag{3.20}
$$

with m_z, m_y and m_x all higher than or equal to T.

By directly implementing the above decomposition scheme without any further expedient, one would find the equivalent method for the implementation of the branching priorities described in section 3.1.1. In this case, by solving the worker problem we would obtain an information with highest value: either we find that

the solution is infeasible/suboptimal or we find a new best cut-off. Nonetheless, the computational effort required to extract such information would be the highest as well, since an extensive exploration of the subtree in the lower level is required each time a design candidate is found in the upper level. This can become a non-negligible burden, in particular if the time horizon of the UC problem is significantly extended: the longer the time horizon, the more severe the impact on the computational time due to the exponential complexity of the subtree to be explored. To support this, it is worth recalling that the approach by Yokoyama and al. introduced in section 3.1.2 aims precisely at avoiding the complete resolution of problem (3.20) by giving up coupling variables and constraints in the model formulation.

In the following sections we propose an alternative decomposition paradigm aiming at avoiding the resolution of the worker problem for each design candidate $(\bar{z}_D, \bar{y}_D)$ without adding any restricting assumption on the general problem structure reported in eq. (3.18), thus ensuring the universality of the method.

3.2.2 Basic concept

The basic idea behind the novel hierarchical method lies in the concept of *local auxiliary problem* $\bar{P}'$. In general, we define as *auxiliary problem* a modified version of an *original problem* P which is easier to solve and that provides a valid lower bound for the objective function of P (or for a part of it). Following this definition, an auxiliary problem differs from a worker problem in that it is completely optional (that is, it is not required to ensure the coherence between upper and lower level). In a decomposition method, an auxiliary problem is thus an *additional* subproblem that can be solved to provide useful cuts aimed at reducing the size of the combinatorial solution space, thus improving the overall computational performance of the algorithm. In the context of a hierarchical decomposition of a MES problem, a *local auxiliary problem* $\bar{P}'$ is defined as an auxiliary problem whose original problem $\bar{P}$ is a worker problem (3.20). On the other hand, a *global auxiliary problem* $\check{P}'$ is defined as an auxiliary problem whose original problem $\check{P}$ is the general problem (3.18). Being its original problem a relaxed version of the worker problem (3.20)

(due to the absence of constraints $z_D = \bar{z}_D$ and $y_D = \bar{y}_D$), a global auxiliary problem differs from a local one in that its solution does provide a valid lower bound for both the general and *all* the possible worker problems. Conversely, a local auxiliary problem provides a lower bound which is only valid for its original worker problem. The critical problems (3.10) and (3.11) used in the decomposition algorithm by Yokoyama et al. are an example of global auxiliary problems. In fact, they are optional (thus they are not worker problems) and their solutions are used to derive a set of global lower bounds $\underset{\sim}{f}_D^{CD}$ and $\underset{\sim}{f}_{O_t}^{CO}$ that are valid for *all* the worker problems encountered during the search.

An important property of local auxiliary problems is that, being derived by a restricted version of the general problem, their lower bounds are always higher than those provided by a corresponding global auxiliary problem. This property is not exploited by any of the decomposition methods previously discussed, and represents the main original feature of the novel decomposition method here proposed. In fact, by relying on a more effective generation of lower bounds in the lower level (thanks to the definition of suitable local auxiliary problems), we are able to overcome the trade-off between universality and performance that makes the two algorithms analyzed in section 3.1.2 complementary, thus going towards a highly performing decomposition method which is also universally applicable. In our case, local auxiliary problems are simplified operation subproblems obtained from a corresponding worker problem by removing one or more constraints. In the proposed decomposition method, they are specifically used to effectively prove the sub-optimality or infeasibilty of a given integer design candidate $(\bar{z}_D, \bar{y}_D)$ before solving the associated complete worker problem (3.20).

3.2.3 Definition of the auxiliary problems

Following the search scheme reported in figure 3.1, when the hierarchical B&B search algorithm reaches the entrance node, the integer design candidate $(\bar{z}_D, \bar{y}_D)$ (and so the value of $\bar{f}_D$) are fixed. At this point, instead of solving the corresponding worker problem on the full time horizon and compute the optimized value of the

remaining part of f, it is possible to define smaller auxiliary problems to separately bound the contributions f_{D^*} and f_{O_n} given by the coupling variables x_D and by the MES operation in the different typical periods. Being additional and optional, auxiliary problems must have two main characteristics:

1. They must be very fast to solve;

2. They must provide valuable information to speed up the B&B search algorithm (e.g. by generating useful integer cuts for the master problem).

The aim is to find the best trade-off between the computational time required to solve the additional subproblems and the reduction of combinatioral complexity obtained through the information extracted from each auxiliary problem.

In general, for a MES problem, we can classify auxiliary problems on the basis of four main *constraining features* here defined:

Fixed integer design *All the integer design variables of the master problem are fixed to a certain integer value.* It allows to reduce the number of integer variables to branch and so the combinatorial complexity of the search tree. This is an inevitable condition for all the worker problems defined in the lower level, where the design candidate is set at the entrance node and represents the actual difference between local and global auxiliary problems.

Continuity *The problem is solved on the whole time horizon.* This characteristics is required if we want to obtain significant information about the contribution given to the objective function by design variables, which need to assume a unique value on the whole time horizon to maintain a physical meaning. We can say they must respect a "continuity constraint". Nonetheless, both global and local auxiliary problems used to bound the objective function of the worker problems may take advantage of the subdivision in typical periods to solve problems defined only on a single period, significantly reducing the number of integer variables of the problem and thus its complexity.

Integrity *The problem is solved maintaining all the integrity constraints active.* This characteristic allows to obtain a value of the objective function well-representing the one of the master problem, but can be removed on certain variables of the worker problem to simplify the search tree and compute lower bounds of the objective function very quickly.

Trade-off *The problem is solved optimizing the whole objective function.* Of course, in general all the different part of the objective function are inevitably interconnected, since design decisions influence the optimal operation strategy. In order to extract precise information about a specific term of f, auxiliary problems can be defined giving up the trade-off and optimizing only a single part of the objective function.

A local auxiliary problem can be properly defined starting from a worker problem by keeping the *fixed integer design* constraint feature and by dropping one or more between *continuity*, *integrity* and *trade-off*. In fact, by also dropping the *fixed integer design* constraining feature we obtain a global auxiliary problem. It must be stressed out that the only way to obtain valid lower bounds for each term of f *separately* is by dropping the trade-off constraining feature (that is, by defining local auxiliary problems in which each of the terms f_D, f_{D^*} and f_{O_n} is optimized on its own). This is a necessary requirement to effectively exploit the multi-period structure of the problem and eventually avoid the solution of the UC on the whole time horizon for a given integer design candidate.

Global auxiliary problems for f_D, f_{D^*} and f_{O_n}

The global auxiliary problems (GAP) set up to obtain a set of valid global lower bounds for f_D, f_{D^*} and $f_{O_n}$are defined starting from the general problem (3.18) by dropping the trade-off and continuity constraining features and by neglecting the coupling constraints h. We have then:

$$\textbf{GAP for } f_D \qquad \begin{aligned} \min \quad & f_D\Big(z_D, y_D\Big) \\ \text{subjected to} \quad & g_n\Big(z_D, y_D, x_D, z_{O_n}, y_{O_n}, x_{O_n}\Big) \leq 0 \\ & z_D \in \{0,1\}^{n_z} \\ & y_D \in \mathbb{Z}^{n_y} \\ & x_D \in \mathbb{R}^{n_x} \\ & z_{O_n} \in \{0,1\}^{m_z} \\ & y_{O_n} \in \mathbb{Z}^{m_y} \\ & x_{O_n} \in \mathbb{R}^{m_x} \end{aligned} \qquad \forall n \in \{1, ..., N\} \tag{3.21}$$

$$\textbf{GAP for } f_{D^*} \qquad \begin{aligned} \min \quad & f_{D^*}\Big(x_D\Big) \\ \text{subjected to} \quad & g_n\Big(z_D, y_D, x_D, z_{O_n}, y_{O_n}, x_{O_n}\Big) \leq 0 \\ & z_D \in \{0,1\}^{n_z} \\ & y_D \in \mathbb{Z}^{n_y} \\ & x_D \in \mathbb{R}^{n_x} \\ & z_{O_n} \in \{0,1\}^{m_z} \\ & y_{O_n} \in \mathbb{Z}^{m_y} \\ & x_{O_n} \in \mathbb{R}^{m_x} \end{aligned} \qquad \forall n \in \{1, ..., N\} \tag{3.22}$$

$$\textbf{GAP for } f_{O_n} \qquad \begin{aligned} \min \quad & f_{O_n}\Big(z_{O_n}, y_{O_n}, x_{O_n}\Big) \\ \text{subjected to} \quad & g_n\Big(z_D, y_D, x_D, z_{O_n}, y_{O_n}, x_{O_n}\Big) \leq 0 \\ & z_D \in \{0,1\}^{n_z} \\ & y_D \in \mathbb{Z}^{n_y} \\ & x_D \in \mathbb{R}^{n_x} \\ & z_{O_n} \in \{0,1\}^{m_z} \\ & y_{O_n} \in \mathbb{Z}^{m_y} \\ & x_{O_n} \in \mathbb{R}^{m_x} \end{aligned} \qquad \forall n \in \{1, ..., N\} \tag{3.23}$$

Both the values of f_D and f_{D^*} are optimized N times (once for each typical period). Hence, since we are looking for a lower bound, only the highest among all the solutions to problems (3.21) and (3.22) are considered. Being completely independent of the specific values of z_D and y_D, all these problems can be solved before starting the actual B&B search. Moreover, since they are obtained from a relaxed version of the general problem (3.18), their solutions provide a set of global lower bounds $\check{\underset{\sim}{f}}_D$, $\check{\underset{\sim}{f}}_{D^*}$ and $\check{\underset{\sim}{f}}_{O_n}$ for f_D, f_{D^*} and f_{O_n} valid both in the upper and in the lower level that can be used to compute the following lower bounds for f and $\bar{f}$:

$$\underset{\sim}{f}^{\circ} := \check{\underset{\sim}{f}}_D + \check{\underset{\sim}{f}}_{D^*} + \sum \check{\underset{\sim}{f}}_{O_n} \leq f \tag{3.24}$$

$$\bar{\underset{\sim}{f}}^{\circ} := \bar{f}_D + \check{\underset{\sim}{f}}_{D^*} + \sum \check{\underset{\sim}{f}}_{O_n} \leq \bar{f} \tag{3.25}$$

where $\bar{f}$ and $\bar{f}_D$ are the values of f and f_D associated with a given integer design candidate $(\bar{z}_D, \bar{y}_D)$.

Local auxiliary problem for f_{D^*}

The local auxiliary problem (LAP) set up to obtain a valid local lower bound for the f_{D^*} term of the objective function associated with a certain integer design candidate $(\bar{z}_D, \bar{y}_D)$ is defined starting from the worker problem (3.20) by dropping the trade-off and the integrity constraining features. We have then:

$$
\begin{aligned}
& & \min \quad & f_{D^*}\Big(x_D\Big) & & \\
& & \text{subjected to} \quad & g_n\Big(z_D, y_D, x_D, z_{O_n}, y_{O_n}, x_{O_n}\Big) \leq 0 & \quad & \forall n \in \{1, \ldots, N\} \\
& & & h\Big(z_D, y_D, x_D, z_{O_1}, y_{O_1}, x_{O_1}, ..., z_{O_N}, y_{O_N}, x_{O_N}\Big) \leq 0 & & \\
& & & z_D \in \{0,1\}^{n_z} & & \\
& & & y_D \in \mathbb{Z}^{n_y} & & \\
\textbf{LAP for } f_{D^*} & & & x_D \in \mathbb{R}^{n_x} & & \\
& & & z_{O_n} \in \mathbb{R}^{m_z} & & \forall n \in \{1, \ldots, N\} \\
& & & y_{O_n} \in \mathbb{R}^{m_y} & & \forall n \in \{1, \ldots, N\} \\
& & & x_{O_n} \in \mathbb{R}^{m_x} & & \forall n \in \{1, \ldots, N\} \\
& & & y_D = \bar{y}_D & & \\
& & & z_D = \bar{z}_D & &
\end{aligned}
\tag{3.26}
$$

Being dependent on the specific values of $\bar{z}_D$ and $\bar{y}_D$, these problems can only be solved in the lower level after each entrance node. Moreover, since they are obtained from a relaxed version of the worker problem (3.20), their solution provides a local lower bound $\underset{\sim}{\bar{f}}_{D^*}$ for f_{D^*} only valid in the subtree associated with the integer design candidate $(\bar{z}_D, \bar{y}_D)$.

It must be noticed that problem (3.26) is not a restricted version of the global auxiliary problem for f_{D^*}, since it contains both a restriction ($z_D = \bar{z}_D$, $y_D = \bar{y}_D$ and the coupling constraint h) and a relaxation ($z_{O_n} \in \mathbb{R}^{m_z}$, $y_{O_n} \in \mathbb{R}^{m_y}$ $\forall n \in \{1, \ldots, N\}$). As a consequence, $\underset{\sim}{\bar{f}}_{D^*}$ may be lower or higher than $\underset{\sim}{\check{f}}_{D^*}$, depending whether the prevailing effect is that of the restriction or that of the relaxation. In any case, we can write that:

$$
\max\left(\underset{\sim}{\check{f}}_{D^*}, \underset{\sim}{\bar{f}}_{D^*}\right) - \underset{\sim}{\check{f}}_{D^*} \geq 0 \tag{3.27}
$$

Local auxiliary problems for f_{O_n}

The local auxiliary problems (LAP) set up to obtain some valid local lower bounds for the f_{O_n} terms of the objective function associated with a certain integer design candidate $(\bar{z}_D, \bar{y}_D)$ are defined starting from the worker problem (3.20) by

dropping the trade-off and the continuity constraining features (that is, by defining one problem for each typical period) and by relaxing the coupling constraints h. We have then:

$$\textbf{LAP for } f_{O_n} \quad \begin{aligned} \min \quad & f_{O_n}\Big(z_{O_n}, y_{O_n}, x_{O_n}\Big) \\ \text{subjected to} \quad & g_n\Big(z_D, y_D, x_D, z_{O_n}, y_{O_n}, x_{O_n}\Big) \leq 0 \\ & z_D \in \{0,1\}^{n_z} \\ & y_D \in \mathbb{Z}^{n_y} \\ & x_D \in \mathbb{R}^{n_x} \\ & z_{O_n} \in \{0,1\}^{m_z} \\ & y_{O_n} \in \mathbb{Z}^{m_y} \\ & x_{O_n} \in \mathbb{R}^{m_x} \\ & y_D = \bar{y}_D \\ & z_D = \bar{z}_D \end{aligned} \qquad \forall n \in \{1, ..., N\} \tag{3.28}$$

Being dependent on the specific values of z_D and y_D, these problems can only be solved in the lower level after each entrance node. Moreover, since they are obtained from a relaxed version of the worker problem (3.20), their solution provides a local lower bound $\underset{\sim}{\bar{f}}_{O_n}$ for f_{O_n} only valid in the subtree associated with the integer design candidate $(\bar{z}_D, \bar{y}_D)$.

It must be noticed that problem (3.26) is actually a restricted version of the global auxiliary problem for f_{D^*}, since it can be obtained from it by adding the additional constraints $z_D = \bar{z}_D$ and $y_D = \bar{y}_D$. As a consequence, $\underset{\sim}{\bar{f}}_{O_n}$ is always higher than $\underset{\sim}{\check{f}}_{O_n}$:

$$\underset{\sim}{\bar{f}}_{O_n} - \underset{\sim}{\check{f}}_{O_n} \geq 0 \quad \forall n \in \{1, \ldots, N\} \tag{3.29}$$

At this point, thanks to the $N+1$ eqs. (3.27) and (3.29) and to eq. (3.25), we are able to write the following chain of inequalities providing $N+2$ local lower bounds for $\bar{f}$, in which the $N+1$ ones indicated as $\underset{\sim}{\bar{f}}^{(k)}$ (with $k \in \{1, ..., N+1\}$) are those obtained after solving k among the $N+1$ local auxiliary problems defined

in section 3.2.3 for a given integer design candidate:

$$\underset{\sim}{\bar{f}}^{\circ} \leq \underset{\sim}{\bar{f}}^{(1)} \leq \underset{\sim}{\bar{f}}^{(2)} \leq \cdots \leq \underset{\sim}{\bar{f}}^{(N+1)} \leq \bar{f} \tag{3.30}$$

where $\underset{\sim}{\bar{f}}^{(k)}$ is equal to

$$\underset{\sim}{\bar{f}}^{(k)} := \bar{f}_D + \max\left(\underset{\sim}{\check{f}}_{D^*}, \underset{\sim}{\bar{f}}_{D^*}\right) + \sum_{n \in U} \underset{\sim}{\check{f}}_{O_n} + \sum_{n \in S} \underset{\sim}{\bar{f}}_{O_n} \tag{3.31}$$

or to

$$\underset{\sim}{\bar{f}}^{(k)} := \bar{f}_D + \underset{\sim}{\check{f}}_{D^*} + \sum_{n \in U} \underset{\sim}{\check{f}}_{O_n} + \sum_{n \in S} \underset{\sim}{\bar{f}}_{O_n} \tag{3.32}$$

depending on whether the local auxiliary problem for f_{D^*} is among the k local auxiliary problems already solved or not. In the above definitions, S is the set of indexes of the k (or $k-1$) local auxiliary problems for f_{O_n} already solved, while U is the set of indexes of the $N-k$ (or $N-(k-1)$) local auxiliary problems for f_{O_n} still unsolved.

3.2.4 Description of the solution algorithm

The aim of the proposed algorithm is to effectively exploit the valuable information provided by the global and local auxiliary problems previosuly defined to periodically generate valid local integer cuts[7]. These cuts are used to prune the subtree rooted at the node in which they are generated, thus significantly reducing the total number of explored nodes both in the upper and in the lower level. A local integer cut is enforced in the upper level every time a valid lower bound $\underset{\sim}{f}$ for f exceeds the current cutoff value $\tilde{f}$. A local integer cut is enforced in the lower level every time a valid lower bound $\underset{\sim}{\bar{f}}$ for $\bar{f}$ exceeds the current cutoff value $\tilde{f}$. For our purposes, an integer cut enforced at a node j is a linear constraint of the form:

$$f < (f^*)_j \qquad \text{(valid locally)} \tag{3.33}$$

where $(f^*)_j$ is the solution of the LP relaxation of node j. In this way, node j is immediately fathomed due to violation of constraint (3.33), and the corresponding subtree pruned. In figure 3.4 it is reported an illustrative example showing how the

[7] An integer cut is *valid* if it only cuts out sub-optimal or infeasible integer solutions. An integer cut is *local* if it is only enforced for the subtree rooted at the node where the cut is generated.

knowledge of a global lower bound $\underset{\sim}{f}$ in a node in the upper level may determine a premature pruning of the corresponding subtree, hence completely avoiding the generation of an integer design candidate and the exploration of the associated subtree in the lower level.

In the following sections, the algorithmic details of the proposed decomposition method will be gradually introduced by separately analyzing its three main blocks: (i) the pre-processing stage, (ii) the upper level of the B&B tree and (iii) the lower level of the B&B tree.

1. Pre-processing stage

Being the global auxiliary problems independent of any type of information acquired during the B&B search of the upper and lower levels, they are actually solved once and for all in a preliminary phase (the so called *pre-processing stage*) to obtain and store the values of the global lower bounds $\check{\underset{\sim}{f}}_D$, $\check{\underset{\sim}{f}}_{D^*}$ and $\check{\underset{\sim}{f}}_{O_n}$. In this phase, the queue Q of the local auxiliary problems is initialized with an arbitrary order.

2. Solution strategy in the upper level

Let's consider a generic node j in the upper level (that is, belonging to the search tree of the master problem). By relying on the information deriving both from the global auxiliary problems and from the LP relaxation of the node it is possible to compute the following lower bound for any possible integer feasible solution associated with a design candidate generated in the subtree rooted at node j:

$$\underset{\sim}{f}_j = \max\left((f_D^*)_j, \check{\underset{\sim}{f}}_D\right) + \max\left((f_{D^*}^*)_j, \check{\underset{\sim}{f}}_{D^*}\right) + \sum \max\left((f_{O_n}^*)_j, \check{\underset{\sim}{f}}_{O_n}\right) \tag{3.34}$$

where $(f_D^*)_j$, $(f_{D^*}^*)_j$ and $(f_{O_n}^*)_j$ are the values of f_D, f_{D^*} and f_{O_n} evaluated for the solution of the LP relaxation of node j. Hence, a sub-optimality check is performed at each node j of the branching tree of the master problem by comparing $\underset{\sim}{f}_j$ with the current cutoff value $\tilde{f}$:

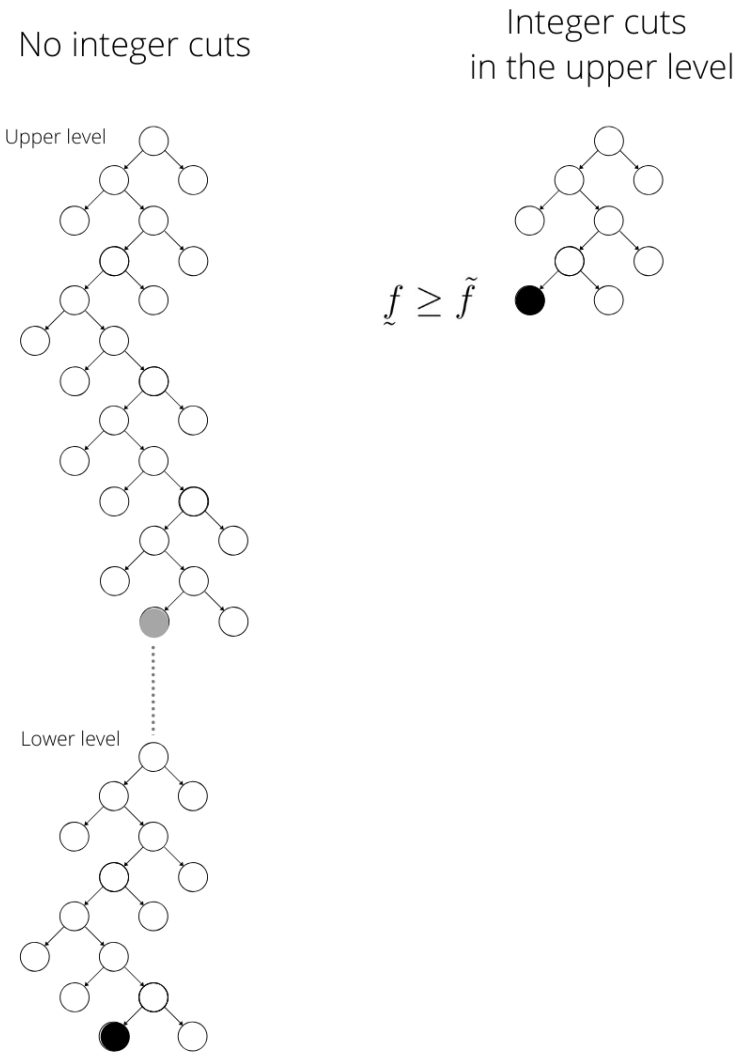


Figure 3.4: Comparison between branching trees: with (right) and without (left) integer cuts in the upper level. Black nodes are fathomed nodes, while the grey node represents an entrance node to the lower level.

- If $\underset{\sim}{f}_j < \tilde{f}$ the search of the current tree branch is continued by branching on node j;

- If $\underset{\sim}{f}_j \geq \tilde{f}$ the search of the current tree branch is interrupted by fathoming node j through a suitable integer cut.

In addition, an integrity test is always carried out on the values of $(z_D^*)_j$ and $(y_D^*)_j$: if all their elements have integer/binary values, then we have found an incumbent of the master problem and node j is labeled as an *entrance node*. In this case, the incumbent is immediately rejected and the lower level is entered with $\bar{z}_D = (z_D^*)_j$ and $\bar{y}_D = (y_D^*)_j$.

3. Solution strategy in the lower level

Each time an entrance node is found in the upper level, the lower level is entered with the values of z_D and y_D fixed and equal to those provided by the solution of the corresponding LP relaxation (that is, $\bar{z}_D$ and $\bar{y}_D$). As a consequence, we can immediately compute the associated values of $\bar{f}_D$ and $\underset{\sim}{\bar{f}}^\circ$ that can be immediately used to make a preliminary check on the sub-optimality and feasibility of the current integer design candidate. The very first check compares the value of $\bar{f}_D$ with its global lower bound $\underset{\sim}{\check{f}}_D$:

- If $\bar{f}_D < \underset{\sim}{\check{f}}_D$ we are sure that the current integer design candidate will eventually lead to an unfeasible solution. This is due to the fact that $\underset{\sim}{\check{f}}_D$ is a globally valid lower bound for f_D. In this case, the lower level is immediately exited without solving any auxiliary/worker problem and the search of a new integer design candidate in the upper level is resumed;

- If $\bar{f}_D \geq \underset{\sim}{\check{f}}_D$ then we can't conclude anything about the feasibility of $(\bar{z}_D, \bar{y}_D)$ and the search algorithm remains in the lower level.

A second check, comparing $\underset{\sim}{\bar{f}}^\circ$ with the current cutoff $\tilde{f}$, is the following:

- If $\underset{\sim}{\bar{f}}^\circ \geq \tilde{f}$ we can conclude that, since $\underset{\sim}{\bar{f}}^\circ \leq \bar{f}$, the current integer design candidate will surely lead to a sub-optimal solution: the lower level is exited

prematurely without solving any auxiliary/worker problem and the search of a new integer design candidate is resumed in the upper level;

- If $\underset{\sim}{\bar{f}}^{\circ} < \tilde{f}$ we can't conclude anything on the final value of $\bar{f}$: the algorithm sets up and solves the local auxiliary problem contained in the queue Q, starting from the first.

In the second case, since a valid lower bound $\underset{\sim}{\check{f}}_{O_n}$ is always available for the objective function $\underset{\sim}{\bar{f}}_{O_n}$ of the n-th local auxiliary subproblem for f_{O_n} (with $n = Q\{k\}$ individuated by the k-th index of the queue Q), the following valid upper bound can be computed before solving the corresponding problem:

$$\tilde{f}_{O_n} = \tilde{f} - \left(\underset{\sim}{\bar{f}}^{(k-1)} - \underset{\sim}{\check{f}}_{O_n}\right) \tag{3.35}$$

At this point the local auxiliary subproblem is solved and a feasibility check is carried out:

- If the local auxiliary problem is infeasible, we can then conclude that, being it a relaxed version of the worker problem, also the latter will be infeasible. In this case the lower level is exited prematurely without solving any further auxiliary/worker problem and the search of a new integer design candidate is resumed in the upper level;

- If the local auxiliary problem is feasible, then the new local lower bound $\underset{\sim}{\bar{f}}^{(k)}$ is computed with the new $\underset{\sim}{\bar{f}}_{O_n}$ or $\max\left(\underset{\sim}{\check{f}}_{D^*}, \underset{\sim}{\bar{f}}_{D^*}\right)$. At this point, a sub-optimality check is done:

  - If $\underset{\sim}{\bar{f}}^{(k)}$ is higher than the current cutoff value $\tilde{f}$, then the lower level is exited prematurely without solving any auxiliary/worker problem and the search of a new integer design candidate is resumed in the upper level;
  - If $\underset{\sim}{\bar{f}}^{(k)}$ is lower than the current cutoff value $\tilde{f}$, then the next auxiliary problem in the queue is set up and solved by the algorithm;

If all the $N+1$ local auxiliary problems are succesfully solved and the final value $\underset{\sim}{\bar{f}}^{(N+1)}$ of the local lower bound is still lower than the current cutoff $\tilde{f}$, then the

algorithm sets up and solves the worker problem to restore the coherence with the upper level. By means of a dedicated callback, the lower bound of this problem is continuously compared with the global cutoff at each node of the subtree: if it is higher, then the solution of the worker problem is aborted and the corresponding integer design candidate immediately discarded. A this point, as new cutoff value may be found or not depending on the feasibility and optimality of the solution of the worker problem. In any case, the algorithm resumes the search of a new integer design candidate in the upper level.

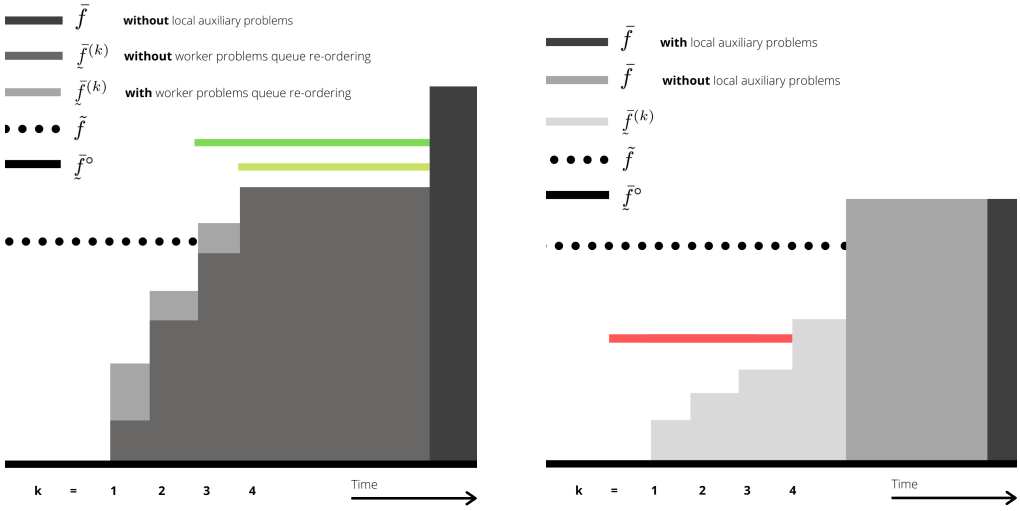
Worker problem queue reordering and further considerations

At this point, we still need to specify how the queue reordering is carried out during the solution procedure. Thanks to eq. (3.30) we are sure that each time a local auxiliary problem is solved in the lower level, it determines an increase $\underset{\sim}{\bar{f}}^{(k)} - \underset{\sim}{\bar{f}}^{(k-1)}$ or $\underset{\sim}{\bar{f}}^{(1)} - \underset{\sim}{\bar{f}}^{\circ}$ of the local lower bound for $\bar{f}$ which is equal to $\underset{\sim}{\bar{f}}_{O_n} - \underset{\sim}{\check{f}}_{O_n} \geq 0$ or to $\max\left(\underset{\sim}{\check{f}}_{D^*}, \underset{\sim}{\bar{f}}_{D^*}\right) - \underset{\sim}{\check{f}}_{D^*} \geq 0$, depending on the specifics type of auxiliary problem. Similarly to what is done with the worker problems of the algorithm by Yokoyama et al. [104], all these increases are stored in a list l_i indexed on the $N+1$ local auxiliary problems and are used to efficiently re-order the queue Q of local auxiliary problems each time the algorithm enters the lower level with a new integer design candidate. This is simply done by prioritizing the local auxiliary problems corresponding to the lists l_i characterised by the highest mean values of their stored increases. In this way, we are solving first those problems who will most likely provide the highest increase in $\underset{\sim}{\bar{f}}^{(k)}$, hence eventually leading to a faster proof of sub-optimality/infeasibility of the given integer design candidate. The beneficial effect of queue reordering is reported in figure 3.5, in which is shown the time evolution of the lower bound $\underset{\sim}{\bar{f}}^{(k)}$ during the solution procedure in the lower level for the two cases $\underset{\sim}{\bar{f}}^{(N+1)} > \tilde{f}$ (case a) and $\underset{\sim}{\bar{f}}^{(N+1)} < \tilde{f}$ (case b). In both cases the initial lower bound is $\underset{\sim}{\bar{f}}^{\circ}$ but, as evident in case (a), queue reordering allows to solve only 3 auxiliary problems instead of 4, hence determining a lower time spent in the lower level. When we are in case (a), solving the auxiliary problems always lead to a premature proof of sub-optimality with respect to a traditional decomposition

method directly solving the worker problem. Nevertheless, when we are in case (b) (that is, when $\underset{\sim}{\bar{f}}^{(N+1)} < \tilde{f}$), all the computational time spent in solving the local auxiliary problems is additional with respect to a reference case without any auxiliary problem (we are introducing an overhead). The same happens when the integer design candidate entering the lower level leads to an integer feasible solution. We name this situation as case (c). Hence, there exists a trade-off between the computational time saved in case (a) and the overhead introduced in cases (b) and (c). In this sense an important observation is that the increase in computational time associated with a higher number of local auxiliary problems associated to a higher number of typical periods is approximatively *linear* (we are growing the number of subproblems without varying the number of integer operation variables in each one of them[8]). At the same time, by increasing N, the computational time required to solve the complete worker may potentially explode due to the combinatorial complexity of the problem (by linearly increasing the number of integer operation variables of the problem we are exponentially growing the number of possible discrete solutions). Considering this, we can reasonably expect that, for problems defined over a sufficiently long time horizon, the time saved thanks to the additional integer cuts deriving from the local auxiliary problems will eventually prevail on the computational overhead introduced when $\underset{\sim}{\bar{f}}^{(N+1)} < \tilde{f}$, and that the highest the number of typical periods, the highest the benefit. The numerical results obtained in the case-study introduced in chapter 4 will help us prove this.

The table below compares the novel decomposition algorithm here introduced with the previous methods by Iyer and Grossman and by Yokoyama et al. analyzed at the beginning of this chapter. It is apparent that, mainly thanks to the introduction of the innovative concept of *local auxiliary problem*, we were able to extend the applicability of the algorithm to MES models containing both coupling constraints and mixed integer/continuous design variables without accepting the burden to

[8]Of course, we are also increasing the number of continuous operation variables x_{O_n}, but this only slows the solution time of the LP relaxations, without affecting the size of the combinatorial solution space.



(a) Case (a). The dark green and light green lines represent the time saved thanks to the local auxiliary problems with and without queue reordering, respectively.

(b) Case (b). The red line represent the overhead time spent solving auxiliary problems.

Figure 3.5: Graphical representation (qualitative) of the solution procedure in the lower level for an integer design candidate leading to a feasible but sub-optimal solution (4 typical periods)

systematically solve the complete working problem, hence overcoming the original trade-off between universality and performance that affected the two approaches.

Feature	*Iyer and Grossmann*	*Yokoyama et al.*	*This work*
Continuous design variables	Yes	No	Yes
Coupling constraints	Yes	No	Yes
Discrete capacities	No	Yes	Yes
Solve complete operation	Always	Never	Occasionally

Numerical implementation of the decomposition algorithm: main considerations

As stated at the beginning of the chapter, one of the three main purposes of the novel decomposition algorithm presented in this Thesis (beyond *universality* and

performance) is *usability*, which basically means that the numerical implementation of the decomposition method should be accessible to academics and professionals with basic programming skills. This has been possible by exploiting the high flexibility of the multi-language POLIMIP modelling environment introduced in section 2.2. In fact, the only additional information required by the user to implement the decomposition method starting from the algebraic formulation a general MES problem is a set of flags indicating the integer design and the integer operation variables. This is conveniently done by providing the additional options `set.design` and `set.intop` to the `dvar` function already called in the script dedicated to variable definition. This semplicity in the numerical integration of the model is not a common feature in MES decomposition methods. In fact, many efficient algorithms do not found practical applications due to an unacceptable complexity in their numerical implementation, which typically requires advanced programming skills in low-level programming languages such The algorithm by Yokoyama et al. [104] extensively analyzed in the previous sections is among them. In fact, as stated by Yokoyama himself at the end of his paper: "[...] to make the proposed method more practical, it will be inevitable to generate the input data and program automatically". This feature has been implemented in the POLIMIP modelling environment, which is able to automatically implement and choose between the method by Yokoyama and the novel method presented in this work depending on the absence/presence of continuous design variables and/or time-coupling constraints in the original problem structure. A further advancement with respect to the implementation proposed by Yokoyama et al. is the use of CPLEX's *generic callbacks* instead of *legacy callbacks* for the modification of the search algorithm of solver in consideration of the decomposition strategy. Generic callbacks are a recent feature of the CPLEX API (introduced in 2017 in version 2.18 [106]) as an improvement of legacy callbacks. In fact, with respect to the latters, they are compatible with many advanced solver features (such as dynamic search) and are characterised by *path invariance* [107]: "Using an empty generic callback will yield the same solution path as using no generic callback. (This convention is not true for CPLEX legacy callbacks.)".

Chapter 4

Assessment of the numerical performances of the novel decomposition algorithm on a real case study

4.1 Description and modelling of the Bovisa case study

In this section we will introduce the case study analysed to measure the performance of the novel decomposition algorithm presented in chap. 3. The problem regards the optimal design of a multi-energy system for the university campus of *Politecnico di Milano* called "Bovisa" in Northern Italy. The campus is the seat of 5 research departments and includes 7 major buildings plus several minor buildings. The MES must be designed to satisfy the demands of electricity, heat and cooling for all the buildings and the laboratories during the year. It is an on-grid system, so the electricity can be bought from the national grid. The system has been modelled to include both integer and continuous design variables, being therefore incompatible with both the methods by Iyer and Grossman [102] and by Yokoyama et al.[104].

4.1.1 Preliminary analysis of the microgrid architecture

The architecture of the system is presented in fig. 4.1. To satisfy the demand of electricity, heating and cooling of the university campus, the components that can be installed are:

Gas Turbine (GT) The system take advantages of cogenerative gas turbines, which are one input-two output components. They consume natural gas and produce simultaneously electricity and heating.

Auxiliary Boiler (AB) The available boilers are single input-single output components, consuming natural gas and producing heating power.

Electric Chiller (EC) This is one of the two technologies available to satisfy the cooling demand. Since they are based on a traditional thermodynamic cycle based on compression-expansion processes, they consume electric energy and produce cold water. The electricity required to run electric chillers can be either produced by means of the gas turbines or bought from the grid.

Absorpion Chiller (AC) By means of thermodynamic and chemical processes, absorption chillers allow to convert heat into refrigeration. Such a technology is very common in multi-energy microgrids, since it allows to recover heat power also during summer to feed the cooling system.

Each technology is available in the form of a catalogue that contains different models characterised by various performance parameters, sizes and costs. For each technology is possible to select only a single model. Yet, for each model we can install up to 4 units that can be operated separately. The optimization process will choose if it is more convenient to install few large units to take advantage of the improved conversion performances and economy of scale or if it is better to select a high number of small units to leverage on flexibility of operation and reduce the UC costs. The catalogue of the available models is reported in tab. 4.1. There are ten different available models of gas turbine and four models for each of the other component. It means that the design problem counts 88 potential equipment units that might be installed to optimize the microgrid. It means that, considering a

time horizon of 14 days, the model will contain more than 29,500 binary variables only to model the on-off operation process of such units. The installation and balance of plants costs associated to each technology are evaluated referring to [111].

The microgrid under analysis is an on-grid system. It has the possibility of buying the electricity from the grid, but it does not have a contract to sell the generated electricity on the market, so it is one-way connection. Moreover, it is also connected to the distributed natural gas network to fuel gas turbines and natural gas boilers. In both cases, it is necessary to stipulate two contracts with the relative electricity and the natural gas providers, that also include a cost proportional to the maximum electric power and natural gas rate required by the system during the year. For this reason, the model has also to contain design variables to evaluate the additional costs of such contracts. We assume a cost of the electricity connection equal to 50€/MW/y, while for the natural gas network 70€/MW/y (referred to the LHV of natural gas). For the cost of the single MWh of electricity and natural gas we referred to historical real market prices, hence also considering typical hourly variations during the entire year.

Clustering of the yearly profiles

The design problem of the microgrid under analysis is solved in three different cases, gradually increasing the length of the time horizon to test the performances of the decomposition algorithm. The profiles for the demands of electricity, heating and cooling are always generated clustering the yearly data directly collected on field in a previous year.

Looking at fig. 4.2, we can extract some insights on the type of profiles we are dealing with. Looking at heating and cooling profiles (fig. 4.2b and 4.2c), it is possible to immediately notice the strong seasonality of the location, where there are remarkable peaks of heating load during winter and equally notable peaks of cooling demand during summer with much more attenuated values (more than halved in some periods) in the intermediate seasons. In addition to that, there

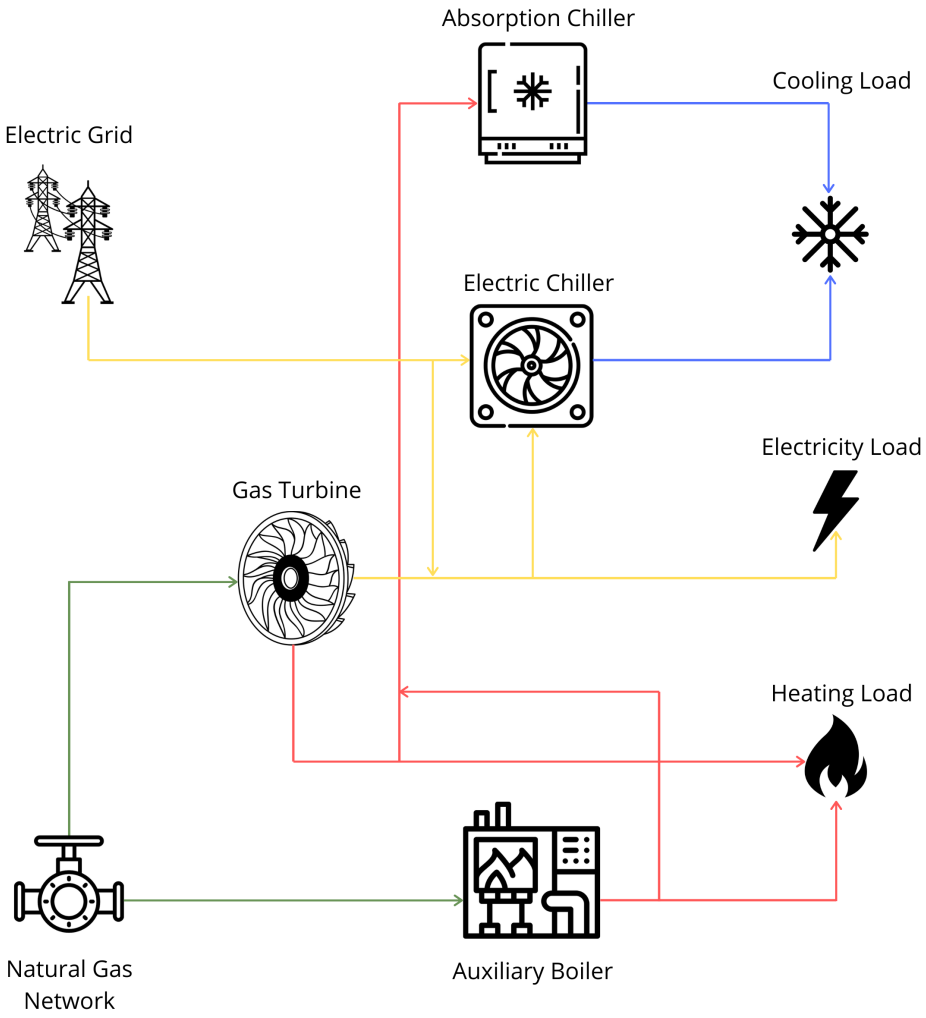


Figure 4.1: The microgrid architecture of the Bovisa case study

					Gas Turbine					
Nominal Power [MW]	0.2	0.25	0.33	1	1.7	1.876	3.515	3.98	4.6	5.38
Heating Rate [MW]	0.295	0.376	0.45	1.299	3.233	3.945	8.92	8.80	9.56	10.91
Efficiency [-]	0.295	0.289	0.311	0.295	0.269	0.247	0.279	0.297	0.293	0.323
Cost [€/kW]	2001	1765	1716	1628	912	880	853	829	804	827
					Auxiliary Boiler					
Nominal Power [MW]	0.7		1		1.4		2			
Efficiency [-]	0.92		0.92		0.92		0.92			
Cost [€/kW]	19.43		16.9		15.29		14.35			
					Electric Chiller					
Nominal Power [MW]	0.564		0.704		0,844		1,056			
COP [-]	5		5		5		5			
Cost [€/kW]	115		115		115		115			
					Absorption Chiller					
Nominal Power [MW]	0,692		1,036		1,382		1,728			
COP [-]	1.2		1.2		1.2		1.2			
Cost [€/kW]	240		240		240		240			

Table 4.1: Catalogue of the available models for the Bovisa case study

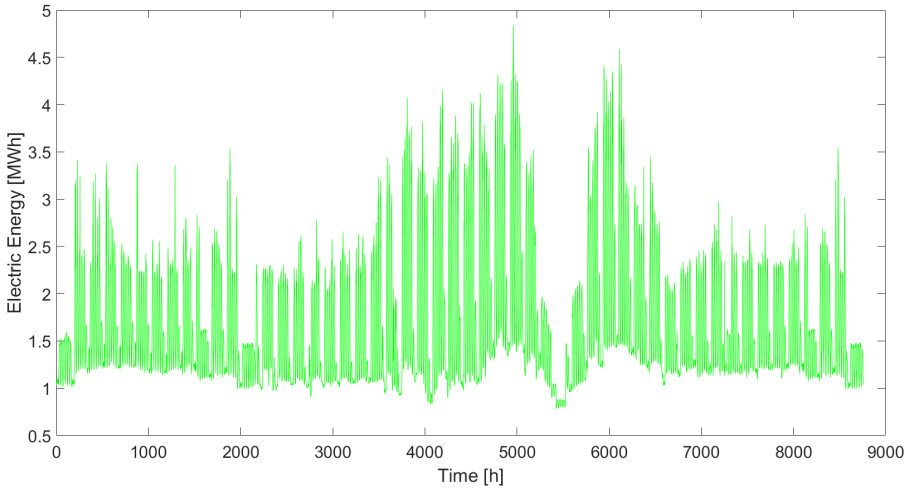
are some peculiarities inherent to the university campus, such as long vacation periods in which the energy demands is strongly reduced. It is possible to notice for instance the drop in the electricity request during August (see fig. 4.2a) as well as the reduction on the loads of both electricity and heating during winter holidays. Moreover, we must consider the variations of the loads during the weekends and the randomness of the consumptions related to laboratory activities.

It is evident that, in order to propose a robust optimal design solution for such a system, it is necessary to run the simulation on a high number of representative days. The hourly profile resolutions for the representative days are generated by means of a traditional k-means algorithm, that allows to build average fictional profiles representative of a family of real daily profiles. To each profile we associate a weight on the basis of the cardinality of the cluster and we estimate the total operational costs of the microgrid properly weighting the OPEX evaluated for each single representative period.

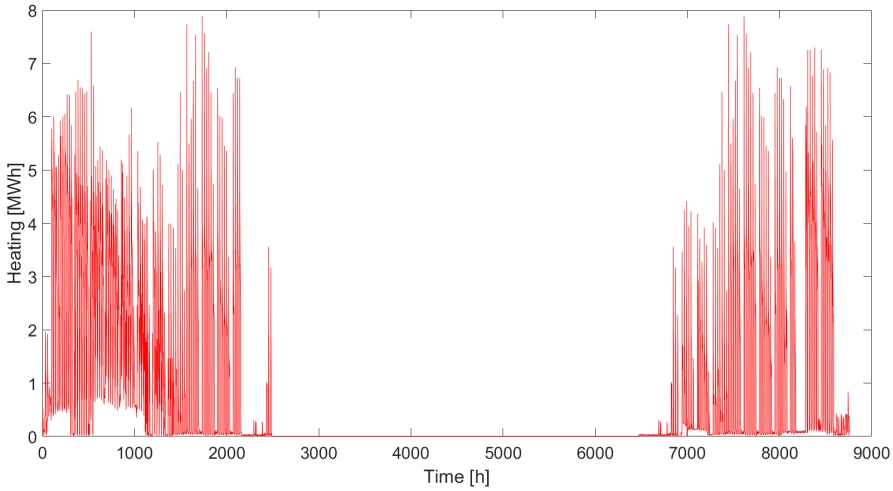
The case study has been firstly analysed clustering only three representative days, as typically done to simulate to main seasons (winter and summer) and an intermediate one. It is clear that the 3-day design problem may produce a design solution not robust to many operating conditions, since it is able to capture only the seasonality of the location. To obtain a more representative solution we need to move to a 7-day and, finally, to a 14-day design problem, in which the model includes more than 3 representative days for each one of the four seasons. Of course, extending the time horizon leads to a significant increase of the integer variables of the model, resulting in a tough computational challenge.

4.1.2 Mathematical modelling of the design problem

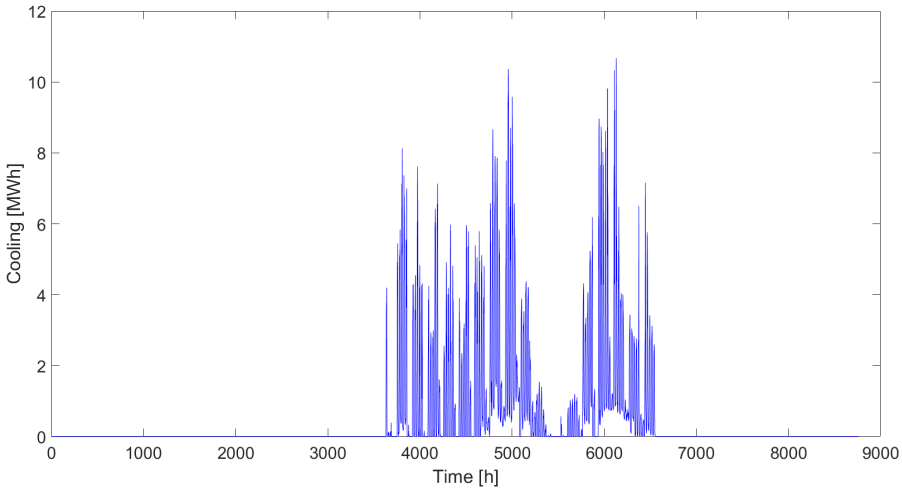
Following a traditional set-oriented algebraic formulation of the problem, we start the descripyion of the MES mathematical model by defining the sets on which the main variables will be indexed.



(a) *Yearly profile of the electricity demand*



(b) *Yearly profile of the heating demand*



(c) *Yearly profile of the cooling demand*

Figure 4.2: Yearly demand profiles of the Bovisa university campus

Sets

$I := \{\text{GT, AB, EC, AC}\}$	Available technologies
$J_i := \{\text{M}_{1\text{i}}, \text{M}_{2\text{i}}..\text{M}_{\text{Ni}}\}$	Models in the catalogue for each technology $i \in I$
$U := \{1,2,3,4\}$	Potential installable units for each model $j \in J_i$
$TYP := \{1,2,3,4....\text{N}_{\text{typ}}\}$	Typical days
$T := \{1,2,3,4...24\}$	Hourly time step of each period

As we can notice, the model has been reported referring to a generic number N_{typ} of representative days that compose the overall time horizon for the unit commitment. This is the case for our specific case study, but in general the reference typical periods may also be weeks or months. To keep the model as general and flexible as possible with respect to the addition of new technologies to the microgrid, we defined a set I containing all the technologies and the group of indexed sets J_i including all the catalogue models for the technology $i \in I$. This could be easily implemented thanks to the new features integrated in the POLIMIP toolbox.

Let us now present the variables introduced in the model. We can organize them highlighting the different categories that will be used to decompose the problem. In particular, integer/binary design variables constitute the high hierarchy, while integer/binary operation variable represents the low hierarchy that will not be branched in the upper level. Also the objective function can be already decomposed in its fundamental parts that will be used to define the global and local auxiliary problems.

Variables

Binary/Integer Design Variables

$$\hat{\gamma}_{i,j} := \text{Selection variable for the model j of the technology i}$$
$$\forall j \in J_i, i \in I$$
$$\hat{\varphi}_{i,j,u} := \text{Investment variable for the unit u of the j}^{\text{th}}\text{ model}$$
$$\forall u \in U, j \in J_i, i \in I$$

Binary/Integer Operation Variables

$\hat{z}_{i,j,u,n,t}$:= On-off variable for u$^{\text{th}}$ unit at each time step t of the typical period n

$\forall t \in T, n \in TYP, u \in U, j \in J_i, i \in I$

$\hat{\delta}_{i,j,u,n,t}$:= Startup variable for u$^{\text{th}}$ unit at each time step t of the typical period n

$\forall t \in T, n \in TYP, u \in U, j \in J_i, i \in I$

Continuous Design Variables

$\hat{el}^{max}$:= Maximum power required to the national electric grid [MW]

$\hat{fu}^{max}$:= Maximum fuel consumption rate required to the natural gas network [MW]

Continuous Operation Variables

$\hat{y}_{i,j,u,n,t}$:= Primary energy output of the u$^{\text{th}}$ unit at each time step t of the typical period n

$\forall t \in T, n \in TYP, u \in U, j \in J_i, i \in I$

$\hat{y}^2_{GT,j,u,n,t}$:= Secondary energy output of the u$^{\text{th}}$ GT unit at each time step t of the typical period n

$\forall t \in T, n \in TYP, u \in U, j \in J_{GT}$

$\hat{x}^{ty}_{i,j,u,n,t}$:= Energy input of the u$^{\text{th}}$ unit at each time step t of the typical period n

$\forall t \in T, n \in TYP, u \in U, j \in J_i, i \in I$

$\hat{x}_{i,j,u,n,t}$:= Energy input of the u$^{\text{th}}$ unit accounting for start-up consumptions

$\forall t \in T, n \in TYP, u \in U, j \in J_i, i \in I$

$\hat{out}_{i,n,t}$:= Total primary energy output from the i$^{\text{th}}$ technology

$\forall t \in T, n \in TYP, i \in I$

$\hat{out}^2_{GT,n,t}$:= Total secondary energy output from gas turbines

$\forall t \in T, n \in TYP$

$\hat{in}_{i,n,t}$:= Total energy input for the i$^{\text{th}}$ technology

$\forall t \in T, n \in TYP, u \in U, j \in J_i, i \in I$

$\hat{el}_{n,t}$:= Purchased electricity from the grid

Objective Function

$\hat{f}_D :=$ Part of the objective function depending on integer design variables

$\hat{f}_{D*} :=$ Part of the objective function depending on continuous design variables

$\hat{f}_{O_n} :=$ Part of the objective function depending on operation variables of the typical period n

$\forall n \in TYP$

It is worth saying that, despite the relative simplicity of the Bovisa microgrid architecture, the accurate design model must include continuous design variable for modelling the maximum electric power and fuel consumption rate required to the external networks. As extensively reported in sec. 3.2, in a hierachical decomposition method continuous design variables act as coupling variables between the UC representative periods, hindering the implementation of multiple worker problems, as done by Yokoyama et al. in [104]. This shows that, even in relatively simple MES, the need of a universal MES decomposition method like that introduced in chap. 3 is crucial to implement an efficient search algorithm without giving up modelling accuracy.

First of all, dealing with a catalogue-based design with discrete equipment capacities requires to define the relationship between investment variables for each microgrid unit $\varphi_{i,j,u}$ and its operation variables. In this case, the model must also provide the possibility of selecting one of many available models for each technology, so on the top of unit investment variables it is necessary to add also *model selection variables* $\hat{\gamma}_{i,j}$. Only one model can be chosen. The investment variables corresponding to each unit $\varphi_{i,\bar{j},u}$ belonging to a discarded model $\bar{j} \in J_i$ must be set to zero, constraining their value to the value of the selection variable $\hat{\gamma}_{i,j}$. Finally, unit investment variables are the one that define the possibility of switching on a given potential machine unit. In particular, if the investment variable is equal to one, then all the on-off variables $z_{i,j,u,n,t}$ can be freely switched on or off in the different time periods. Otherwise (that is, if the investment variable is zero), all the $z_{i,j,u,n,t}$ are inevitably set to zero, meaning that the potential unit becomes inoperable by the microgrid.

Constraints

Catalogue Selection and Investment Decision

$$\sum_i \hat{\gamma}_{i,j} \leq 1 \qquad \forall j \in J_i, i \in I$$

$$\hat{\varphi}_{i,j,1} = \hat{\gamma}_{i,j} \qquad \forall j \in J_i, i \in I$$

$$\sum_{n \in TYP} \sum_{t \in T} \hat{z}_{i,j,u,n,t} \leq \varphi_{i,j,u} \cdot \mathbb{N}_{ty} \cdot 24 \qquad \forall u \in U, j \in J_i, i \in I$$

In order to avoid operation symmetries that might hinder convergence of the optimization process, it is fundamental to prioritize the investment decision as well as the unit starting process. Given four potential unit for each model, it is not possible to invest on the second unit if the first unit has not be already selected, as well as on the third with respect to the second and so on. Very similarly, we must impose that if more than one unit can be switched on, the system must turn on the units of the same model starting from the first one and following a cardinal order.

Constraints

Prioritization Constraints

$$\hat{\varphi}_{i,j,u} \leq \hat{\varphi}_{i,j,u-1} \qquad \forall u \in U \setminus \{1\}, j \in J_i, i \in I$$

$$\hat{z}_{i,j,u,n,t} \leq \hat{z}_{i,j,u-1,n,t} \qquad \forall t, \in T, n \in TYP, u \in U \setminus \{1\}, j \in J_i, i \in I$$

As far as the characteristic curves of the machine are concerned, we decided to model them with a simple linear relation based on the information available from the catalogues. In general, it might be possible to approximate even more complex off-design curves (quadratic or cubic relations) by means of piecewise interpolation that can be implemented introducing so-called SOS constraints available in CPLEX. Nonetheless, the difference in the performances is generally very limited, as well as the impact on the design solution, such that the increase of complexity of the model may not be justified [34]. We still decided to consider the variation of the operational curves associated to the operating temperature ϑ,

because it does not affect at all the complexity of the model, but it may significantly vary the performance of the machines, in particular for the refrigeration units.

Constraints

$$\hat{y}_{i,j,u,n,t} = m_{i,j}\hat{x}^{ty}_{i,j,u,n,t} + (q_{i,j} + m^T_{i,j}\vartheta_{n,t})\hat{z}_{i,j,u,n,t} \qquad \forall t \in T, n \in TYP, u \in U, j \in J_i, i \in I$$

$$x^l_{i,j}\hat{z}_{i,j,u,n,t} \leq x^{ty}_{i,j,u,n,t} \leq x^u_{i,j}\hat{z}_{i,j,u,n,t} \qquad \forall t \in T, n \in TYP, u \in U, j \in J_i, i \in I$$

$$\hat{out}_{i,n,t} = \sum_{j \in J_i}\sum_{u \in U} \hat{y}_{i,j,u,n,t} \qquad \forall t \in T, n \in TYP, i \in I$$

$$\hat{in}_{i,n,t} = \sum_{j \in J_i}\sum_{u \in U} \hat{x}_{i,j,u,n,t} \qquad \forall t \in T, n \in TYP, i \in I$$

One of the main advantage of MILP optimization is the possibility of modelling in the details the operation dynamics of the machines. In particular, we imposed a minimum up time of 3 hours for all the machine and ramp limits equal to 30% of the nominal output. In addition to that, we assumed an additional consumption associated to start-up equal to 5% of the nominal input.

Constraints

Start-up, minimum up time and ramp limits

Start-up, minimum up time and ramp limits

$$\hat{\delta}_{i,j,u,n,t} \geq \hat{z}_{i,j,u,n,t} - \hat{z}_{i,,ju,n,t-1} \qquad \forall t \in T, n \in TYP, u \in U \setminus \{1\}, j \in J_i, i \in I$$

$$\hat{x}_{i,j,u,n,t} = \hat{x}^{ty}_{i,j,u,n,t} + SU\hat{\delta}_{i,j,u,n,t} \qquad \forall t \in T, n \in TYP, u \in U \setminus \{1\}, j \in J_i, i \in I$$

$$\hat{y}_{i,j,u,n,t-1} - \hat{y}_{i,j,u,n,t} \leq ru_i + (1 - \hat{z}_{i,j,u,n,t})M \qquad \forall t \in T \setminus \{1\}, n \in TYP, u \in U, j \in J_i, i \in I$$

$$\hat{y}_{i,j,u,n,t} - \hat{y}_{i,j,u,n,t-1} \leq rd_i + (1 - \hat{z}_{i,j,u,n,t})M \qquad \forall t \in T \setminus \{1\}, n \in TYP, u \in U, j \in J_i, i \in I$$

$$\sum_{t-t^{min}+1}^{t} \hat{z}_{i,j,u,n,t} \geq t^{min}\hat{\delta}_{i,j,u,n,t-t^{min}+1} \qquad \forall t \in \{t^{min}, \ldots, T^{end}\}, n \in TYP, u \in U, j \in J_i, i \in I$$

Finally, it is necessary to impose the energy balance at each node of the microgrid. We can report the one for the electricity since it is more general. The structure is always based on an input-output balance at the considered node.

Constraints

Energy balance at the electricity node

$$\sum_{i \in OUT^{el}} \hat{out}_{i,n,t} - \sum_{i \in IN^{el}} \hat{in}_{i,n,t} \geq d^{el}_{n,t} - \hat{el}_{n,t} \qquad \forall t \in T, n \in TYP$$

OUT^{el} and IN^{el} represents the generic subsets of I including the technologies that respectively generate and consume electricity. The last constraints are the one

regarding the continuous design variables, that are used to evaluate the cost of the supplying contracts with the providers of electricity and natural gas.

Constraints

Energy balance at the electricity node

Maximum fuel consumpiton and power from the grid

$$\hat{el}_{n,t} \leq \hat{el}^{max} \qquad \forall t \in T, n \in TYP$$

$$\sum_{i \in FU} \hat{in}_{i,n,t} \leq fu^{max} \qquad \forall t \in T, n \in TYP$$

FU is the generic subset of I composed by the machine consuming natural gas as input. The only missing part is the one associated to the objective function. To easily implement the MILP decomposition it is convenient to define the objective function decomposing it in its main components from the beginning. The aim of the optimization is to minimize the cost associated to the multi-energy microgrid under analysis. For this reason, we decided to refer to the *annuity* as parameter to be optimized. The annuity is the annual cost associated to the running microgrid, including both CAPEX and OPEX. In order to turn the initial investments into an equivalent annual expenditure we can use the so-called Capital Recovery Factor (CRF) assuming a discount rate of 5% and a lifetime of 15.

Constraints

Objective Function

$$\hat{f}_D = \sum_{i \in I} \sum_{j \in J_i} \sum_{u \in U} C_{i,j} \hat{\varphi}_{i,j,u} CRF$$

$$\hat{f}_{D*} = \hat{el}^{max} C^{el} + \hat{fu}^{max} C^{fu}$$

$$\hat{f}_{O_n} = \left(\sum_{i \in FU} \sum_{t \in T} \hat{in}_{i,n,t} c^{fu}_{n,t} + \sum_{t \in T} \hat{el}_{n,t} c^{el}_{n,t} \right) \frac{8760}{H_n} \qquad \forall n \in TYP$$

$$\hat{Objective} = \hat{f}_D + \hat{f}_{D*} + \sum_{n \in TYP} \hat{f}_{O_n}$$

4.2 Analysis of the numerical performances

The decomposition algorithm presented in chap. 3 has been tested referring to the Bovisa design problem. After setting up the model for the microgrid architecture reported in sec. 4.1, the problem has been solved gradually extending the number

of representative days composing the time horizon, from 3 up to 7 and finally 14 typical days. The decomposition algorithm has been implemented by means of the POLIMIP modelling environment, relying on IBM CPLEX 12.9 as low-level solver. In particular, advanced solver features such as "root node processing" and "dynamic search" have been disabled in the master problem to avoid any type of conflict with the custom search logic implemented by means of the user-defined generic callbacks called at each node of the upper level.

For each case, we also solved the problem by means of a conventional B&B algorithm as a benchmark. All the solver parameters were left untouched to the default values automatically set by CPLEX. The only exception concerns branching priority orders: a higher branching priority was given to integer and binary design variables in order to fairly compare the two solution approaches. In fact, a conventional CPLEX instance without branching priority orders would not exploit at all the hierarchical relationship between design and operation variables, hence performing worse than what possible for a MES design problem. For all the simulations we used a 4-core 2.9 GHz personal computer with a 16Gb RAM. The main purpose of our test is twofold:

1. Assessing the effectiveness of the integer cuts implemented both at the upper and at the lower level by the decomposition algorithm. This is necessary to understand if the global and local auxiliary problems have been properly defined and if they provide useful information to speed up the B&B tree exploration;

2. Assessing the computational performance of the novel decomposition algorithm referring to the conventional B&B algorithm enforced with branching priorities as benchmark. In particular, we want to analyse the behaviour of the algorithm with increasing length of the time of horizon and so the complexity of the problem, when the conventional algorithm struggles for convergence.

4.2.1 Performance analysis: 3-day design problem

The design problem presented in sec. 4.1 has been firstly solved clustering the profiles in 3 representative days. The optimal design solution is reported in tab. 4.3. It is worth saying that the very same solution is obtained if the problem is solved either with the decomposition algorithm or with the conventional B&B implemented by CPLEX, as expected.

As we can notice, all the technologies have been installed, in general with a high number of units. Focusing on the gas turbines, the global optimization leads to 4 small turbines instead of a single large turbine model. This is probably related to the fact that a high operation flexibility is required in such microgrid, where many components interacts sharing inputs and outputs. Moreover, load profiles are characterized by a significant variability too, not only because of the strong seasonality of the location but also for the randomness of the heavy university loads (mainly related to laboratory activities). Since in the model we introduce detailed constraints regarding the unit commitment (ramp limits, minimum uptime, start-up costs), the economy of scale of large models is not sufficient to counterbalance the operation flexibility given by smaller units. Moreover, since the model well-describes also off-design condition of the machines and imposes a minimum load for each unit, when the demand is extremely low (like the demand of cold water during winter days) if small units are available it is possible to limit energy waste. More detailed comments are provided at the end of this section.

Let us now focus on the performances of the novel decomposition algorithm. First of all, it is necessary to verify if the auxiliary problems have been well-defined and if the additional integer cuts introduced in the decomposition are effective. As extensively discussed when presenting the decomposition algorithm in chap. 3, the role of local auxiliary problems is to evaluate lower bounds of the global objective function that allows to discard the corresponding integer design candidate before solving the complete worker problem in the lower level. Looking at fig. 4.3, we can state that the results are extremely promising. Considering 139 design candidates

	Design Solution			
Selected Technology	GT3	AB1	EC4	AC1
Number of Units	4	1	3	3
Maximum Power from the Grid	2.95 MW			
Maximum NG Network Request	4.97 MW			
Objective Function	2,642,976 €/y			

Table 4.2: Design solution of the 3-day design problem

entering the lower level, only 51 complete worker problems are solved, while 128 are prematurely discarded thanks to local auxiliary problems (about two on five). In addition to that, we can see that over 695 potential local auxiliary problems, only 525 (around 75%) are actually solved in the lower level. This means that in general is sufficient to solve the auxiliary problems referred to 2 of the 3 typical periods to find the design candidate is infeasible of suboptimal. This percentage is higher during the first branching operations and tends to decrease as the number of design candidates entering the lower level increases. The reordering of the local auxiliary problems allows the algorithm to gradually learn which are the typical periods that favour infesibility or suboptimality. As a result, the decomposition algorithm spends only 27.4% in the lower level (around 192 seconds), meaning the computational burden introduced by the solution of the auxiliary problems is acceptable if compared to the branching operations of the upper level. Finally, the results are convincing also for what global auxiliary problems are concerned. In spite they require around 45% of the overall solution time to be solved, they contribute to discard 131 design candidates even before entering the lower level (in addition to improving the effectiveness of the lower bounds in the lower level).

Now let us compare the computational performances of our decomposition algorithm with the conventional default B&B algorithm implemented by CPLEX and used as benchmark. As reported in fig. 4.4, the decomposition algorithm is competitive also in a limited time horizon of 3 days. The solution time is halved with respect to the traditional algorithm. Of course, this result seems not extremely

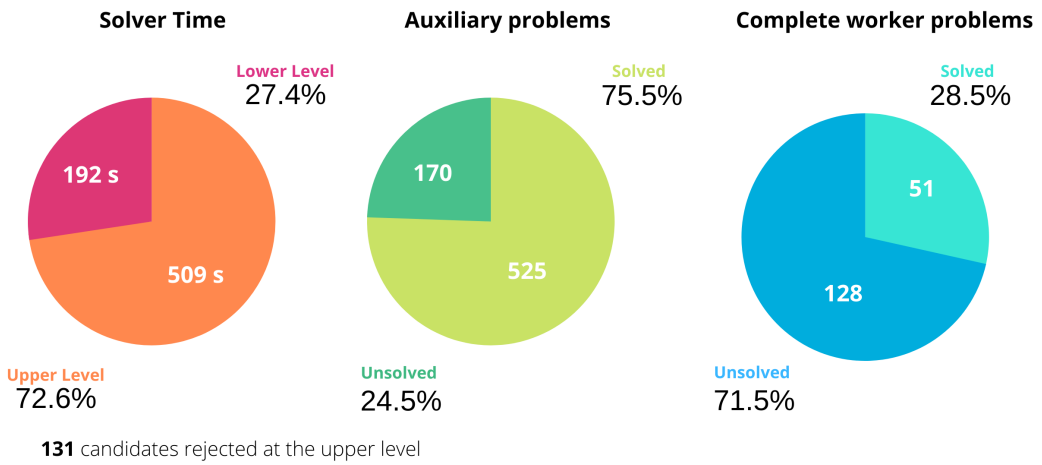
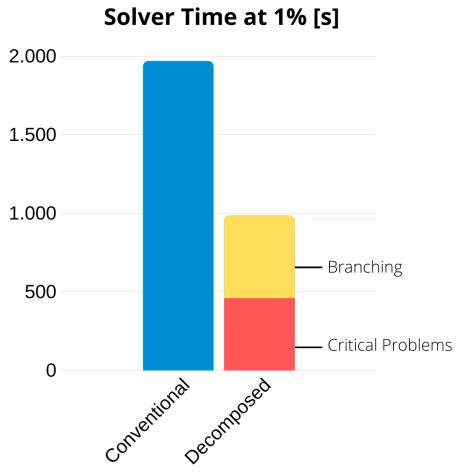


Figure 4.3: Analysis of the effectiveness of the introduced auxiliary problems and cuts in the 3-day design problem

remarkable, since the traditional algorithm takes a more than acceptable amount of time to get to 1% gap (around 2,000s). Yet, it is clear that with such microgrid architecture a time horizon of 3 days (72 operation hours) can be easily optimized by a traditional branch and bound algorithm, that can also take advantage of CPLEX advanced features like dynamic search and root node processing. In fact, being the solution of the complete worker problem relatively easy, the additional computational effort required by the decomposition algorithm to solve the auxiliary problems requires only provides a limited advantage. Nonetheless, it is obvious our decomposition algorithm is meant to be applied when the conventional B&B takes long time to converge, so when the problem is sufficiently complex. It is interesting to investigate what happens when increasing the time horizon from 7 to 14 typical days.

4.2.2 Performance analysis: 7-day design problem

First of all, let us observe the design solution presented in tab. 4.3. As we can notice, increasing the number of representative days the optimal design solution includes a higher number of units for boiler and chiller and a larger model for the absorption chiller, too. Performing a 7-day design optimization, the model is able to better describe the peaks and the variability of heating and cooling demands, that



	Conventional	Decomposed
1% gap	1,969 s	986 s
0.01% gap	4,555 s	1,161 s

Figure 4.4: Computational performances of the novel decomposition algorithm in the 3-day design problem

	Design Solution			
Selected Technology	GT3	AB1	EC4	AC2
Number of Units	4	3	4	3
Maximum Power from the Grid	3.58 MW			
Maximum NG Network Request	5.64 MW			
Objective Function	2,716,181 €/y			

Table 4.3: Design solution of the 7-day design problem

are strongly seasonal. It goes without saying that increasing the number of typical periods for the simulation of the unit commitment allows to obtain a solution which is more realistic, but also more robust to different operative conditions that may occur when operating the plant.

Let us focus more on the analysis of the performance of the decomposition algorithm. fig. 4.5 shows that out of 133 design candidates entering the lower level, only 9 (6.8%) are solved, and all of them have generated a new cutoff value for the search algorithm (that is to say that the algorithm has not wasted time solving a complete sub-problem that finally revealed itself for being suboptimal). As far as the local auxiliary problems is concerned, only around 59% has been actually

run in the lower level (with respect to 76% in the case of 3-day design), confirming that the more auxiliary problems the algorithm has to solve the smarter it becomes in recognizing which are the typical periods that might most likely generate an infeasible or suboptimal result. Moreover, 83 design candidates are rejected at the upper level relying on the information extracted by global auxiliary problems. The ratio between time in the upper level and time in the lower level is still around 2-to-1, meaning that even when the number of auxiliary problems significantly increases (long time horizon) the introduced integer cuts keep the computational burden of the lower level relatively low.

In this case, the decomposition algorithm starts showing notable performances. As expected, moving from 3 to 7 typical days, the complexity of the problem has exploded and the search tree for the conventional B%B algorithm has grown exponentially. CPLEX with default settings and branching priorities takes more than 24,200 s (6 hours and 40 minutes) to reach a relative gap of 1%, while the decomposition algorithm achieves the same gap in around 1,621 s (27 minutes), outperforming the conventional one and being 15 times faster (see fig. 4.6). It is also interesting to highlight that the decomposition algorithm requires only additional 500 seconds to reach full convergence, while the conventional one still needs more than 20,000 s. A similar phenomenon occurs also in the 3-day design problem (see fig. 4.4). Generally, the decomposition algorithm at 1% gap has already gathered a lot of information from the solution of worker and auxiliary problems. Integer cuts start being very effective and the full converge is achieved quite quickly.

4.2.3 Performance analysis: 14-day design problem

Finally, we tested our algorithm trying to further increase the complexity of the problem, increasing the length of the time horizon to 14 typical days. Despite the relatively simple architecture of the microgrid, the extension of the time horizon introduced a very high number of binary variables to the problem, creating the case which the decomposition algorithm has been thought for. Tab. 4.4 shows how the increase of the number of representative days still have non-negligible impact

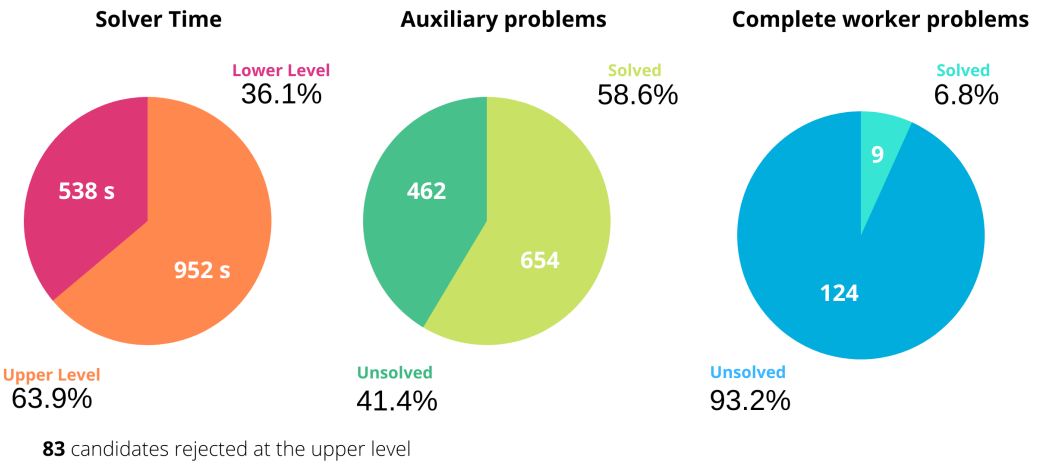


Figure 4.5: Analysis of the effectiveness of the introduced auxiliary problems and cuts in the 7-day design problem

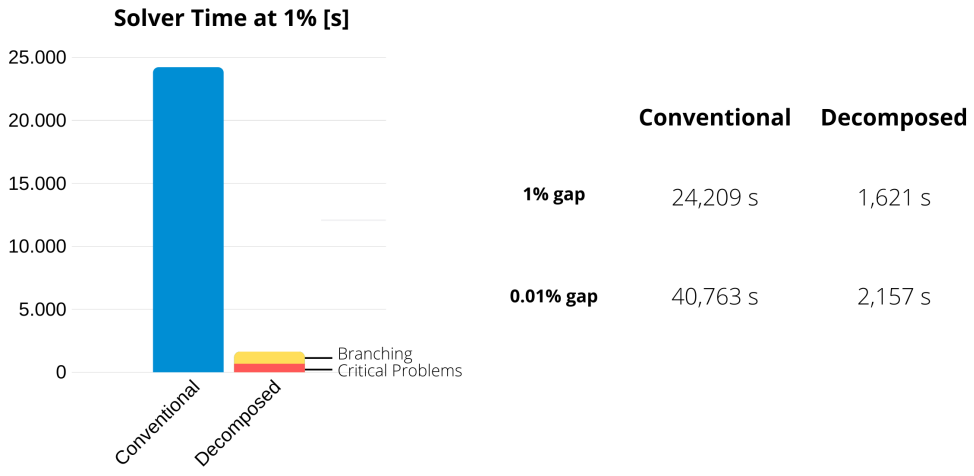


Figure 4.6: Computational performances of the novel decomposition algorithm in the 7-day design problem

	Design Solution			
Selected Technology	GT3	AB2	EC4	AC4
Number of Units	4	4	4	3
Maximum Power from the Grid	4.35 MW			
Maximum NG Network Request	6.33 MWh			
Objective Function	2,799,185 €/y			

Table 4.4: Design solution of the 14-day design problem

on the design of the microgrid, increasing the size of the system and changing the contracts with the electric grid and the natural gas provider. Overall, moving from 3 to 14 representative days, we have an increase of the objective function of around 5.9%. And it is worth saying that the architecture of this microgrid is extremely simple, as it does not include storage, unpredictable renewable sources or seasonal constraints. In general, the impact of the extension of the time horizon might be much higher.

All the considerations made in the previous cases still hold for 14-day design problem (see fig. 4.7). The percentage of complete worker problems solved is slightly increased, but it is still below 10%. Moreover, this might be caused by the fact that upper level cuts in this case have become extremely effective, rejecting 342 design candidates at the upper level, and reducing the number of design candidates entering the lower level. Extending the time horizon, and so the number of typical periods, it is necessary to solve a higher number of global auxiliary problems, but it is also more likely that the profile of a certain representative day leads to higher global lower bounds $\underset{\sim}{\check{f}}_D$ and $\underset{\sim}{\check{f}}_{D^*}$ (see subsec. 3.2.4), hence favouring the rejection of a higher number of design candidates in the upper level. It is worth recalling that integer cuts in the upper level are the most effective in reducing the solution space and improving computational performances.

The 14-day design problem is the one really demonstrating the effectiveness of the decomposition algorithm. As reported in fig. 4.8, when increasing the time

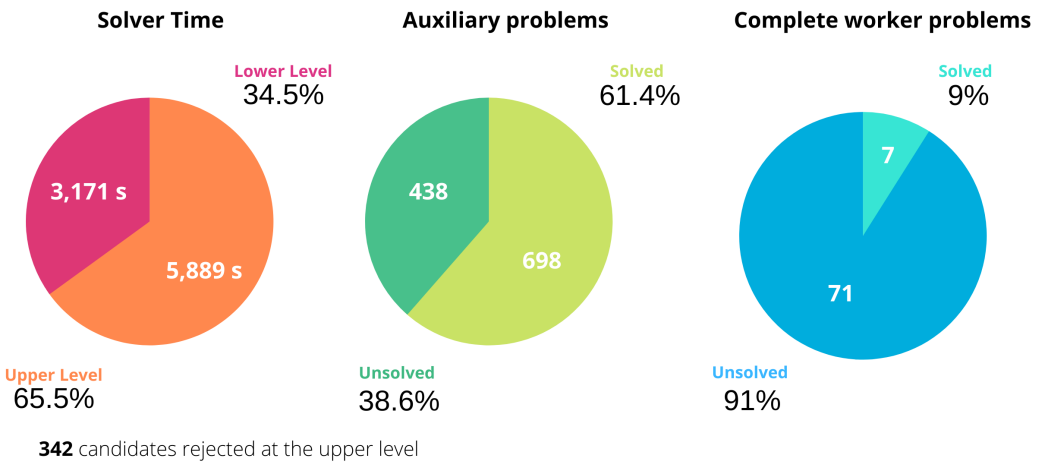


Figure 4.7: Analysis of the effectiveness of the introduced auxiliary problems and cuts in the 14-day design problem

horizon from 7 to 14 representative days, the conventional B&B algorithm does not converge in practical times. After 50,000s the relative gap is still above 5% and the search procedure is stuck. Introducing additional information that allow to prune extensive portions of the tree at the upper level, thus limiting the exploration in the lower level, is the only way to obtain the global optimum solution in a reasonable amount of time. The decomposition algorithm achieves 1% relative gap in less than 7,200 s (2 hours) and reaches the default gap of 0.01% in only 7,654 s.

The results obtained in the three cases are summarized in fig. 4.9. The presented graph finally enforces the theoretical discussion presented in chap. 3 with numerical evidence and highlights the potentialities of the decomposition algorithm developed in this work. Conventional B&B shows the typical exponential growth in computational time when increasing the time horizon of the design problem, such that in the 14-day design problem is not able to converge to the optimal solution in an acceptable amount of time. Introducing the concept of auxiliary problems and exploiting the multi-period nature of MES design problems, the decomposition algorithm is able to effectively deal with an increasing number of integer operation variables, thanks to a more intelligent exploration of the lower level. Increasing the time horizon from 3 days to 14 days to obtain a more robust design solution,

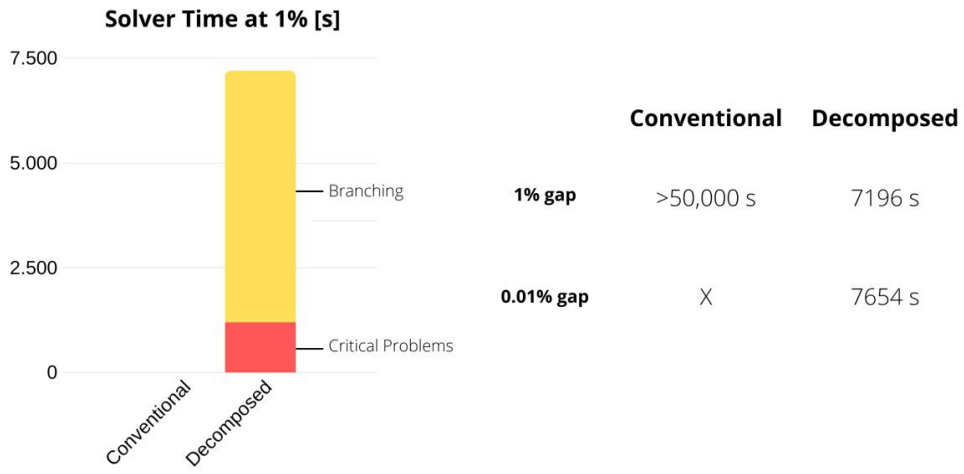


Figure 4.8: Computational performances of the novel decomposition algorithm in the 14-day design problem

the algorithm moves from being the most efficient way to evaluate the optimum solution (15 times faster the conventional one) to being the only possible way to find the optimum design candidate in practical times.

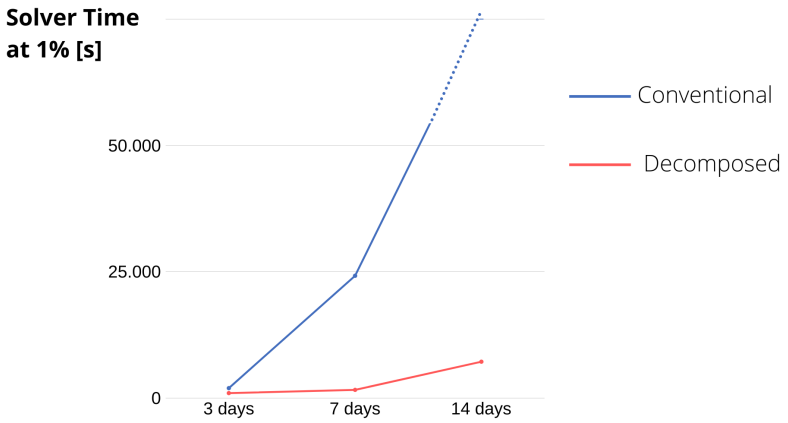


Figure 4.9: Computational performance comparison increasing the time horizon

Analysis of the design and of the corresponding economic dispatch solution

To complete the analysis, it is possible to extract some insights from the design and operation solution obtained for the most representative case, that is to say the design problem solved considering 14 representative days. Such analysis is necessary to confirm the validity of the implemented model and to better understand the main characteristics of the MES.

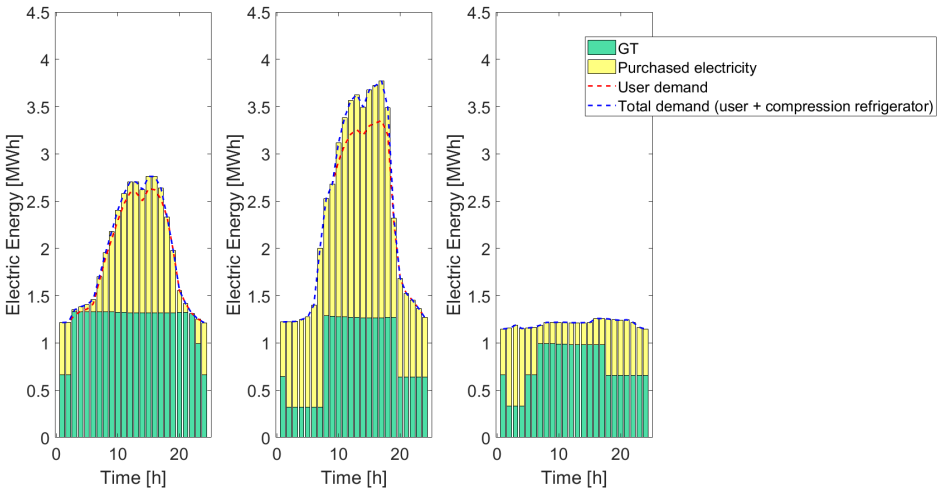
As already underlined at the beginning of this section, the optimal design solution tends to select many smaller units instead of a large one, in particular concerning gas turbines. Looking at the load profiles (see fig. 4.2) and at the optimal design solution (see tab. 4.4), it is possible to observe that the catalogue of gas turbines perfectly fits the size of the system. Many possible models of different size and with different number of units can produce a feasible solution, introducing many trade-offs to the problem. The optimal solution includes a small model with 4 units (the maximum available), leveraging more on the flexibility of the system instead of the economy of scale. Moreover, model GT3 is a relatively new model, with very good efficiency (considering a small gas turbine), but also a relatively high cost per kW. Due to its high efficiency, the gas turbine is used to satisfy both the electric and thermal baseloads. Hence, it results particularly convenient to invest on maximizing its performances even at a cost of a higher CAPEX.

Looking at fig. 4.10, we can make some more general considerations on the operation strategy. As far as electricity is concerned, gas turbines are generally operated following the heating demand and not the electricity request. In general, given the yearly input data on the price of electricity and the price of natural gas, the electricity generation from gas turbine seems very convenient when it is possible to also recover waste heat. Once the waste heat is fully exploited, the remaining electricity demand can be satisfied simply buying electricity from the grid.

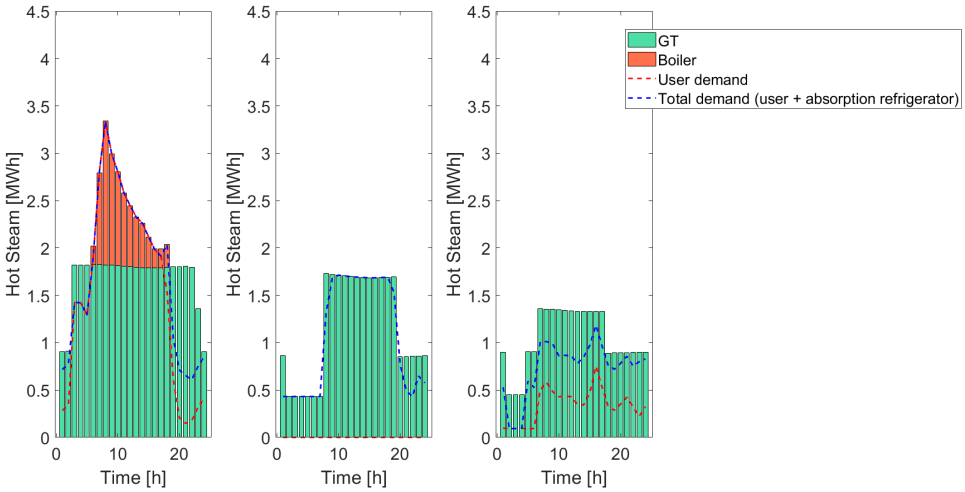
For the very same reasons, auxiliary boilers are installed only to cover the winter

peaks of the heating load, when the nominal heat rate of the installed gas turbines is not sufficient (see fig. 4.10b). As far as the cooling is concerned, the optimal solution makes an extensive use of absorption chillers, since they allow to make the electricity generation through gas turbines more convenient also during summer, when the heating demand is null (central part of fig. 4.10). Electric chillers are mainly used to satisfy the cooling demand peaks during summer or to supply the minimum base load required during winter (left-hand side of fig. 4.10) when all the heat is consumed by the heating system and there is no way to feed absorption chiller.

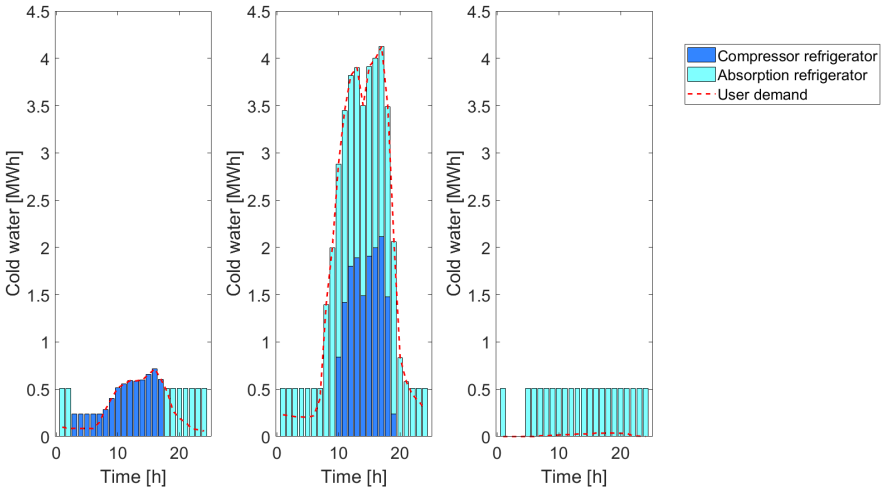
Finally, we can notice that dynamics constraints such as minimum up-time and start-up costs favour a regular exploitation of the different units, without continuous on-off operations. This produce a solution that can be more easily implemented on a real control system accounting also for faster dynamics of the machines. The introduction of a minimum load for the machines highlights the need for a dissipation system, mainly for the cooling system. In fact, there are days in which the cooling demand is extremely low, but not inexistent (probably related to research activities and laboratories) and, also operating the chillers at minimum load, a significant amount of cooling energy still needs to be dissipated.



(a) *Electricity UC of three out of the 14 typical days of the problem*



(b) *Heating UC of three out of the 14 typical days of the problem*



(c) *Cooling UC of three out of the 14 typical days of the problem*

Figure 4.10: ED solution of the 14-day design problem

Conclusions

In this Thesis work, starting from the legacy of Iyer and Grossman [102] and Yokoyama [104], we introduced a novel MILP decomposition method exploiting the inherent hierarchy beween design and operation in MES design problems and we demonstrated its effectiveness with respect to conventional approaches by means of a case study involving the design of a multi-energy microgrid for a university campus.

The main innovative elements of our proposed approach are both mathematical and numerical. From a mathematical standpoint, our method does not introduce any restricting assumption on the MES problem structure with respect to previous decomposition algorithms, hence ensuring the **universality** of the approach without sacrificing computational **performance**. Moreover, from a numerical perspective, an effort was made to guarantee a high **usability** of the decomposition algorithm for practical MES design purposes.

- **Universality**. Our method can be applied to MES problems not compatible with previous state-of-the-art decomposition paradigms. In particular, the MES model may contain both continuous and integer/binary design variables, as well as time-coupling constraints. This capability is particularly important in solving MES designs with complex architectures requiring a high modelling flexibility. In its absence, it is either impossible to model some system arichitecture/operation dynamics or it is necessary to introduce ad-hoc modelling strategy that eventually compromises the accurancy of the model and might add unnecessary problem complexity. For instance, the algorithm proposed by Yokoyama et al. allows to use only integer design variables, thus requiring to discretize design choices that in principle should be

modelled as continuous. Other than introducing a model approximation, this fact considerably increases the numerical complexity of the problem adding a high number of integer variables.

- **Performance**. Thanks to the introduction of the innovative concept of *local auxiliary problem* we were able to mantain a high generality of the method without giving up computational performances. This was demonstrated through a real case study in which the novel approach has been compared to the conventional MILP search strategy implemented in the commercial solver IBM CPLEX. Gradually increasing the time horizon and so the complexity of the design problem, conventional B&B showed an exponential growth in convergence time, while the decomposition algorithm demonstrated is capability of taking advantage of the multi-period nature of the problem to reduce complexity. In the case of a 7-day design problem, the decomposition algorithm outperformed the conventional one reaching the same optimal solution 15 times faster. When moving to 14-day design problem, the decomposition algorithm achieves convergence in around 2 hours, while the convetional search strategy is totally uncapable to find the optimum design solution in practical times. These results clearly highlight that the novel algorithm is a method to optimally design complex energy system that will be otherwise unsolvable with standard MILP optimization strategies when considering a sufficiently representative time horizon. In general, for a given MES architecture the algorithm improves the quality the design solution allowing for a higher complexity in operation modelling without an excessive increase in computational time.

- **Usability**. The practical usability of the developed decomposition method is a feature indipendent from the previous two. Nonetheless, it is not less important. In fact, the implementation of MILP decomposition requires an in-depth knowledge of commercial solver API and of the corresponding programming language. Yet, MES desing is a field of interest for academics and professionals that do not necessarly have advanced programming skills. For this

reason, we developed a dedicated code which automates the implementation of the decomposition algorithm in the POLIMIP, the MATLAB modelling environment developed in this Thesis work. Usability of the algorithm is crucial also beacause it favours future applications to new case studies and further developments.

Finally, we want to stress that the formal description of the novel decomposition method proposed in this work has been carried out mantaining a general profile that laid the ground for an actual *decomposition framework for Multi-Energy Systems*. In fact, the high-level concepts of "auxiliary problem" and "constraining feature" can be intended as mathematical tools that can be further expanded and/or modified to create new decomposition methods for specific types of MES problems. Some examples of possible future developments are:

- *Assessment of the performances of the algorithm when applied to different MES architectures and operation models.* The case study presented in the present work is just an initial one aiming at proving the correctness and the effectiveness of the proposed decomposition method. Similar benchmarks must be carried out on longer time horizons and on different MES models such as microgrids containing thermal/electric storages, renewable sources and seasonal constraints to better evaluate the actual competitiveness of the algorithm.

- *Implementation of an heuristic operation logic in the lower level (instead of a predictive one).* This aim can be pursued starting from the observation that the predictive relaxed operation in the upper level always provides a lower bound for an heuristic (sub-optimal) operation in the lower level. It must be pointed out that, with respect to other two layer optimization paradigms performing the same task (e.g. PSO in the upper level + heuristic in the lower level), the advantage of a suitably defined MILP search method is its guarantee of global optimality.

- *Application of the decomposition framework to a multi-year design problem for optimal investment planning considering yearly load variation scenarios.*

Today, this kind of problems are rarely tackled with MILP approaches due to their unmanageable size. However, by relying on a modified version of the decomposition method here presented (e.g. by adding a further hierarchical level) we would obtain a more efficient exploration of the combinatorial solution space, thus increasing the feasibility of a multi-year MILP design model for Multi-Energy Systems.

Appendix A

POLIMIP: an example of implementation on a real case study

In this appendix, the part of the matlab codes used to implement the case study presented in 4 are reported. This represents a typical example of application of the POLIMIP modelling environment to define and run a MILP optimization problem eventually applying the decomposition algorithm developed in this work.

In order to implement MILP optimization in POLIMIP environment, it is necessary to define six fundamental scripts: `sets.m`, `variables.m`, `parameters.m`, `constraints.m`, `objective.m` and `options.m`. All these scripts are based on the syntax presented in sec. 2.2 and here it is possible to find a practical example of how to implement them. In addition to them, we might find 3 additional scripts:

- `priorities.m` for adding branching priorities to the conventional B&B algorithm.

- `preprocessing.m` to insert input data in matrix form before indexing them in `parameters.m`. In general, if input data are composed by 0-dimension or 1-dimension matrices it is possible to define them also in the `parameters.m`. Yet, in case of multi-dimension matrices that may occasionally also be pre-

processed and rearranged, it is a good practise to use the additional script `preprocessing.m`.

- `postprocessing.m` to print the design solution and plot the ED solution.

After defining such script, the process of creating the MILP problem and solving it either using a conventional B&B algorithm or the decomposition algorithm is fully automated by the modelling environment, that relies on other hidden `.m` and `.py` scripts. The user should only run `createprob` and `solveprob` on the MATLAB command line, to respectively create and solve the optimization problem.

A.1 Sets

`sets.m` is generally the first script to be added as it defines the sets of the problem and their cardinality. All the elements of the sets will be represented through a cardinal number and saved in the structure `s`.

Listing A.1: sets.m

```
%Technologies
dset('I',4)
%Models
J = [10 4 4 4];
for i = s.I
dset('J{i}',J(i))
end
%Units
dset('U',4);
%Time steps
dset('T',TP_length)
%Typical periods
dset('TYP',nclus)
```

A.2 Variables

`variables.m` is the script used to define the problem variables, properly indexed according to the defined sets. All the variables will be saved in the dedicated structure `v`. As reported in list A.2, there are two ways to define a indexed variable:

1. Directly inserting the name of the set as a string vector in `dvar` function. This is typically done when the we deal with conventional Cartesian sets.

2. Using for-cycles by means of the structure `s`. This is necessary when defining variables indexed on non-Cartesian sets, that are generally represented by a family of indexed sets with different cardinality. Referring to our case study, an example may be the family of sets of available models that are in turn indexed on the set of the technologies.

During variable definition, it is also possible to insert the flags `set.design` and `set.intop` for identifying design and integer operation variables, in order to automatically implement the decomposition algorithm.

A.3 Parameters

`parameters.m` is the script used to define parameters of the problem properly indexing the input data previously inserted in matrix form. All the parameters will be saved in structure `p`. The syntax is very similar to the one used for variables (see list A.3). It is important to ensure the coherence between the matrix dimensions of input data and the dimensions of the indexing sets.

A.4 Constraints

`constraints.m` is the script used to define constraints of the problem. Taking advantage of the structure `s`, `v` and `p`, it is possible to write constraints following rigorously the mathematical formulation of the problem.

Listing A.2: variables.m

```
%% INTEGER OPERATION VARIABLES
% On-off and startup variables
for i = s.I
    for j = s.J{i}
        dvar('z','bin',"TYP","T",i,j,"U",'set.binop')
        dvar('delta','bin',"TYP","T",i,j,"U",'set.binop'
            )
    end
end
%% DESIGN VARIABLES
for i = s.I
    for j = s.J{i}
        dvar('gamma','bin',i,j,'set.design') % Catalogue
            selection
    end
end
for i = s.I
    for j = s.J{i}
        dvar('phi','bin',i,j,"U",'set.design')   % Unit
            selection
    end
end
% Maximum electricity demand
dvar('max_el','real','set.design')
% Maximum fuel consumption
dvar('max_fuel','real','set.design')
%% REAL OPERATION VARIABLES
% Total input and output for the technology i
dvar('in','real',"TYP","T","I")
dvar('out','real',"TYP","T","I")
dvar('out2','real',"TYP","T","I(1)")
```

Listing A.3: parameters.m

```
%% Performance characteristics
% for each capacity j
% of each technology i
for i = s.I
    for j=s.J{i}
        dpar('m',i,j)   % Slope
        dpar('q',i,j)   % Intercept
        dpar('m_T',i,j)% Temperature slope
        dpar('xl',i,j)  % Lower limit
        dpar('xu',i,j)  % Upper limit
        dpar('yu',i,j)  % Nominal output
        dpar('rampup',i,j) % Rampup
        dpar('rampdown',i,j) % Rampdown
        dpar('su',i,j) % Surtup consumptions
        dpar('minup',i,j) % Minimum up-time
    end
end
%% Energy input cost
% for each technology i at time t in T, of typical
   period n in TYP
phi(:,:,1) = cost_ng;
phi(:,:,2) = cost_ng;
phi(:,:,3) = 0;
phi(:,:,4) = 0;
dpar('phi',"TYP","T","I")
```

Listing A.4: constraints.m

```
%% Performance characteristics
%input-output relationship
for i = s.I
    for j = s.J{i}
        for u = s.U
            constr('v.y(s.TYP,s.T,i,j,u) == p.m(i,j) .*
               v.xty(s.TYP,s.T,i,j,u)  + (p.q(i,j)+p.m_T
               (i,j)*p.Temp(s.TYP,s.T)).* v.z(s.TYP,s.T,
               i,j,u)');
            constr(' p.xl(i,j) * v.z(s.TYP,s.T,i,j,u) <=
                v.xty(s.TYP,s.T,i,j,u) <= p.xu(i,j)*v.z(
               s.TYP,s.T,i,j,u) ');
        end
    end
end
%% Maximum fuel demand
constr('v.max_fuel >= v.in(s.TYP,s.T,s.I(1)) + v.in(s.
   TYP,s.T,s.I(2))')
%% Maximum electricity demand
constr('v.max_el >= v.el(s.TYP,s.T)')
```

Listing A.5: objective.m

```
Objective =  v.fD + v.fD_star + sum(v.f0);
```

Listing A.6: options.m

```
initialize
% Select solver
whatsolver = 'cplex';
%Absolute gap of the solver (default is 1e-4)
%N.B. only applied if solved in MATLAB
gap = 1e-4;
% Number of periods
nclus = 14;
% Length of typical period [h]
TP_length = 24;
% User defined branching priorities
usepriority = 1;
% 0: Export conventional, 1: Export hierarchical
   decomposition, 2: Export both conventional and
   hierarchical decomposition
export = 2;
% 0: Conventional B&B, 1: Solve decomposed problem
   calling Python from MATLAB
solve_decomposed = 1;
% Switch on opportunistic mode (0 for default, -1 for
   opportunistic, 1 for deterministic)
parallel_opportunistic = 1;
% Set number of threads (0 for default)
N_threads = 1;
```

Appendix B

Complete mathematical model of the Bovisa Case Study

Sets

$I := \{\text{GT, AB, EC, AC}\}$	Available technologies
$J_i := \{\text{M}_{1\text{i}}, \text{M}_{2\text{i}}..\text{M}_{\text{Ni}}\}$	Models in the catalogue for each technology $i \in I$
$U := \{1,2,3,4\}$	Potential installable units for each model $j \in J_i$
$TYP := \{1,2,3,4....\text{N}_{\text{typ}}\}$	Typical days
$T := \{1,2,3,4...24\}$	Hourly time step of each period

Variables

Binary/Integer Design Variables

$\hat{\gamma}_{i,j}$:= Selection variable for the model j of the technology i

$\forall j \in J_i, i \in I$

$\hat{\varphi}_{i,j,u}$:= Investment variable for the unit u of the j^{th} model

$\forall u \in U, j \in J_i, i \in I$

Binary/Integer Operation Variables

$\hat{z}_{i,j,u,n,t}$:= On-off variable for u$^{\text{th}}$ unit at each time step t of the typical period n

$\forall t \in T, n \in TYP, u \in U, j \in J_i, i \in I$

$\hat{\delta}_{i,j,u,n,t}$:= Startup variable for u$^{\text{th}}$ unit at each time step t of the typical period n

$\forall t \in T, n \in TYP, u \in U, j \in J_i, i \in I$

Continuous Design Variables

$\hat{el}^{max}$:= Maximum power required to the national electric grid [MW]

$\hat{fu}^{max}$:= Maximum fuel consumption rate required to the natural gas network [MW]

Continuous Operation Variables

$\hat{y}_{i,j,u,n,t}$:= Primary energy output of the u$^{\text{th}}$ unit at each time step t of the typical period n

$\forall t \in T, n \in TYP, u \in U, j \in J_i, i \in I$

$\hat{y}^2_{GT,j,u,n,t}$:= Secondary energy output of the u$^{\text{th}}$ GT unit at each time step t of the typical period

$\forall t \in T, n \in TYP, u \in U, j \in J_{GT}$

$\hat{x}^{ty}_{i,j,u,n,t}$:= Energy input of the u$^{\text{th}}$ unit at each time step t of the typical period n

$\forall t \in T, n \in TYP, u \in U, j \in J_i, i \in I$

$\hat{x}_{i,j,u,n,t}$:= Energy input of the u$^{\text{th}}$ unit accounting for start-up consumptions

$\forall t \in T, n \in TYP, u \in U, j \in J_i, i \in I$

$\hat{out}_{i,n,t}$:= Total primary energy output from the i$^{\text{th}}$ technology

$\forall t \in T, n \in TYP, i \in I$

$\hat{out}^2_{GT,n,t}$:= Total secondary energy output from gas turbines

$\forall t \in T, n \in TYP$

$\hat{in}_{i,n,t}$:= Total energy input for the i$^{\text{th}}$ technology

$\forall t \in T, n \in TYP, u \in U, j \in J_i, i \in I$

$\hat{el}_{n,t}$:= Purchased electricity from the grid

Objective Function

$\hat{f}_D :=$ Part of the objective function depending on integer design variables

$\hat{f}_{D*} :=$ Part of the objective function depending on continuous design variables

$\hat{f}_{O_n} :=$ Part of the objective function depending on operation variables of the typical period n

$\forall n \in TYP$

Parameters

Energy demand and weather prediction

$d^{el}_{n,t} :=$ Electricity demand	$\forall t \in T, n \in TYP$
$d^{ht}_{n,t} :=$ Heating demand	$\forall t \in T, n \in TYP$
$d^{co}_{n,t} :=$ Cooling demand	$\forall t \in T, n \in TYP$
$\vartheta_{n,t} :=$ Temperature profile	$\forall t \in T, n \in TYP$
$H_n :=$ Yearly weight of the typical period n according to the clustering	$n \in TYP$

Machine operation characteristics

$m_{i,j} :=$ Performance slope	$\forall j \in J_i, i \in I$
$q_{i,j} :=$ Performance intercept	$\forall j \in J_i, i \in I$
$m^T_{i,j} :=$ Performance temprerature dependence	$\forall j \in J_i, i \in I$
$m^2_{GT,j} :=$ Performance slope for the GT seconday output	$\forall j \in J_{GT}$
$q^2_{GT,j} :=$ Performance intercept for the GT seconday output	$\forall j \in J_{GT}$
$m^{2T}_{GT,j} :=$ Performance temperature dependence for the GT seconday output	$\forall j \in J_{GT}$
$x^l_{i,j} :=$ Minimum input	$\forall j \in J_i, i \in I$
$x^u_{i,j} :=$ Maximum input	$\forall j \in J_i, i \in I$
$y^u_{i,j} :=$ Nominal output	$\forall j \in J_i, i \in I$
$ru_{i,j} :=$ Rump-up limit	$\forall j \in J_i, i \in I$
$rd_{i,j} :=$ Rump-down limit	$\forall j \in J_i, i \in I$
$su_{i,j} :=$ Start-up consumption	$\forall j \in J_i, i \in I$

Cost parameters

$IC_{i,j}$:= Investment cost $\forall j \in J_i, i \in I$

CRF := Capital Recovery Factor

C^{el} := Cost of connection to the national electric grid

C^{fu} := Cost of connection to the natura gas network

$c^{fu}_{i,n,t}$:= Cost of the fuel per MWh $\forall t \in T, n \in TYP, i \in FU$

$c^{el}_{n,t}$:= Market price of the elctricit $\forall t \in T, n \in TYP$

Constraints

Catalogue Selection and Investment Decision

$$\sum_i \hat{\gamma}_{i,j} \leq 1 \qquad \forall j \in J_i, i \in I$$

$$\hat{\varphi}_{i,j,1} = \hat{\gamma}_{i,j} \qquad \forall j \in J_i, i \in I$$

$$\sum_{n \in TYP} \sum_{t \in T} \hat{z}_{i,j,u,n,t} \leq \varphi_{i,j,u} \cdot \mathbb{N}_{ty} \cdot 24 \qquad \forall u \in U, j \in J_i, i \in I$$

Prioritization Constraints

$$\hat{\varphi}_{i,j,u} \leq \hat{\varphi}_{i,j,u-1} \qquad \forall u \in U \setminus \{1\}, j \in J_i, i \in I$$

$$\hat{z}_{i,j,u,n,t} \leq \hat{z}_{i,j,u-1,n,t} \qquad \forall t, \in T, n \in TYP, u \in U \setminus \{1\}, j \in J_i, i \in I$$

Input-output relationships

$$\hat{y}_{i,j,u,n,t} = m_{i,j}\hat{x}^{ty}_{i,j,u,n,t} + (q_{i,j} + m^T_{i,j}\vartheta_{n,t})\hat{z}_{i,j,u,n,t} \qquad \forall t \in T, n \in TYP, u \in U, j \in J_i, i \in I$$

$$x^l_{i,j}\hat{z}_{i,j,u,n,t} \leq x^{ty}_{i,j,u,n,t} \leq x^u_{i,j}\hat{z}_{i,j,u,n,t} \qquad \forall t \in T, n \in TYP, u \in U, j \in J_i, i \in I$$

$$\hat{out}_{i,n,t} = \sum_{j \in J_i} \sum_{u \in U} \hat{y}_{i,j,u,n,t} \qquad \forall t \in T, n \in TYP, i \in I$$

$$\hat{in}_{i,n,t} = \sum_{j \in J_i} \sum_{u \in U} \hat{x}_{i,j,u,n,t} \qquad \forall t \in T, n \in TYP, i \in I$$

$$\hat{out}^2_{GT,n,t} = \sum_{j \in J_i} \sum_{u \in U} \hat{y}^2_{GT,j,u,n,t} \qquad \forall t \in T, n \in TYP$$

Start-up, minimum up time and ramp limits

$$\hat{\delta}_{i,j,u,n,t} \geq \hat{z}_{i,j,u,n,t} - \hat{z}_{i,,ju,n,t-1} \qquad \forall t \in T, n \in TYP, u \in U \setminus \{1\}, j \in J_i, i \in I$$

$$\hat{x}_{i,j,u,n,t} = \hat{x}^{ty}_{i,j,u,n,t} + SU\hat{\delta}_{i,j,u,n,t} \qquad \forall t \in T, n \in TYP, u \in U \setminus \{1\}, j \in J_i, i \in I$$

$$\hat{y}_{i,j,u,n,t-1} - \hat{y}_{i,j,u,n,t} \leq ru_i + (1 - \hat{z}_{i,j,u,n,t})M \qquad \forall t \in T \setminus \{1\}, n \in TYP, u \in U, j \in J_i, i \in I$$

$$\hat{y}_{i,j,u,n,t} - \hat{y}_{i,j,u,n,t-1} \leq rd_i + (1 - \hat{z}_{i,j,u,n,t})M \qquad \forall t \in T \setminus \{1\}, n \in TYP, u \in U, j \in J_i, i \in I$$

$$\sum_{t-t^{min}+1}^{t} \hat{z}_{i,j,u,n,t} \geq t^{min}\hat{\delta}_{i,j,u,n,t-t^{min}+1} \qquad \forall t \in \{t^{min}, \ldots, T^{end}\}, n \in TYP, u \in U, j \in J_i, i \in I$$

Energy balances

$$\sum_{i \in OUT^{el}} \hat{out}_{i,n,t} - \sum_{i \in IN^{el}} \hat{in}_{i,n,t} \geq d^{el}_{n,t} - \hat{el}_{n,t} \qquad \forall t \in T, n \in TYP$$

$$\sum_{i \in OUT^{ht}} \hat{out}_{i,n,t} + \hat{out}^2_{GT,n,t} - \sum_{i \in IN^{ht}} \hat{in}_{i,n,t} \geq d^{ht}_{n,t} \qquad \forall t \in T, n \in TYP$$

$$\sum_{i \in OUT^{co}} \hat{out}_{i,n,t} - \sum_{i \in IN^{co}} \hat{in}_{i,n,t} \geq d^{co}_{n,t} \qquad \forall t \in T, n \in TYP$$

Maximum fuel consumpiton and power from the grid

$$\hat{el}_{n,t} \leq \hat{el}^{max} \qquad \forall t \in T, n \in TYP$$

$$\sum_{i \in FU} \hat{in}_{i,n,t} \leq fu^{max} \qquad \forall t \in T, n \in TYP$$

Objective function

$$\hat{f}_D = \sum_{i \in I} \sum_{j \in J_i} \sum_{u \in U} C_{i,j}\hat{\varphi}_{i,j,u} CRF$$

$$\hat{f}_{D*} = \hat{el}^{max} C^{el} + \hat{fu}^{max} C^{fu}$$

$$\hat{f}_{O_n} = \left(\sum_{i \in FU} \sum_{t \in T} \hat{in}_{i,n,t} c^{fu}_{n,t} + \sum_{t \in T} \hat{el}_{n,t} c^{el}_{n,t} \right) \frac{8760}{H_n} \qquad \forall n \in TYP$$

$$\min \hat{f} = \hat{f}_D + \hat{f}_{D*} + \sum_{n \in TYP} \hat{f}_{O_n}$$

Acronyms

MES Multi-Energy System

Multi-Energy System consists in the integration of distributed energy resources and multiple energy loads operating as a single, autonomous grid either in parallel to or "islanded" from the existing utility grid.
[2]

RES Renewable Energy Sources

Renewable energy sources, also called renewables, are energy sources that replenish (or renew) themselves naturally. Typical examples are solar energy, wind and biomass.
ec.europa.eu

CHP Combined Heat and Power

CHP is the production of both heat and electricity from the same device or power plant. By capturing the excess heat, CHP allows a more total use of energy than conventional generation, potentially reaching an efficiency of 70-90 percent.
collinsdictionary.com

DH District Heating

District heating (also known as heat networks or teleheating) is a system for distributing heat generated in a centralized location through a system of insulated pipes for residential and commercial heating requirements such as space heating and water heating.
wikipedia.org

UC Unit Commitment

The unit commitment problem in electrical power production is a large family of mathematical optimization problems where the production of a set of electrical generators is coordinated in order to achieve some common target, usually either match the energy demand at minimum cost or maximize revenues from energy production.
wikipedia.org

ED Economic Dispatch

Economic dispatch is the short-term determination of the optimal output of a number of electricity generation facilities, to meet the system load, at the lowest possible cost, subject to transmission and operational constraints.
wikipedia.org

TL Two-layer (optimization)

Two-layer optimization frameworks are generally composed of two nested algorithms. The outer loop generates potential design solution, while the inner loop optimize the UC and estimate the operating expenses (OPEX) for the corresponding system configuration.
[23]

OS One-shot (optimization)

One-shot optimization frameworks make use of a single model containing both design and operation variables. System sizing and dispatch are solved simultaneously in a single optimization problem.
[23]

PSO Particle Swarm Optimization

Particle swarm optimization is a robust evolutionary strategy inspired by the social behavior of animal species living in large colonies like birds, ants or fish.
sciencedirect.com

GA Genetic Algorithm

A genetic algorithm is a search heuristic that is inspired by Charles Darwin's theory of natural evolution. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation.
medium.com

LP Linear Programming

Linear programming is a method to achieve the best outcome (such as maximum profit or lowest cost) in a mathematical model whose requirements are represented by linear relationships. Linear programming is a special case of mathematical programming (also known as mathematical optimization).
en.wikipedia.org

MILP Mixed-Integer Linear Programming

Mixed integer-linear programs are linear programs in which some variables are required to take integer values, and arise naturally in many applications. The integer variables may come from the nature of the products (e.g., a machine may, or may not, be rented).
www.ibm.com

B&B Branch and Bound

Branch and Bound is an algorithm design paradigm for discrete and combinatorial optimization problems, as well as mathematical optimization. A branch-and-bound algorithm consists of a systematic enumeration of candidate solutions by means of state space search: the set of candidate solutions is thought of as forming a rooted tree with the full set at the root.
en.wikipedia.org

API Application Programming Interface

An application programming interface (API) is an interface or communication protocol between a client and a server intended to simplify the building of client-side software. It has been described as a "contract" between the client and the server, such that if the client makes a request in a specific format, it will always get a response in a specific format or initiate a defined action.
en.wikipedia.org

ICE Internal Combustion Engine

Combustion, also known as burning, is the basic chemical process of releasing energy from a fuel and air mixture. In an internal combustion engine (ICE), the ignition and combustion of the fuel occurs within the engine itself. The engine then partially converts the energy from the combustion to work.
energy.gov

PV Photovoltaic (panels)

Photovoltaics (PV) is the conversion of light into electricity using semiconducting materials that exhibit the photovoltaic effect, a phenomenon studied in physics, photochemistry, and electrochemistry. A photovoltaic system employs solar modules, each comprising a number of solar cells, which generate electrical power. PV installations may be ground-mounted, rooftop mounted, wall mounted or floating.
wikipedia.org

Bibliography

Cited references

Pubblications and Books

[2] Peter Asmus. 'Microgrids, Virtual Power Plants and Our Distributed Energy Future'. In: *Electricity Journal* 23.10 (2010), pp. 72–82. ISSN: 10406190. URL: http://dx.doi.org/10.1016/j.tej.2010.11.001 (cit. on pp. 2, 125).

[3] P. Mancarella. *Smart multi-energy grid: concepts, benefits and challenges.* In: IEEE PES general meeting. San Diego, US. 2012 (cit. on p. 3).

[4] IRENA International Renewable Energy Agency. *Global Energy Transformation: A Roadmap to 2050*. 2019, p. 52. ISBN: 978-92-9260-059-4. URL: http://irena.org/publications/2018/Apr/Global-Energy-Transition-A-Roadmap-to-2050%7B%5C%%7D0Awww.irena.org (cit. on p. 1).

[5] Gianfranco Chicco and Pierluigi Mancarella. 'Distributed multi-generation: A comprehensive view'. In: *Renewable and Sustainable Energy Reviews* 13.3 (2009), pp. 535–551. ISSN: 13640321 (cit. on p. 3).

[7] Pierluigi Mancarella. 'MES (multi-energy systems): An overview of concepts and evaluation models'. In: *Energy* 65 (2014), pp. 1–17. ISSN: 03605442. URL: http://dx.doi.org/10.1016/j.energy.2013.10.041 (cit. on pp. 3, 4).

[8] Silke van Dyken, Bjorn H. Bakken, and Hans I. Skjelbred. 'Linear mixed-integer models for biomass supply chains with transport, storage and processing'. In: *Energy* 35.3 (2010), pp. 1338–1350. ISSN: 03605442. URL: http://dx.doi.org/10.1016/j.energy.2009.11.017 (cit. on p. 3).

[9] IRENA International Renewable Energy Agency. *Innovation Outlook Mini-Grids*. 2016. ISBN: 9789295111431 (cit. on p. 4).

[10] Sahand Ghavidel et al. 'A review on the virtual power plant: Components and operation systems'. In: Sept. 2016, pp. 1–6 (cit. on p. 4).

[11] Carlo Cecati et al. 'An overview on the Smart Grid concept'. In: Dec. 2010, pp. 3322–3327 (cit. on p. 4).

[12] Martin Geidl et al. 'Energy hubs for the future'. In: *IEEE Power and Energy Magazine* 5.1 (2007), pp. 24–30. ISSN: 15407977 (cit. on p. 6).

[13] Göran Andersson and Martin Geidl. 'Optimal Power Flow of Multiple Energy Carriers'. In: *IEEE Transactions on Power Systems* 22.1 (2007), pp. 145–155 (cit. on p. 6).

[14] Lukas Kriechbaum, Gerhild Scheiber, and Thomas Kienberger. 'Grid-based multi-energy systems-modelling, assessment, open source modelling frameworks and challenges'. In: *Energy, Sustainability and Society* 8.1 (2018). ISSN: 21920567 (cit. on pp. 5, 6).

[15] M. Geidl and G. Andersson. 'Operational and topological optimization of multi-carrier energy systems'. In: *2005 International Conference on Future Power Systems*. Nov. 2005, (cit. on p. 6).

[16] Ga Koeppel. 'Reliability considerations of future energy systems: multi-carrier systems and the effect of energy storage'. In: 17058 (2007), p. 249. URL: http://www.eeh.ee.ethz.ch/uploads/tx%7B%5C_%7Dethpublications/eth-diss-17058.pdf (cit. on p. 6).

[17] AD Hawkes and MA Leach. 'Impacts of Temporal Precision in Optimisation Modelling of Micro Combined Heat and Power'. In: *Energy* 30 (2005), pp. 1759–1779 (cit. on p. 7).

[18] Juan Manuel Esca and Francisco Gonzalez-longatt. 'Stochastic Unit Commitment in Microgrids based on Model Predictive Control'. In: (2018) (cit. on p. 8).

[19] M. Y. Nguyen, Y. T. Yoon, and N. H. Choi. 'Dynamic programming formulation of Micro-Grid operation with heat and electricity constraints'. In: *2009 Transmission Distribution Conference Exposition: Asia and Pacific*. Oct. 2009, pp. 1–4 (cit. on p. 8).

[20] Luca Moretti et al. 'A design and dispatch optimization algorithm based on mixed integer linear programming for rural electrification'. In: *Applied Energy* 233-234.August 2018 (2019), pp. 1104–1121. ISSN: 0306-2619. URL: https://doi.org/10.1016/j.apenergy.2018.09.194 (cit. on pp. 9, 13, 38).

[21] Munir Husein and Il-yop Chung. 'Day-Ahead Solar Irradiance Forecasting for Microgrids Using a Long Short-Term Memory Recurrent Neural Network : A Deep Learning Approach'. In: (2019) (cit. on p. 9).

[22] Hao Liang and Weihua Zhuang. 'Stochastic Modeling and Optimization in a Microgrid: A Survey'. In: *Energies* 7 (Apr. 2014), pp. 2027–2050 (cit. on p. 9).

[23] Rémy Rigo-mariani et al. 'Comparison of optimization frameworks for the design of a multi-energy microgrid'. In: *Applied Energy* 257.July 2019 (2020), p. 113982. ISSN: 0306-2619. URL: https://doi.org/10.1016/j.apenergy.2019.113982 (cit. on pp. 11, 12, 126).

[24] H. K. Fathy et al. 'On the coupling between the plant and controller optimization problems'. In: *Proceedings of the 2001 American Control Conference. (Cat. No.01CH37148)*. Vol. 3. June 2001, 1864–1869 vol.3 (cit. on p. 10).

[25] Xiongwen Zhang et al. 'Components sizing of hybrid energy systems via the optimization of power dispatch simulations'. In: 52 (2013), pp. 165–172 (cit. on p. 11).

[26] J. Kennedy and R. Eberhart. 'Particle swarm optimization'. In: *Proceedings of ICNN'95 - International Conference on Neural Networks*. Vol. 4. Nov. 1995, 1942–1948 vol.4 (cit. on p. 11).

[28] Stefano Mazzoni et al. 'Energy storage technologies as techno-economic parameters for master- planning and optimal dispatch in smart multi energy systems'. In: *Applied Energy* 254 (Aug. 2019) (cit. on p. 12).

[29] Davide Fioriti et al. 'Comparison among deterministic methods to design rural mini-grids : effect of operating strategies'. In: *2019 IEEE Milan PowerTech* (), pp. 1–6 (cit. on p. 12).

[30] Bei Li, Robin Roche, and Abdellatif Miraoui. 'Microgrid sizing with combined evolutionary algorithm and MILP unit commitment'. In: *Applied Energy* 188 (2017), pp. 547–562. ISSN: 0306-2619. URL: http://dx.doi.org/10.1016/j.apenergy.2016.12.038 (cit. on p. 12).

[31] Davide Fioriti et al. 'Stochastic sizing of isolated rural mini-grids , including effects of fuel procurement and operational strategies'. In: *Electric Power Systems Research* 160 (2018), pp. 419–428. ISSN: 0378-7796. URL: https://doi.org/10.1016/j.epsr.2018.03.020 (cit. on pp. 12, 36).

[32] Davide Fioriti and Davide Poli. 'A novel stochastic method to dispatch microgrids using Monte Carlo scenarios'. In: *Electric Power Systems Research* 175.June (2019), p. 105896. ISSN: 0378-7796. URL: https://doi.org/10.1016/j.epsr.2019.105896 (cit. on p. 13).

[33] Adam Hawkes and Matthew Leach. 'Modeling high level system design and unit commitment for a microgrid'. In: *Applied Energy* 86 (July 2009), pp. 1253–1265 (cit. on p. 13).

[34] Aldo Bischi et al. 'A detailed MILP optimization model for combined cooling, heat and power system operation planning'. In: *Energy* 74.C (2014), pp. 12–26. ISSN: 03605442. URL: http://dx.doi.org/10.1016/j.energy.2014.02.042 (cit. on pp. 14, 91).

[35] Luca Moretti et al. 'A design and dispatch optimization algorithm based on mixed integer linear programming for rural electrification'. In: *Applied Energy* 233-234.August 2018 (2019), pp. 1104–1121. ISSN: 0306-2619. URL: https://doi.org/10.1016/j.apenergy.2018.09.194 (cit. on p. 14).

[36] Paolo Gabrielli et al. 'Optimal design of multi-energy systems with seasonal storage'. In: *Applied Energy* 219.May 2017 (2018), pp. 408–424. ISSN: 03062619. URL: https://doi.org/10.1016/j.apenergy.2017.07.142 (cit. on p. 14).

[37] Julia Sachs and Oliver Sawodny. 'Multi-objective three stage design optimization for island microgrids'. In: *Applied Energy* 165 (2016), pp. 789–800. ISSN: 03062619. URL: http://dx.doi.org/10.1016/j.apenergy.2015.12.059 (cit. on p. 14).

[38] K. Heidenberger. 'Dynamic project selection and funding under risk: A decision tree based MILP approach'. In: *European Journal of Operational Research* 95.2 (1996). cited By 33, pp. 284–298 (cit. on p. 15).

[39] T. Öncan and M. Cağirici. 'MILP formulations for the order batching problem in low-level picker-to-part warehouse systems'. In: *IFAC Proceedings Volumes (IFAC-PapersOnline)* 46.9 (2013). cited By 1, pp. 471–476. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84884306965&doi=10.3182%2f20130619-3-RU-3018.00372&partnerID=40&md5=7a8ceba4d71fc1a9a209f682034b5862 (cit. on p. 15).

[40] S. Belil, S. Kemmoé-Tchomté, and N. Tchernev. 'MILP-based approach to mid-term production planning of batch manufacturing environment producing bulk products'. In: *IFAC-PapersOnLine* 51.11 (2018). cited By 0, pp. 1689–1694. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85052876242&doi=10.1016%2fj.ifacol.2018.08.213&partnerID=40&md5=c5a6dc6d58bb0830ec7344462ecbf660 (cit. on p. 15).

[41] C.A. Méndez et al. 'State-of-the-art review of optimization methods for short-term scheduling of batch processes'. In: *Computers and Chemical Engineering* 30.6-7 (2006). cited By 581, pp. 913–946. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-33748102266&doi=10.1016%2fj.compchemeng.2006.02.008&partnerID=40&md5=4c1a9f99a4ead7998db3a0c1f4a9ba44 (cit. on p. 15).

[42] M.G. Earl and R. D'Andrea. 'Iterative MILP methods for vehicle-control problems'. In: *IEEE Transactions on Robotics* 21.6 (2005). cited By 98, pp. 1158–1167. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-29844454811&doi=10.1109%2fTRO.2005.853499&partnerID=40&md5=13b2a14f67f795595faa49743e0beb45 (cit. on p. 15).

[43] A. Bischi et al. 'A rolling-horizon optimization algorithm for the long term operational scheduling of cogeneration systems'. In: *Energy* 184 (2019). cited By 8, pp. 73–90. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85042304353&doi=10.1016%2fj.energy.2017.12.022&partnerID=40&md5=eb41d74c817192bdb2d87b8a8186800d (cit. on p. 15).

[44] George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization*. New York, NY, USA: Wiley-Interscience, 1988. ISBN: 0-471-82819-X (cit. on p. 16).

[45] A. H. Land and A. G. Doig. 'An Automatic Method of Solving Discrete Programming Problems'. In: *Econometrica* 28.3 (1960), pp. 497–520. ISSN: 00129682, 14680262. URL: http://www.jstor.org/stable/1910129 (cit. on p. 17).

[47] Santanu S. Dey and Marco Molinaro. *Theoretical challenges towards cutting-plane selection*. 2018 (cit. on p. 20).

[48] Ralph Gomory. 'Outline of an Algorithm for Integer Solutions to Linear Programs'. In: *Bulletin of the American Mathematical Society* 64 (Sept. 1958), pp. 275–278 (cit. on p. 20).

[49] Manfred Padberg and Giovanni Rinaldi. 'A Branch-and-cut Algorithm for the Resolution of Large-scale Symmetric Traveling Salesman Problems'. In: *SIAM Rev.* 33.1 (Feb. 1991), pp. 60–100. ISSN: 0036-1445. URL: http://dx.doi.org/10.1137/1033004 (cit. on p. 20).

[52] Tobias Achterberg, Timo Berthold, and Gregor Hendel. 'Rounding and Propagation Heuristics for Mixed Integer Programming'. In: Jan. 2012, pp. 71–76 (cit. on p. 20).

[54] J. F. Benders. 'Partitioning Procedures for Solving Mixed-variables Programming Problems'. In: *Numer. Math.* 4.1 (Dec. 1962), pp. 238–252. ISSN: 0029-599X. URL: http://dx.doi.org/10.1007/BF01386316 (cit. on p. 21).

[55] George B. Dantzig and Philip Wolfe. 'Decomposition Principle for Linear Programs'. In: *Operations Research* 8.1 (1960), pp. 101–111. ISSN: 0030364X, 15265463. URL: http://www.jstor.org/stable/167547 (cit. on p. 21).

[56] R. Raman and I.E. Grossmann. 'Relation between MILP modelling and logical inference for chemical process synthesis'. In: *Computers Chemical Engineering* 15.2 (1991), pp. 73–84. ISSN: 0098-1354. URL: http://www.sciencedirect.com/science/article/pii/009813549187007V (cit. on p. 21).

[80] J. Löfberg. 'YALMIP: A toolbox for modeling and optimization in MATLAB'. In: cited By 4569. 2004, pp. 284–289. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-20344396845&partnerID=40&md5=adc6ba5e395f9c0b072613d82dfd8e62 (cit. on p. 25).

[88] L. Guo et al. 'A two-stage optimal planning and design method for combined cooling, heat and power microgrid system'. In: *Energy Conversion and Management* 74 (2013). cited By 143, pp. 433–445. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84881092554&doi=10.1016%2fj.enconman.2013.06.051&partnerID=40&md5=78071e4447e4f1fcaecbe389232a35fa (cit. on p. 36).

[90] Sebastian Goderbauer, Martin Comis, and Felix J.L. Willamowski. 'The synthesis problem of decentralized energy systems is strongly NP-hard'. In: *Computers Chemical Engineering* 124 (2019), pp. 343–349. ISSN: 0098-1354. URL: http://www.sciencedirect.com/science/article/pii/S0098135418302448 (cit. on p. 36).

[91] Lane A. Hemaspaandra. 'SIGACT News Complexity Theory Column 36'. In: *SIGACT News* 33.2 (June 2002), pp. 34–47. ISSN: 0163-5700. URL: http://doi.acm.org/10.1145/564585.564599 (cit. on p. 36).

[93] Karen Lindberg et al. 'Methodology for optimal energy system design of Zero Energy Buildings using mixed-integer linear programming'. In: *Energy and Buildings* 127 (May 2016) (cit. on p. 37).

[94] Sheila Samsatli and Nouri Samsatli. 'A general mixed integer linear programming model for the design and operation of integrated urban energy systems'. In: *Journal of Cleaner Production* 191 (Apr. 2018) (cit. on p. 37).

[95] Gonçalo Ferreira Cardoso et al. 'Battery aging in multi-energy microgrid design using mixed integer linear programming'. In: 2018 (cit. on p. 37).

[96] Paolo Gabrielli et al. 'Optimal design of multi-energy systems with seasonal storage'. In: *Applied Energy* (Oct. 2017) (cit. on p. 38).

[97] Yun Yang, Shijie Zhang, and Yunhan Xiao. 'Optimal design of distributed energy resource systems coupled with energy distribution networks'. In: *Energy* 85 (Apr. 2015) (cit. on p. 38).

[98] C. Weber and Neelkumar Shah. 'Optimisation based design of a district energy system for an eco-town in the United Kingdom'. In: *Fuel and Energy Abstracts* 36 (Feb. 2011), pp. 1292–1308 (cit. on p. 38).

[99] Ed Klotz and Alexandra Newman. 'Practical guidelines for solving difficult mixed integer linear programs'. In: *Surveys in Operations Research and Management Science* 18 (Oct. 2013), pp. 18–32 (cit. on p. 38).

[101] Ted Ralphs and Matthew Galati. 'Decomposition Methods for Integer Programming'. In: Feb. 2011 (cit. on p. 39).

[102] R.R. Iyer and Ignacio Grossmann. 'Synthesis and Operational Planning of Utility Systems for Multiperiod Operation'. In: *Computers Chemical Engineering* 22 (July 1998), pp. 979–993 (cit. on pp. 47, 51, 58, 59, 81, 107).

[103] Ted Ralphs et al. 'Parallel Solvers for Mixed Integer Linear Optimization'. In: *Industrial and Systems Engineering (ISE)* (2016) (cit. on p. 42).

[104] Ryohei Yokoyama et al. 'Optimization of energy supply systems by MILP branch and bound method in consideration of hierarchical relationship between design and operation'. In: *Energy Conversion and Management* 92 (2015), pp. 92–104. ISSN: 01968904. URL: http://dx.doi.org/10.1016/j.enconman.2014.12.020 (cit. on pp. 47, 50, 54, 56–58, 76, 79, 81, 90, 107).

[111] Davis Langdon Engineering Services, ed. *Spoon's Mechanical and Electrical Services Price Book*. Spoon Press, 2011 (cit. on p. 83).

Online Material

[1] European Commission. *Tools and technologies for coordination and integration of the European energy system*. 2017. URL: https://ec.europa.eu/info/funding-tenders/opportunities/portal/screen/opportunities/topic-details/lce-05-2017 (cit. on p. 1).

[46] *Mixed-Integer Programming (MIP) – A Primer on the Basics*. URL: https://www.gurobi.com/resource/mip-basics/ (cit. on p. 17).

[50] *Branch and cut in CPLEX*. URL: https://www.ibm.com/support/knowledgecenter/SSSA5P_12.5.1/ilog.odms.cplex.help/refcppcplex/html/branch.html (cit. on p. 20).

[51] *What are heuristics?* URL: https://www.ibm.com/support/knowledgecenter/SSSA5P_12.7.0/ilog.odms.cplex.help/CPLEX/UsrMan/topics/discr_optim/mip/heuristics/43_heur_defn.html (cit. on p. 20).

[53] *Benders strategy*. URL: https://www.ibm.com/support/knowledgecenter/en/SSSA5P_12.7.1/ilog.odms.cplex.help/CPLEX/Parameters/topics/BendersStrategy.html (cit. on p. 21).

[57] *CPLEX*. URL: https://www.ibm.com/analytics/cplex-optimizer (cit. on p. 21).

[58] *Gurobi*. URL: https://www.gurobi.com (cit. on p. 21).

[59] *BARON*. URL: https://minlp.com/baron (cit. on p. 21).

[60] *Mosek*. URL: https://www.mosek.com/ (cit. on p. 21).

[61] *LINDO*. URL: https://www.lindo.com (cit. on p. 21).

[62] *GLPK*. URL: https://www.gnu.org/software/glpk/ (cit. on p. 21).

[63] *LPsolver*. URL: http://lpsolve.sourceforge.net/5.5/ (cit. on p. 21).

[64] *BLIS*. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.410.8877&rep=rep1&type=pdf (cit. on p. 21).

[65] *CBC*. URL: https://github.com/coin-or/Cbc (cit. on p. 21).

[66] *MINTO*. URL: https://neos-server.org/neos/solvers/milp:MINTO/AMPL.html (cit. on p. 21).

[67] *SCIP*. URL: https://scip.zib.de (cit. on p. 21).

[68] *SYMPHONY*. URL: https://www.coin-or.org/SYMPHONY/index.htm (cit. on p. 21).

[69] *A GAMS Example*. URL: https://www.gams.com/products/simple-example/ (cit. on pp. 22, 23).

[70] *GAMS modeling software*. URL: https://www.gams.com/ (cit. on p. 24).

[71] *AMPL modeling software*. URL: https://ampl.com/ (cit. on p. 24).

[72] *PYOMO*. URL: http://www.pyomo.org (cit. on p. 24).

[73] *YALMIP*. URL: https://yalmip.github.io (cit. on pp. xv, 24).

[74] *JuMP*. URL: https://github.com/JuliaOpt/JuMP.jl (cit. on p. 24).

[75] *PuLP*. URL: https://pythonhosted.org/PuLP (cit. on p. 24).

[76] *ZIMPL*. URL: https://zimpl.zib.de (cit. on p. 24).

[77] *CVX*. URL: http://cvxr.com/cvx/ (cit. on p. 24).

[78] *CVXPY*. URL: https://github.com/cvxgrp/cvxpy (cit. on p. 24).

[79] *TOMLAB*. URL: http://tomopt.com/tomlab/ (cit. on p. 24).

[81] *Thread about sparse ND variables (Google Groups: YALMIP)*. URL: https://groups.google.com/forum/#!searchin/yalmip/Barbato/yalmip/kgkZyOwG5OU/w1-UdUN2CwAJ (cit. on p. 29).

[82] *ndSparse class*. URL: https://www.mathworks.com/matlabcentral/fileexchange/29832-n-dimensional-sparse-arrays#feedbacks (cit. on p. 30).

[83] *Issuing priority orders*. URL: https://www.ibm.com/support/knowledgecenter/en/SSSA5P_12.9.0/ilog.odms.cplex.help/CPLEX/UsrMan/topics/discr_optim/mip/para/50_priority_orders.html (cit. on p. 33).

[84] *More Questions on Setting Branching Priorities in the MATLAB CPLEX API*. URL: https://www.ibm.com/developerworks/community/forums/html/topic?id=e31fc168-b44b-49c3-81eb-042aa0213abb&ps=100 (cit. on p. 33).

[85] *Query and Control Call backs (branch callback) available in MATLAB API?* URL: https://www.ibm.com/developerworks/community/forums/html/topic?id=77777777-0000-0000-0000-000014464324 (cit. on p. 34).

[86] *Call Python from MATLAB*. URL: https://www.mathworks.com/help/matlab/matlab_external/call-python-from-matlab.html (cit. on p. 35).

[87] *Get Started with MATLAB Engine API for Python*. URL: https://www.mathworks.com/help/matlab/matlab_external/get-started-with-matlab-engine-for-python.html (cit. on p. 35).

[92] *Web of Science*. URL: http://apps.webofknowledge.com/ (cit. on p. 36).

[100] *Starting from a solution: MIP start*. URL: https://www.ibm.com/support/knowledgecenter/SSSA5P_12.7.1/ilog.odms.cplex.help/CPLEX/UsrMan/topics/discr_optim/mip/para/49_mipStarts.html (cit. on p. 38).

[105] *What are callbacks?* URL: https://www.ibm.com/support/knowledgecenter/SSSA5P_12.5.0/ilog.odms.cplex.help/CPLEX/UsrMan/topics/progr_adv/callbacks_basic/02_cb_defn.html (cit. on p. 54).

[106] *What's new in CPLEX Optimization Studio 12.8*. URL: https://developer.ibm.com/docloud/blog/2017/10/17/whats-new-cplex-optimization-studio-12-8/ (cit. on p. 79).

[107] *Path invariance and generic callbacks*. URL: https://www.ibm.com/support/knowledgecenter/en/SSSA5P_12.9.0/ilog.odms.cplex.help/CPLEX/UsrMan/topics/progr_adv/callbacks/pathInvarianceCallbacks.html (cit. on p. 79).

[108] *Incumbent callback*. URL: https://www.ibm.com/support/knowledgecenter/SSSA5P_12.9.0/ilog.odms.cplex.help/refpythoncplex/html/cplex.callbacks.IncumbentCallback-class.html (cit. on p. 55).

[110] *User Cut callback*. URL: https://www.ibm.com/support/knowledgecenter/SSSA5P_12.9.0/ilog.odms.cplex.help/refpythoncplex/html/cplex.callbacks.UserCutCallback-class.html (cit. on p. 54).