INTEGRATION OF DIFFERENT MODELS IN THE DESIGN OF CHEMICAL PROCESSES: APPLICATION TO THE DESIGN OF A POWER PLANT

José A. Caballero*; Miguel A. Navarro*, Ignacio E. Grossmann**

*Department of Chemical Engineering. University of Alicante. Ap. Correos 99. 03080 Alicante. Spain

** Department of Chemical Engineering. Carnegie Mellon University. Pittsburgh, PA. USA.

Abstract

With advances in the synthesis and design of chemical processes there is an increasing need for more complex mathematical models with which to screen the alternatives that constitute accurate and reliable process models. Despite the wide availability of sophisticated tools for simulation, optimization and synthesis of chemical processes, the user is frequently interested in using the 'best available model'. However, in practice, these models are usually little more than a black box with a rigid input-output structure. In this paper we propose to tackle all these models using generalized disjunctive programming to capture the numerical characteristics of each model (in equation form, modular, noisy, etc.) and to deal with each of them according to their individual characteristics. The result is a hybrid modular –equation based approach that allows synthesizing complex processes using different models in a robust and reliable way. The capabilities of the proposed approach are discussed with a case study: the design of a utility system power plant that has been decomposed into its constitutive elements, each treated differently numerically. And finally, numerical results and conclusions are presented.

keywords

Process synthesis, Generalized Disjunctive Programming; Utility Systems, Modular Optimization, kriging.

1. Introduction

The model of a chemical plant can be theoretically represented by a large system of nonlinear algebraic equations. However, depending on the data specified and the final objective of the model user four types of different problems are considered [1, 2]:

In a *Simulation Problem*, the feeds and design variables of each unit must be specified. The unknowns are the variables representing the additional (product) streams. They usually have a rigid input-output structure, but at the same time are robust and reliable.

A *Design Problem* is similar to the simulation problem, but some of the design variables (i.e. reactor volume; number of trays in a distillation column, etc) are unspecified. A number of constraints are then imposed on some of the stream variables to satisfy the extra degrees of freedom.

In an *Optimization Problem* some variables associated with the feed and equipment design can be left unspecified, in this case a performance function must be added to the model. Inequality constraints may also be added to the model.

In a *Synthesis Problem*, alongside the optimal operating conditions (feed and design variables), we are also interested in the best configuration, for a given objective, from a structural point of view (combination of unit operations or technologies). We have the added difficulty of solving a problem in which the set of equations change depending on the selected equipment.

The simulation and design problems can be theoretically represented by a large system of nonlinear algebraic equations of the form:

$$\mathbf{f}(\mathbf{x}) = \mathbf{0} \tag{1}$$

where f is a vector of functions and x is a vector of variables. The variables represent flow rates, compositions, temperatures, pressures, etc., and the functions are obtained from physical and chemical principles expressing conservation of mass and energy, chemical equilibrium, kinetics and transport phenomena. Modeling a chemical plant can involve hundreds of thousands of equations and variables. In some cases it is possible to write and solve the complete set of equations directly using general modeling systems (e.g. GAMS [3] , AMPL [4]) or chemical

engineering oriented modeling systems (e.g. ASCEND [5], gPROMS [6]) that include databases of chemical and thermodynamic properties. But as the model becomes more complex more specialized knowledge is required to, for example, provide good initializations, and avoid physically meaningless solutions.

Alternatively, instead of solving all the equations simultaneously, it is possible to use a modular approach. In this case the equations of a given module (i.e. unit operation like a distillation column, heat exchanger, etc.) are solved using tailored numerical algorithms. And then all the modules are solved following a pre-specified calculation sequence. The advantages of a modular approach are:

- Different sub-modules can be prepared and tested separately.
- The solution methods can be specifically designed for that module, e.g. the model of a distillation column or a complex reactor. Therefore, the module is robust and reliable.
- Because of the rigid requirements, data can be easily checked for consistency and completeness.
- The modular structure allows easy addition of new modules.

Due to these advantages, it is not surprising that modular simulation is still the dominant approach. However, when we move to design, optimization or synthesis problems the modular approach loses some of its attractiveness. The straightforward approach consists of performing simulations that attempt to satisfy the design or optimization objectives. However, repeated runs of the simulation rapidly lead to long computational times. Therefore, the design and optimization is usually performed in an equation based environment (all equations solved simultaneously) using general modeling systems or field specific modeling systems [7]. In the case of synthesis, the model takes the form of a Mixed-integer (Non)Linear Programming problem [2, 8, 9] (MINLP) where discrete decisions are related to integer (binary) variables or a Generalized Disjunctive Programming Problem (GDP) [10]

Developments in the design, optimization and synthesis of chemical processes over the last years have been impressive at all levels: from individual unit operations, to subsystems, and even to complete flowsheet optimization (see for instance the following books [8, 11-14]). However, due to the necessity of using equation based approaches, most of these designs rely

on shortcut or aggregated methods [15] or on some assumptions that must be verified using rigorous models (i.e. a chemical process simulator). Moreover, with advances in the synthesis and design of chemical processes there is an increasing need for more complex mathematical models with which to screen the alternatives that constitute accurate and reliable process models. Due to the complexity of the models it is not practical, and perhaps not even possible to write a mathematical model each time we need to use it in a new simulation, design or synthesis problem. Instead we would like to reuse the best available mathematical model [16]. While this is really straightforward in a modular simulation environment (we only have to add one more module), in design, optimization and synthesis problems we should try to avoid the brute force approach of repeated simulations mentioned previously. In this paper we present a modeling framework that captures the specific characteristics of each of these models and allows their efficient utilization in design, optimization and synthesis problems.

However, we do not confine our attention to only chemical process simulators in this paper. It is clear that a number of state of the art models of importance to the chemical engineering community come standard with commercial process simulators. Trying to use chemical process simulators as an external module for solving synthesis problems by a MINLP approach is not something new. Harsh et al [17] developed an interface between a MINLP and FLOWTRAN for the purpose of retrofitting an ammonia process. Diwekar et al [18], devised a process synthesizer using Aspen Plus. They illustrate their method on some small problems and the structural optimization of the hydrodealkylation of the toluene process. Their algorithm is basically an implementation of the modeling and decompositions strategy [19]. Reneaume et al [20] point out that in a given constraint $h(\mathbf{d},\mathbf{s}) \leq 0$ (**d** is a vector of decision variables and **s** are variables of interest calculated by the process simulator) the **s** variables depend implicitly on the **d** variables. But this implicit function varies depending on what the structural decisions are. In other words, **s** depends on the structural decisions and therefore the linearization of a given constraint (e.g. in the construction of the MILP Master Problem) can lead to the linearization of different functions, and consequently to the failure of the algorithm. They solved the problem by adding 'pseudo-torn' streams, the function of which is to explicitly separate the dependency of **s** variables from the structural decisions. Diaz & Bandoni [21] used a process simulator specifically designed for ethylene plants, SISER [22], for the structural design of an ethylene

plant using a combination of different types of models (rigorous and simplified), which includes correlations and results to check against an actual ethylene plant. Caballero et al. [23] proposed a specific algorithm for the rigorous design of distillation columns, combining process simulators and a modified version of the outer approximation algorithm [24-26]. Later Brunet el al. [27] used this algorithm in the optimization of distillation columns in an ammonia water absorption cooling cycle. They also extended the approach to mutiobjective optimization by considering Life Cycle Assessment (LCA), without changing the topology in the process simulator.

In all the above-cited works, the full model is solved by a process simulator interfaced with a MINLP solver. Caballero et al [28] presented a hybrid approach for the design of hybrid distillation vapor membrane separation systems, in which the differential and algebraic equations of the membrane modules are calculated in equation form. Caballero & Grossmann [29] presented a detailed modeling framework that combines a process simulator with complex algebraic (equation based) models involving both continuous and discrete variables, although the topology of the flow sheet was not modified. A similar approach was followed by Brunet el al. [30, 31] who extended the methodology to consider multiobjective optimization by including LCA in biotechnological processes in which the reactions have complex kinetics that cannot be solved by the process simulator.

It is worth mentioning that deterministic optimization methods, like the approach proposed in this paper, are not the only alternative for dealing with these problems. Stochastic methods have proved to be a good alternative for solving hybrid simulation-optimization problems. Although there is a vast literature on metaheuristic optimization, combination with chemical process simulators is a relatively recent development [32-39]. Besides in the case of synthesis other approaches can be used such as local/global optimization techniques [40-42].

In this paper we present a modeling framework for dealing with synthesis problems including models that might come from different sources and exhibit completely different numerical behavior. This includes, but it is not limited to, chemical process simulators, thermodynamic property servers, third party proprietary software and models from computer fluid dynamics, or even experimental models. We believe that disjunctive programming is a framework that is very well suited for dealing with these kinds of problems because it allows «encapsulating» each

model and then following a different approach based on the characteristics of each. The connectivity between those models is in equation form – to avoid the implicit relation between design and calculated variables [20]. Logical relationships, including strong relations between alternatives (i.e., to ensure only feasible solutions) and soft relations (i.e., designer preferences) can be added easily. The model is solved using logic-based algorithms without reformulating it as an MINLP [9, 43]. As far as we know this is the first framework for modeling chemical process that allows a modular approach, combining explicit –equation based- models with simulation (black box) models, each of them with its own numerical characteristics, into a Generalized Disjunctive Programming environment with logic based solvers.

In the rest of the paper we first provide a brief overview of generalized disjunctive programming (GDP). Then we describe the modelling framework, its characteristics and the different numerical treatment alternatives depending on the characteristics of a given model. The capabilities of the proposed approach are illustrated by means of a case study: the design of a utility system power plant that has been decomposed into its constitutive elements, each one requiring different numerical treatment. Finally, numerical results and conclusions are presented.

2. Generalized Disjunctive Programming

The most basic formulation of an optimization problem including binary variables (in the context we are interested, and usually related with a decision, i.e., to install or not a particular process unit) –MINLP- is as follows:

$$\begin{aligned} &\min : Z = f(x,y) \\ &s.t.\ g_j(x,y) \leq 0 \quad j \in J \\ &\quad x \in X \subseteq \Re^n;\ y \in \left\{0,1\right\}^p \end{aligned} \tag{2}$$

An alternative approach for representing discrete and continuous optimization problems is by using models consisting of algebraic constraints, logic disjunctions and logic propositions [44]. This approach not only facilitates the development of the models by making the formulation process intuitive, but also keeps the underlying logical structure of the problem in the model,

which can be exploited to find the solution more efficiently. A particular case of these models is Generalized Disjunctive Programming (GDP).

The general structure of a GDP problem can be represented as follows [45]

$$
\begin{aligned}
&\min Z = f(x) + \sum_{k \in K} c_k \\
&s.t. \quad g(x) \leq 0 \\
&\quad \bigvee_{i \in D_k} \begin{bmatrix} Y_{i,k} \\ r_{i,k}(x) \leq 0 \\ c_k = \gamma_{i,k} \end{bmatrix} \quad k \in K \\
&\quad \Omega(Y) = True \\
&\quad x^{lo} \leq x \leq x^{up} \\
&\quad x \in \Re^n, c_k \in \Re^1 \\
&\quad Y_{i,k} \in \{True, False\}, i \in D_k,\ k \in K
\end{aligned}
\tag{3}
$$

where $f : R^n \rightarrow R^1$ is a function of the continuous variables x in the objective function, $g : R^n \rightarrow R^l$ belongs to the set of global constraints, the disjunctions $k \in K$, are composed of a number of terms $i \in D_k$, that are connected by the OR operator. In each term there is a Boolean variable $Y_{i,k}$, a set of relations $r_{i,k}(x) \leq 0$ and a cost variable c_k. If $Y_{i,k}$ is True, then $r_{i,k} \leq 0$ is enforced; otherwise they are ignored. Also $\Omega(Y) = True$ are logic propositions for the Boolean variables.

In order to take advantage of the existing MINLP solvers, GDPs are often reformulated as an MINLP. To do so, two main transformations should be made, namely the disjunctive constraints must be expressed in terms of algebraic equations and the propositional logic needs to be expressed in terms of linear equations. The disjunctive constraints can be transformed by using either the big-M [46] or the Hull Relaxation [10]. The transformation of propositional logic can be accomplished as described in the work by Willians [47] to get a set of linear equalities and inequality constraints in terms only of binary variables. An important drawback of MINLP reformulation is that all the equations $r_{i,k}(x) \leq 0$ appear in the final formulation, even though they might be inactive, which in some situations could produce numerical problems.

In order to fully exploit the logic structure of GDP problems, two other solution methods have been proposed, namely the Disjunctive Branch and Bound method [10] and the logic based outer approximation method [43]. The basic idea of the disjunctive Branch and Bound method is to directly branch to the constraints corresponding to particular terms in the disjunctions, while considering the convex hull of the remaining disjunctions. Although the tightness of the relaxation at each node is comparable with that obtained when solving the HR reformulation using a MINLP solver, the size of the solved problems is smaller and the numerical robustness is improved, even though we still have to reformulate the problem. The idea underlying the Logic Based Outer Approximation consists of iteratively solving a master problem given by a linear GDP and a nonlinear (NLP), with fixed values of Boolean variables serving as an upper bound. Therefore, for fixed values of the Boolean variables, $Y_{\widehat{i},k} = True; Y_{i,k} = False \;\; \widehat{i} \neq i$ the corresponding NLP is as follows:

$$
\begin{aligned}
&\min Z = f(x) + \sum_{k \in K} c_k \\
&s.t. \;\; g(x) \leq 0 \\
&\qquad \left.\begin{aligned} r_{\widehat{i},k}(x) &\leq 0 \\ c_k &= \gamma_{\widehat{i},k} \end{aligned}\right\} \widehat{i} \in D_k \\
&\quad x^{lo} \leq x \leq x^{up} \\
&\quad x \in R^n,\; c_k \in R^1
\end{aligned}
\tag{4}
$$

It is important to note that only the constraints that belong to the active terms in the disjunction (i.e. associated Boolean variables $Y_{\widehat{i},k} = True$) are imposed.

In the context of process networks, dealt with in this paper, the disjunctions in the GDP are two-termed. Basically the decision in the disjunction is whether to select or not a given alternative (i.e., in the case study presented later, to install or not a gas turbine, a boiler, etc). The second term of the disjunction simply states that if a given option is not selected all the variables related to that option are set to zero.

$$
\begin{bmatrix} Y_i \\ r_i(x) \leq 0 \\ c_i = \gamma_i \end{bmatrix} \vee \begin{bmatrix} \neg Y_i \\ B^i x = 0 \\ c_i = 0 \end{bmatrix}
\tag{5}
$$

For this particular case, the master problem can be written as follows:

$$
\begin{aligned}
&\min\ Z = \alpha + \sum_{k \in K} c_k \\
&s.t.\ \left.\begin{array}{c} \alpha \geq f(x^l) + \nabla f(x^l)(x - x^l) \\ g(x^l) + \nabla g(x^l)(x - x^l) \leq 0 \end{array}\right\} l = 1,2....,L \\
&\left[\begin{array}{c} Y_i \\ r_i(x^l) + \nabla r_i(x^l)(x - x^l) \leq 0\ l \in L_i \\ c_i = \gamma_i \end{array}\right] \vee \left[\begin{array}{c} \neg Y_i \\ B^i x = 0 \\ c_i = 0 \end{array}\right]\ i \in D \qquad (6) \\
&\Omega(Y) = True \\
&\alpha \in R^1
\end{aligned}
$$

The problem in equation (6) can be easily reformulated and solved as a Mixed Integer Linear Programming (MILP) problem.

It should be noted that before applying the above master problem we need at least one linear approximation of each of the terms $i \in D$ in the disjunctions. Here there are two alternatives, select the smallest number of NLP sub-problems that include at least once each disjunction [43] or perform sub-Lagrangian optimization of the non-existing terms with respect to a base case. This constitutes the modeling and decomposition approach [19]. In the appendix A there is a comprehensive description on how to adapt the logic based outer approximation algorithm to implicit models (modular approach).

3. Modeling framework

We present a new modeling environment that is capable of dealing with synthesis problems involving mathematical models from different sources, exhibiting different numerical behavior and with different degrees of end-user access to the original code. As far as we know this is the first time that the intuitive modeling framework provided by GDP, is combined with a modular hybrid simulation-optimization, in which the numerical treatment of each module is different depending on its characteristics. We believe that GDP is the ideal framework for dealing with these problems for at least the following reasons:

i. GDP representation maintains the underlying logic structure of the problem. The formulation of the problem is intuitive.

ii. Each module can be «encapsulated» in a disjunction. In this way all the procedures specific to each module can be isolated from the rest. For example, the generation of accurate derivatives for both the NLP and Master problems might be based on different methodologies in each module.

iii. Different models can be used for the same unit, without the necessity of 'rewriting' the model, simply by calling a different module. The specific characteristics of the new model will be automatically incorporated.

iv. We can use a range of models, from completely explicit (based on equations) to completely implicit (all equations in third party modules like a process simulator), going through mixed approaches in which parts of the model are implicit and others explicit. For example, it is possible to add explicit constraints that affect the behavior of an implicit model directly.

Figure 1 shows a scheme of the actual implementation. It is composed of three main modules: An algebraic modeling language; a module for the evaluation of each external module, and a logic based solver. The modeling language has the following characteristics

- The complete modeling system is developed in Matlab [48]
- Permits indexing of variables, algebraic equations and implicit models. In other words, the same model can be used in different parts of the problem with different values of the parameters and independent variables.
- Use of Boolean variables, disjunctions and logic equations. Allowing the direct formulation of the problem as a disjunctive problem without MINLP reformulation.
- Specific differentiation methods for each sub-model (algebraic, automatic differentiation, finite differences based on complex variables; etc).
- Determination of the sparsity pattern for each individual model and calculation of the global Jacobian Matrix.
- Interfaced with different commercial solvers for NLP, LP, MILP models through Matlab-Tomlab [48, 49], and with our implementations of a simple Branch and bound algorithm, the outer approximation algorithm [24-26], the LP-NLP based branch and bound

algorithm [50] for MINLP models, and disjunctive solvers without MINLP reformulation [43].

- Communication with process simulators and other third party models, except those developed in Matlab, is accomplished by the Windows COM capabilities.

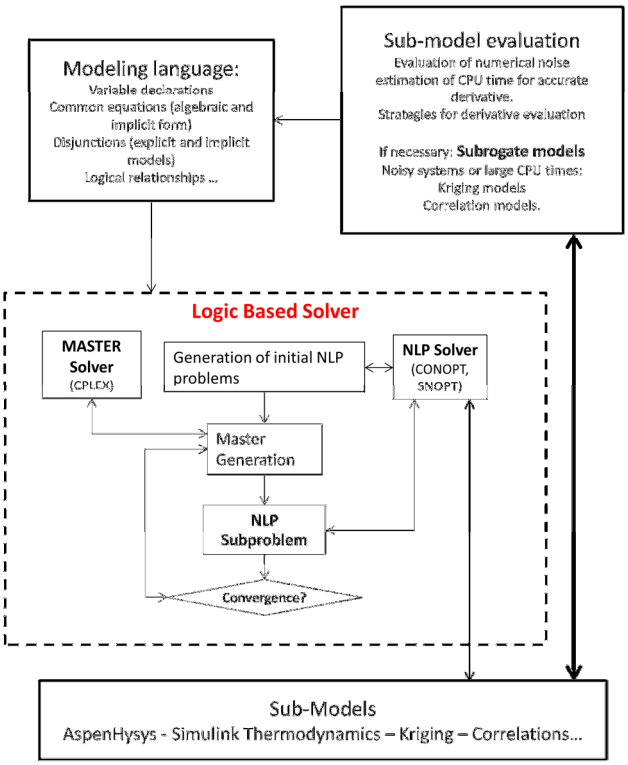

Figure 1. Scheme of the modeling framework

The module that evaluates the external models has the two major difficulties to cope with when external modules are used:

1. Rigid input-output structure. This is the case of almost all chemical process simulators (a remarkable exception is AspenHysys^{TM}) and most modules designed for a specific task. Consider for example a splitter in a process simulator. The user must specify the feed stream and the split fractions, and the module calculates the exit streams. If we can reverse the information flow we may be able to simplify the calculation procedure. In the simulation

of chemical processes the information flow usually coincides with the mass flow. Therefore, if in the model there are recycles we must use an iterative process to solve the problem. In some situations, however, we can reduce the recycle structure of the model by correctly selecting the design variables [51]. If the module does not have a rigid input-output structure (i.e. unit operations in AspenHysys^{TM}) we could eventually take advantage of this fact. The evaluation module performs a 'structural' analysis of the model in order to establish what the best calculation sequence is. If the selection of design variables does not allow reducing the recycle structure of the problem then it is convenient let the optimizer to simultaneously search for the optimum and converges the flow sheet. Even though this approach increases the number of explicit variables seen by the solver it also increases it robustness without sacrificing the performance because in this way we avoid unnecessary iterations in the process simulator.

2. Calculation of accurate derivatives. Accurate derivatives are fundamental for solving any deterministic optimization problem. Depending on the origin or the external module and its characteristics, the following cases are relevant:

    2.1. The model is in equation form (with or without integer variables). Under these conditions first and second derivatives are available (or easy to obtain).

    2.2. We have access to the code of all the external modules. In this case it is possible to automate a set of procedures that generate code for performing operations like automatic differentiation, sparsity pattern determination or discontinuity function evaluation. Tolsma et al [52] implemented these procedures in the modeling system ABACUS II.

    2.3. We have no access to the code that has an input-output structure, but numerical derivative information is available. This is the case of some thermodynamic packages that provide information both about the property and its derivatives with respect to some variables.

    2.4. We have no access to the code and derivative information is not available. Here we can distinguish between different cases

2.4.1. Derivative information can be obtained easily and accurately, estimated using a numerical approach. If the external module admits complex arithmetic we can calculate derivatives virtually without numerical error [53]. In other cases we must use a finite difference scheme by perturbing the independent variables.

2.4.2. A characteristic of external modules is that they introduce numerical noise, e.g. the solution varies slightly with identical initial values. This is common in systems that solve complex numerical equations within a finite tolerance. In process simulators this behavior arises in distillation columns, chemical reactors or other complex operations. If the numerical noise is relatively small, it is still possible to implement a finite difference approach by increasing the perturbation of independent variables, but at the expense of getting approximations of the Jacobian that eventually could produce unexpected behaviors in the solver. Some important additional considerations must be taken into account if derivatives are calculated by finite differences:

A finite difference scheme that employs a noisy model should never be used in a recycle because recycles act as noise amplifiers. This is a very common problem in process flowsheet simulation. If this is the case the tolerances for closing recycles must be at least a couple of orders of magnitude smaller than the perturbation factors. A much better approach consists of letting the NLP solver converge the recycles. Although the number of variables seen by the NLP solver increases and the number of explicit equality constraints also increases, in general the model is more robust and the computational time will usually not increase (e.g., by avoiding converging all the recycles each time the simulation is called). This is the approach we follow in this work.

If the model cannot be solved fast enough (say, in a fraction of a second) the time necessary to calculate derivatives could be very large making the optimization impractical.

2.4.3. If the model is very noisy or the computational time is too large to allow practical implementation, then we cannot use it directly. In this case we can use a shortcut

or aggregated model, but of course this is what we want to avoid from the beginning. Alternatively, it is possible to use a surrogate model. These include, among others, polynomial correlations; splines; neural networks; radial basis functions; kriging models; etc. We have implemented kriging models [54]. A detailed description on the use of kriging models in optimization can be found in Refs. [55-60]. Here we follow the implementation proposed by Caballero & Grossmann [60], which can handle both noisy or deterministic systems by implementing an interpolating or non-interpolating approach depending on the characteristics of the model. It also uses an adaptive approach that contracts or moves the domain between consecutive iterations if necessary in order to ensure accurate results. Finally, there is explicit treatment of constraints in the case of noisy systems.

The following remarks deserve special attention. It is assumed that all the implicit models are continuous and differentiable. Even in noisy systems, the underlying model is assumed to be continuous and differentiable. It is not uncommon for computer models to include "max/min" operators; "If" sentences, etc. that can destroy the differentiability and continuity assumptions. If the user is able to anticipate this behavior, a correct MINLP or GDP reformulation or a disjunctive model can be developed. Otherwise, the numerical behavior of the model could lead to difficulties. All MINLP and GDP algorithms require convexity to guarantee convergence to a global optimal solution. In an implicit model it is difficult to prove convexity, even if the underlying model is convex. Since in general we cannot ensure convexity, there is no guarantee that a global optimum solution can be found.

The evaluation module determines the characteristics of each external module (method of differentiation; if necessary, the optimal perturbation value of each variable that minimizes the effect of numerical noise; or whether or not to transform the original model into a surrogate model). All this information is sent to the modeling language, which generates the model, determines the sparsity pattern, and decides the calculation sequence. Then the logic based solver is executed and the problem is solved. In this paper we have used a logic-based outer approximation algorithm modified to work with external modules (implicit equations). A comprehensive description of this algorithm can be found in appendix A.

4. Case Study

As an example we present the synthesis of a utility system in which it is assumed that different components are simulated by modules that exhibit different numerical behavior. There are different commercial tools that can perform the design very efficiently (Aspen Utility Planner™; Ariane™ by ProSim™) using databases and cost correlations obtained directly from industrial applications. The objective in this work is to show that efficient synthesis of a complex system using different models is possible.

There are a many papers on the optimization and design of utility systems [61-67] ranging from simplified linear models to thermodynamic based approaches, going through detailed studies about startup and operation [68, 69].

Rigorous models for power production of steam turbines have been developed by Mavromatis and Kokossis [70, 71]. Bruno et al. [72] proposed a superstructure optimization formulated as a MINLP based on the previous work by Papoulias & Grossmann [62]. Manninen & Zhu [73] decomposed the problem into a Master problem that specifies major structural features on a design and flowsheet level, in which an exergy analysis identifies relevant modification options. Varbanov et al.[74] developed more accurate models for steam and gas turbines that provide a better description of part-load performance.

In this work we use a superstructure inspired by the previous work of Bruno et al. [72]. Figure 2 shows the superstructure, which includes:

- A gas turbine with a heat recovery steam generator (HRSG) with no additional fuel. The gas turbine can satisfy the electricity demands by supplying exhaust gases to a HRSG boiler that in turn provides vapor to a very high pressure (VHP) header.
- High pressure boilers fired by fuel. Boiler blowdown is necessary in all boilers; in this paper we assume a fixed blowdown rate of 3% of the steam entering the boiler.
- A waste heat boiler that operates at medium pressure and recovers heat from process flue gases or other units such as reactors.
- We assume four pressure headers VHP, HP, MP and LP. In addition, we have to consider a vacuum level for the condensation turbines and an atmospheric level that recovers the water

returned from the process (previously treated). The demineralized water needed to compensate for plant losses and consumption is also assumed to be at atmospheric pressure.

- A de-aerator is included to remove the dissolved gases by steam stripping before feeding the water into boilers. However, extra boiler feedwater could be an option. In this case we assume that the de-aerator uses the LP steam of the plant.
- Steam turbines are used to generate electricity and to satisfy mechanical power needs. Here we consider the following turbines.
    - ✓ Backpressure turbines working between any two pressure levels (VHP-HP; VHP-MP, VHP-LP; HP-MP, HP-LP, MP-LP).
    - ✓ Extraction backpressure turbines exhausting from VHP to HP and MP and from HP to MP and LP.
    - ✓ Condensing turbines from HP and MP.
    - ✓ Extracting condensing turbines from MP to LP and vacuum.
- The steam is distributed to steam consumers, to steam turbines or to the next pressure header through letdown valves.
- Electric motors can be used to meet the required power demands.
- Finally, utility pumps are included to change the pressure in all the liquid streams.

Before continuing with the discussion it is worth introducing some discussion about superstructure based optimization. Superstructure optimization is ideal when we have to deal with a problem with a large number of alternatives for performing the same tasks, because evaluate all those alternatives is not practical (even not possible). Of course, if a device is not in the set of superstructure alternatives, not optimization method can place it there, therefore the designer must use his/her knowledge of the system to generate a compact superstructure that include all the alternatives of interest. Besides, superstructure optimization must be necessarily followed by a critical analysis of the solution. Generation of superstructures is a research field itself [75]. In systems like the one presented here, where the different alternatives can be clearly specified, and the number of possible combinations can be large a superstructure based optimization approach is a good approach.

Before providing a description of the actual implementation of each unit and its characteristics, because we are using a modular approach, it is convenient to analyze the information flow in the flow sheet (or superstructure). If we assume that the information flow coincides with the mass flow, which is the usual approach in modular simulation, then any valid flow sheet must include a recycle involving all unit operations. For example, we should assume a water mass flow rate entering the boilers and fresh water entering the de-aerator, and after calculating the entire flow sheet the water exiting the de-aerator should equal the assumed mass flow rate entering the boilers. As commented before, it is convenient to let the optimizer converge this recycle in order to minimize the effects of numerical noise. However, if we can change the direction of the information flow we could solve a given flow sheet by means of small recycles involving a single unit operation that can be efficiently handled by the implicit model. In this case it can be accomplished just by reversing the calculation in the boilers: The (steam) water mass flow rate is calculated in terms of the fuel flow rate and operating conditions. It is then possible to calculate the fresh water mass flow rate in the de-aerator and no recycles appear in the flow sheet. This kind of analysis can be automated, see for example the book by Westerberg et al [51].

Following is a description of the main equipment involved in the power plant and their particular models for implementation.

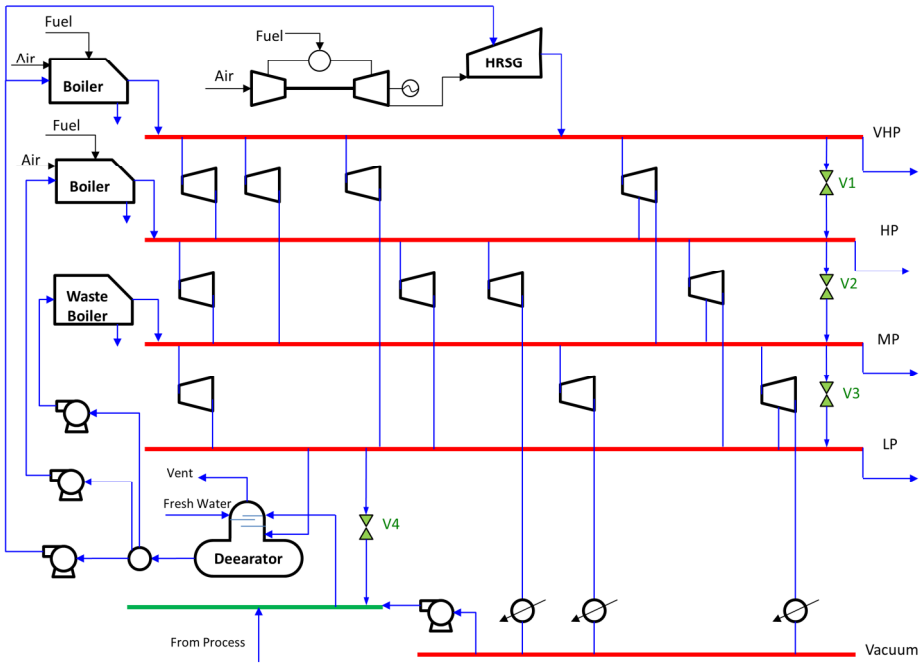

Figure 2. Superstructure of the utility system plant. It is possible to include N turbines of each class in parallel.

Gas turbine and HRSG

The gas turbine is introduced in the model as a kriging metamodel. To generate data that sets up the kriging model [54-56, 60] a gas turbine model was created in Aspen Hysys™. See Figure 3.The model consists of a compressor that receives air at ambient conditions. The resulting compressed air is introduced together with pressurized fuel (natural gas) in a Gibbs reactor that simulates the combustion chamber of the gas turbine. The gases leaving the combustion chamber are introduced in a turbine where they are expanded to atmospheric pressure. The work generated by the gas turbine is used to move the compressor and to generate electricity which is simulated by an energy balance. The exhaust gases are sent to a HRSG that is used to generate more steam, and simulated as a regular heat exchanger. All relevant data about the different equipment units are given in Table 1. Table 2 shows utilities data.

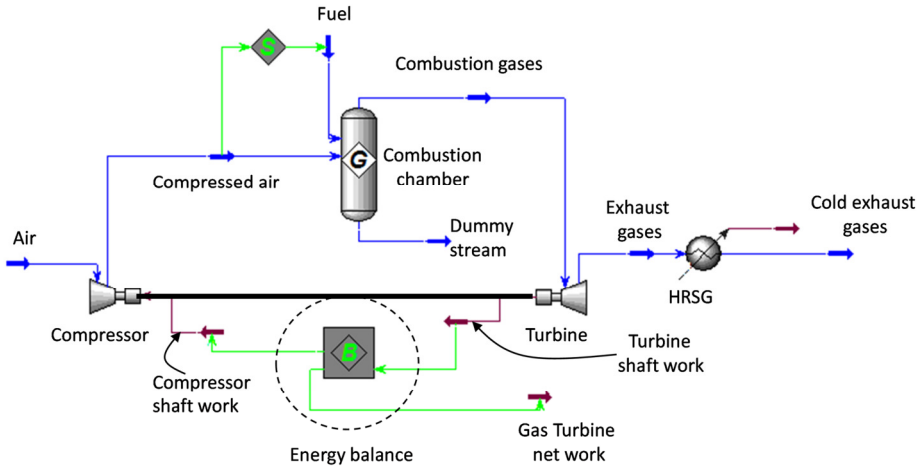

Figure 3. Scheme of the actual implementation of a gas turbine in Aspen-Hysys

Pure thermodynamic based approaches do not match the real cases with a sufficient degree of accuracy. Instead we combine the thermodynamic model with data obtained from actual turbines in order to obtain gas turbine efficiencies that can be included in the model. One of the major factors that affect gas turbine performance is its size. This is measured in terms of rated power production at ISO conditions (1 atm; 15 ºC and 60% relative humidity). Turbines from different manufactures feature different efficiencies for the same size. Data on turbine efficiencies is published by manufactures, for example General Electric reference library documents [76]. Varvanov et al [74] showed that regressing that data set the following equation can be formulated:

$$Q_{f,\max}\,(MW) = 21.9917 + 2.6683\,W_{gt,\max} \tag{7}$$

Using the same set of data the gas turbine efficiencies can be correlated with the following equation:

$$\eta_{GT} = 0.3 + 3.207{\cdot}10^{-4}\,W_{gt,\max}(MW) \tag{8}$$

The gas turbine can be calculated using three independent variables. We have used the air to fuel ratio (wt.), fixed between 40 and 100; the compressor output pressure (between 1000 and 1500 kPa corresponding to compression ratios between 10 and 15 that corresponds with the compression ratios in the data set of actual turbines that are being used as design basis); and the fuel mass flow rate.

In the Aspen-Hysys model the efficiency of the gas turbine is forced to match the correlated efficiency by dynamically modifying the compressor and turbine efficiencies.

The model, as implemented in Aspen Hysys is slightly noisy, and although it slows the optimization down a bit it can be used as is. However, as commented above, for illustrative purposes it was substituted by a kriging metamodel. The kriging approach has the advantage of allowing very fast evaluation of the model. Moreover, it is possible to get numerical derivatives without error (to within computer precision) by using complex derivatives.

To calibrate the kriging model we have used 100 points distributed by a min-max approach (minimizing the maximum distance between two points) based on 1000 kg/h of fuel, which in fact leaves the kriging model with two degrees of freedom. In this case the maximum error introduced by the kriging interpolation is under 0.8%. Figure 4 shows the values predicted by the kriging model, those calculated by Aspen-Hysys and the error for 500 random points.

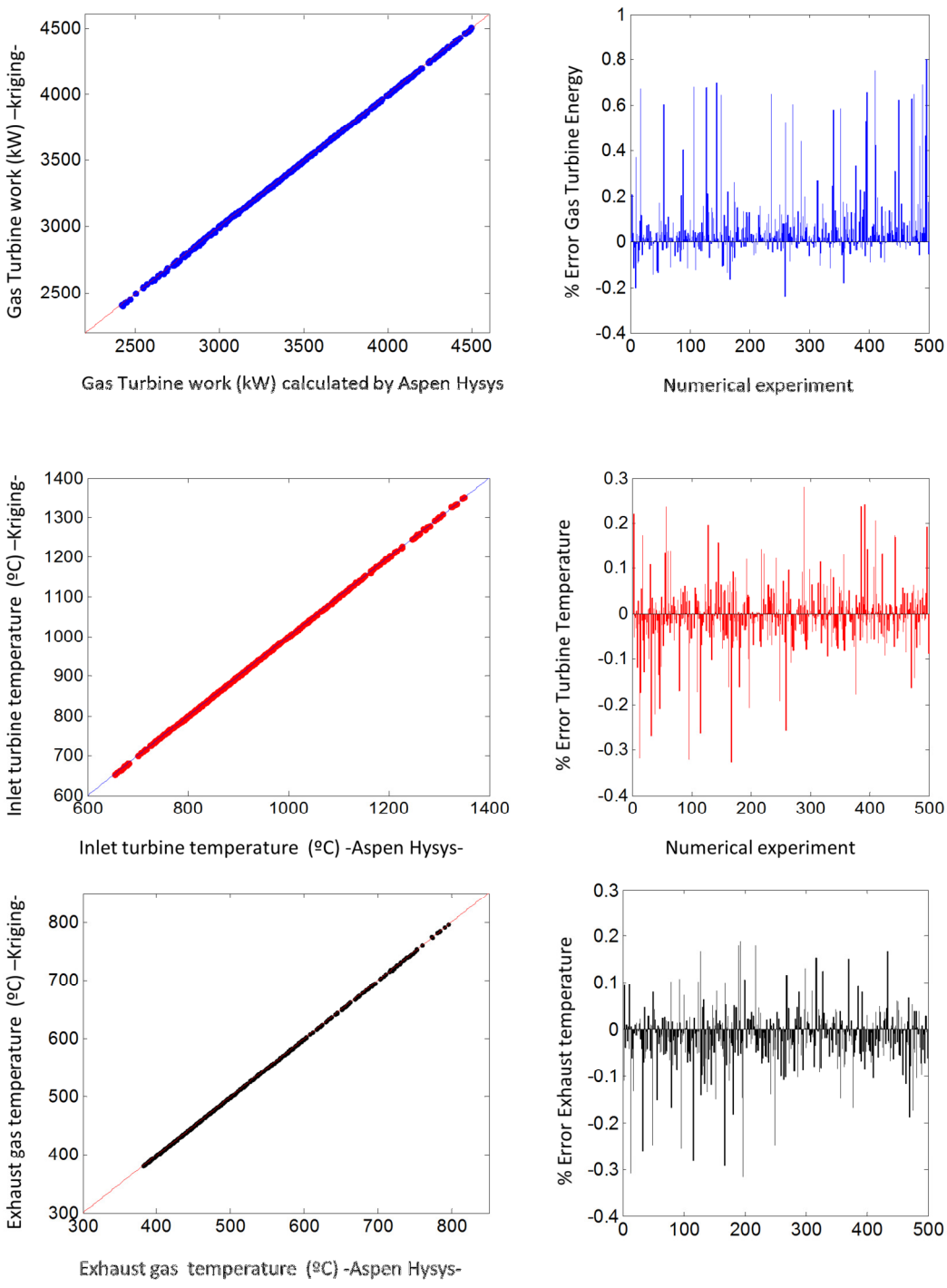

Figure 4. Left: calculated values and those predicted by the kriging model, for the gas turbine.Right, % kriging error of 500 random points.

Therefore, the model for the gas turbine can be written as follows:

$$\begin{aligned}
&\left[W_{gt}, T^{out}_{combustion}, T^{out}_{turbine}, Q_{HRSG}, T^{stack}_{HRSG}\right] = Kriging_Turbine(AF_{gt}, P^{out}_{compressor}, Fuel_{gt}) \\
&40 \le AF \le 100 \\
&1000 \le P^{out}_{compressor} \le 1500\ (kPa) \\
&T^{out}_{combustion} \le 1200\ (^{o}C) \\
&T^{out}_{turbine} \le 600\,(^{o}C) \\
&T^{out}_{HRSG} \ge 160\ (^{o}C)
\end{aligned} \tag{9}$$

where:

AF_{gt}	Gas Turbine air to fuel ratio (weight basis)
$P^{out}_{compressor}$	Output pressure from the compressor in the gas turbine (kPa)
$Fuel_{gt}$	Gas turbine mass fuel flow rate (kg/h)
W_{gt}	Effective work provided by the gas turbine
$T^{out}_{combustion}$	Temperature of the output stream from the combustion chamber in the gas turbine
$T^{out}_{turbine}$	Temperature of the exhaust gases from the gas turbine
Q_{HRSG}	Maximum heat available from the Heat Recovery Steam Generator
T^{stack}_{HRSG}	Temperature for the gases that exit from the HRSG.

For the HRSG we have to decide at which pressure we want to generate the extra steam and what the minimum approach temperature between the gases and steam has to be. By means of an energy balance we can calculate the extra steam mass flow rate and conditions (pressure and temperature). This calculation is performed using Hysys as a thermodynamic calculator for estimating specific water and steam enthalpies at the inlet and outlet conditions. Alternatively, it is possible to generate a flow sheet in Aspen Hysys to perform this calculation. However, the first option avoids interaction with the graphical interface and calculations are faster as a result. The model for the HRSG can be written as follows

$$\left[F_{HRSG}^{Steam}, T_{HRSG}^{Steam}\right] = HRSG\left(Q_{HRSG}, T_{HRSG}^{outgas}, P_{HRSG}, \Delta T_{HRSG}\right) \quad (10)$$

where:

P_{HRSG} Steam pressure in the HRSG (kPa)

F_{HRSG}^{Steam} Steam flow rate exiting from HRSG (kg/h)

T_{HRSG}^{Steam} Temperature of the vapor exiting from the HRSG (ºC)

ΔT_{HRSG} Minimum approach temperature in the HRSG (ºC).

Boilers

For the boilers we follow a similar approach to that followed in the case of the gas turbine. A flow sheet is built in Aspen Hysys to rigorously simulate the boiler. In this case we take advantage of the Aspen Hysys capability for transmitting information from one unit to another as soon as all the degrees of freedom of a unit have been provided by the user See Figure 5.

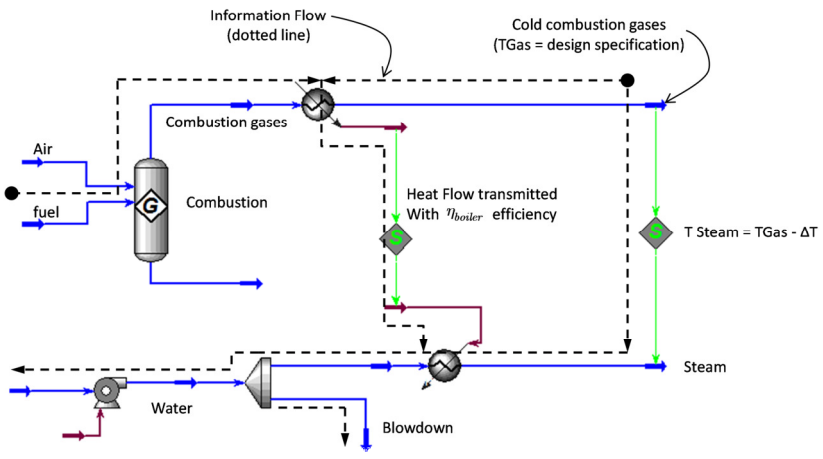

Figure 5. Scheme of the implementation of a Boiler in Aspen Hysys. Dotted lines indicate information flow, which is different to the mass flow.

The design variables, see Figure 5, are the fuel flow rate and the temperature of the gases after heat exchange with the water to generate steam. The calculation sequence is as follows:

A fuel and an air stream (10% excess) are introduced in a Gibbs reactor that simulates the combustion of fuel in the boiler. The combustion gases exchange heat with water at high pressure to generate steam. Instead of using a single heat exchanger, we use two heat exchangers in which the heat is transmitted with a given efficiency (90%). Note that if the output gas temperature is known (design variable), the amount of heat available for generating steam is then also known. The temperature of the steam is fixed by assuming a minimum temperature difference between the stack temperature of combustion gases (exit temperature) and the steam. With this information Aspen-Hysys can calculate the steam flow rate, the blowdown, the high pressure water flow rate and the pump. See Figure 5.

The model can be developed by using a given mass flow rate as basis (i.e. 100 kg/h). Under those conditions if the steam pressure is known the complete model depends on just one design variable. In our case this is the temperature of combustion gases after heat exchange. Efficiency and approach temperature are fixed parameters. See Table 1 for all the relevant data for boilers.

As in the case of gas turbines the model is slightly noisy, and it can be used as is, because it is possible to get accurate derivatives by a finite difference approximation. However, it is much more efficient, from a numerical point of view, to use a polynomial or cubic spline interpolation for which the error is negligible and we have analytical derivatives. This is the approach we use in this work.

Conceptually, the model for the boiler can be written as follows:

$$\begin{aligned} \left[F_{Boiler}^{Steam}, T_{Boiler}^{Steam} \right] &= Boiler\left(T_{Boiler}^{Gas}, Fuel_{Boiler}, P_{Boiler} \right) \\ T_{Steam\,dew}(P_{Boiler}) + \Delta T &\le T_{Boiler}^{Gas} \le 1900 \quad (^{o}C) \end{aligned} \tag{11}$$

where:

P_{Boiler} Steam pressure in the boiler

$Fuel_{Boiler}$	Mass flow rate of fuel consumed by the boiler
T_{Boiler}^{Gas}	Temperature of the gases exiting from the boiler
T_{Boiler}^{Steam}	Boiler steam temperature
F_{Boiler}^{Steam}	Boiler steam mass flow rate

Steam Turbines

A steam turbine converts steam energy into power. The total power of the expansion is further split into power used by the shaft and energy losses (mechanical friction, heat losses and kinetic losses). The three main factors affecting turbine performance are the turbine size (maximum power), the pressure drop across the turbine, and the operating load.

An ideal steam turbine assumes an isentropic expansion. Any model of an actual turbine captures its real behavior by means of the isentropic efficiency:

$$\eta_{IS} = \frac{H_{in} - H_{out}}{H_{in} - H_{IS}} \tag{12}$$

In the literature there are different correlations that capture the isentropic efficiency in terms of the actual load and pressure difference [12, 72, 74]. These correlations vary depending on the manufacturer and the turbine size and model. However, according to Varbanov et al [74] although the machine efficiency varies noticeably in a non-linear way with the load, it retains relatively high values, that go from 88% for smaller turbines to 96% for the larger ones. Bruno et al. [72] presented correlations for turbines working at their maximum load that relates the steam turbine efficiency to the inlet pressure, the shaft power and exhaust conditions (condensing and non-condensing turbines). In this work we include those correlations plus a fixed 0.95 efficiency to take into account a possible deviation from full load operation.

For calculating the ideal turbine we can use a turbine model as implemented in Aspen Hysys or using Aspen Hysys as a thermodynamic calculator server. We follow the second approach

because it affords us better control over the model and we do need to interact with the Aspen Hysys graphical interface, which slows down the calculations.

A complex turbine can be decomposed into its basic components including simple turbine units, splitters and mixers. Any minor differences between the extraction turbine and the decomposed model can be compensated for later in the design by small adjustments in the specification in terms of efficiency [12].

For condensation turbines, we use cooling water as cold utility. Therefore, the pressure of the vacuum header is fixed to allow enough of a temperature difference between the condensing steam and the cooling water.

Conceptually, the model for a turbine can be written as follows:

$$\left[W_{Turbine},\ F_{Turbine}^{SteamOut_k}, T_{Turbine}^{SteamOut_k}, Q_{cond}\right] = Turbine\left(F_{Turbine}^{SteamIn}, P_{Turbine}^{SteamIn}, T_{Turbine}^{SteamIn}, P_{Turbine}^{out_k}, Type\right)$$

$$\begin{aligned} & F_{Turbine}^{SteamOut_k} \geq \alpha_k\, F_{Turbine}^{SteamIn} \\ & \sum_{k \in Turbine\,outputs} \alpha_k = 1 \\ & \underline{F_{Turbine}^{SteamIn}} \leq F_{Turbine}^{SteamIn} \leq \overline{F_{Turbine}^{SteamIn}} \\ & \underline{\alpha_k} \leq \alpha_k \leq \overline{\alpha_k} \end{aligned} \tag{13}$$

where

$F_{Turbine}^{SteamIn}$	Mass steam flow rate entering the turbine
$P_{Turbine}^{SteamIn}$	Pressure of the inlet steam
$T_{Turbine}^{SteamIn}$	Temperature of the inlet steam
$P_{Turbine}^{out_k}$	Pressure of each of the output streams
Type	Type of steam turbine: back pressure or condensing
$W_{Turbine}$	Turbine power
$F_{Turbine}^{SteamOut_k}$	Mass steam flow rate in each turbine output

$T_{Turbine}^{SteamOut_k}$	Temperature of each output stream
Q_{cond}	Condenser heat load (only for condensing turbines)

Vapor Headers

A vapor header combines the steam (water) flows coming from different sources: Boilers, HRSG, steam turbines, letdown valves, etc at different temperatures but at the same pressure and distributes steam to where it is needed (process, steam turbines, letdown valves, deaerator,…).

The header conditions (temperature if the pressure is fixed) can be calculated just from mass and energy balances on all the streams entering the header (it is equivalent to a mixer). The enthalpy of each stream is either determined from the unit that is generated or calculated by Aspen Hysys.

Finally, a simple mass balance relates the input streams to the output streams (including process demands) in each header.

Valves

A letdown valve is simulated as an iso-enthalpic expansion between two pressure levels. The inlet conditions are calculated by the corresponding header. Therefore, the only computation needed is the calculation of the outlet temperature for a given pressure and enthalpy (PH-flash).

De-aerator

To simulate the de-aerator we follow the approach described by Smith [12]. The mass flow and conditions (temperature or enthalpy) of the following water (steam) streams are known: water returned from the process, water coming from the atmospheric or vacuum headers and water leaving from the de-aerator to the boilers or HRSG.

Assuming that the de-aerator works with LP steam extracted from the LP main, and that a fixed percentage (5%) of the steam is vented, we can calculate the treated water makeup and the LP steam requirements (See Figure 6) by mass and energy balances

$$
\begin{aligned}
&F_{TW} + \sum_{i \in Water\ Return} F_i + F_{Deaerator}^{LP_{Steam}} = F_{Boilers}^{Water} + F_{Vent} \\
&H_{TW} F_{TW} + \sum_{k \in Water\ Return} H_k F_k + H_{LP_{Steam}} F_{Deaerator}^{LP_{Steam}} = H_{Boiler}^{Water} F_{Boilers}^{Water} + H_{Vent} F_{Vent} \\
&F_{Vent} = \alpha\, F_{Deaerator}^{LP_{Steam}}
\end{aligned}
\tag{14}
$$

where *F* makes reference to mass flow rates. *H* makes reference to specific enthalpies.

TM	Reference to Treated Water Makeup
Water Return	Set of all the water flows returning to the Power plant: condensate, from process, etc.
α	Fraction of LP steam vented.

Note that enthalpies of the water going to boilers (and HRSG) and Vent streams are calculated with reference to the de-aerator operating conditions.

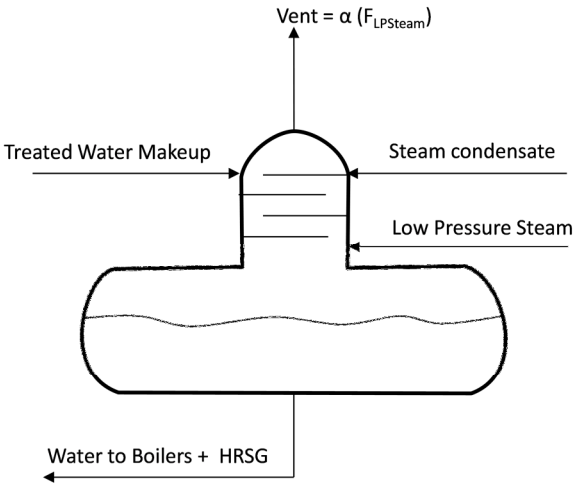

Figure 6. Scheme of a de-aerator.

To perform the economic evaluation we use the total annualized cost (TAC). To annualize the capital costs we use the following relationship [12]:

$$TAC = Operating\,Costs + f\,(Capital\,Cost)$$
$$f = \frac{i(1+i)^N}{(1+i)^N - 1} \tag{15}$$

where i is the interest rate per year and n is the number of years. Fuel, cooling water and other cost data are given in Table 2. To estimate the capital costs we use the correlations in Turton el al [13], except for the gas turbine whose purchased cost was correlated from manufacturing data, supplied by Nye Thermodynamics Corporation [77]:

$$Gas\,Turbine\,Cost\left(M\$\right) = 0.7886\left(W_{GT}\right)^{0.7395} \quad (W_{GT}\ in\ MW) \tag{16}$$

The final model consists of an objective function (TAC minimization) subject to fixed equipment equations: vapor headers, letdown valves and de-aerator. Note that the valves do not require a disjunction because the flow passing through a valve takes a zero value if the valve does not exist. Conditional equipment equations included as two term disjunctions are: Gas Turbine (Y_{GT}), HRSG (Y_{HRSG}) , Boilers (Y_{Boiler_k}) and Steam Turbines (Y_{ST}). Where Y makes reference to the boolean variable of each unit.

We must include the following logical relationships:

- If the HRSG exists then the gas turbine must exist:

$$Y_{HRSG} \Rightarrow Y_{GT} \tag{17}$$

- If a gas turbine is not selected then the HRSG cannot be selected

$$\neg Y_{GT} \Rightarrow \neg Y_{HRSG} \tag{18}$$

- At least the HRSG or a boiler must be selected

$$Y_{HRSG} \vee \left(\underset{k \in Boilers}{\vee} Y_{Boiler_k} \right) \tag{19}$$

The mechanical power must be supplied by a given equipment unit (Steam turbines or electric motors). In that case, we need to allocate the power among the available equipment. To that end we define a new boolean variable:

$Z_{eq,pd}$ True if equipment '*eq*' satisfies the demand '*pd*'

The following disjunctions explicitly enforce that relationship

$$\begin{bmatrix} Z_{eq,pd} \\ W_{eq} = PowerDemand_{pd} \end{bmatrix} \quad (20)$$

The following logical relationships correctly ensure feasible assignments:

- If the mechanical power demand '*pd*' is assigned to equipment unit 'eq' then unit 'eq' must be selected:

$$Z_{eq,pd} \Rightarrow Y_{eq} \quad (21)$$

- Each mechanical power demand must be assigned to some equipment unit

$$\underset{pd}{\vee}\left(Z_{eq,pd} \right) \ \forall eq \quad (22)$$

Although it is not strictly necessary, if we want to guarantee that a given unit of equipment be assigned to a single mechanical power demand we can include the following relationship:

$$at \ most_{eq}\left(Z_{eq,pd} \right) \ \forall pd \quad (23)$$

Note that steam demands are included in the mass balances on each steam header, and therefore it is not explicitly necessary to allocate a Boolean variable to every steam demand.

As a final remark, it is worth mentioning that adding new equipment (i.e. more gas turbines, or boilers) does not modify the equations of existing equipment units; thus, adding another is straightforward. Due to the modular approach, modifying equipment, or adding or removing alternatives does not modify the structure of the rest of the model.

5. Examples

Here we present three instances of the model that illustrates the versatility of both the case study and the modeling framework. All the relevant data for each of the three instances are given in Table 3

The first case is a power plant with high power (electricity) demand (50 MW) and different demands of VHP (100 bar) stream 1 t/h; HP (40 bar) steam 20 t/h; MP steam (20 bar) 5 t/h and LP steam (3 bar) 10 t/h. In this first case there is no mechanical power demand. The optimal solution includes a gas turbine with HRSG, and four steam turbines working between different pressure levels. See Figure 7.

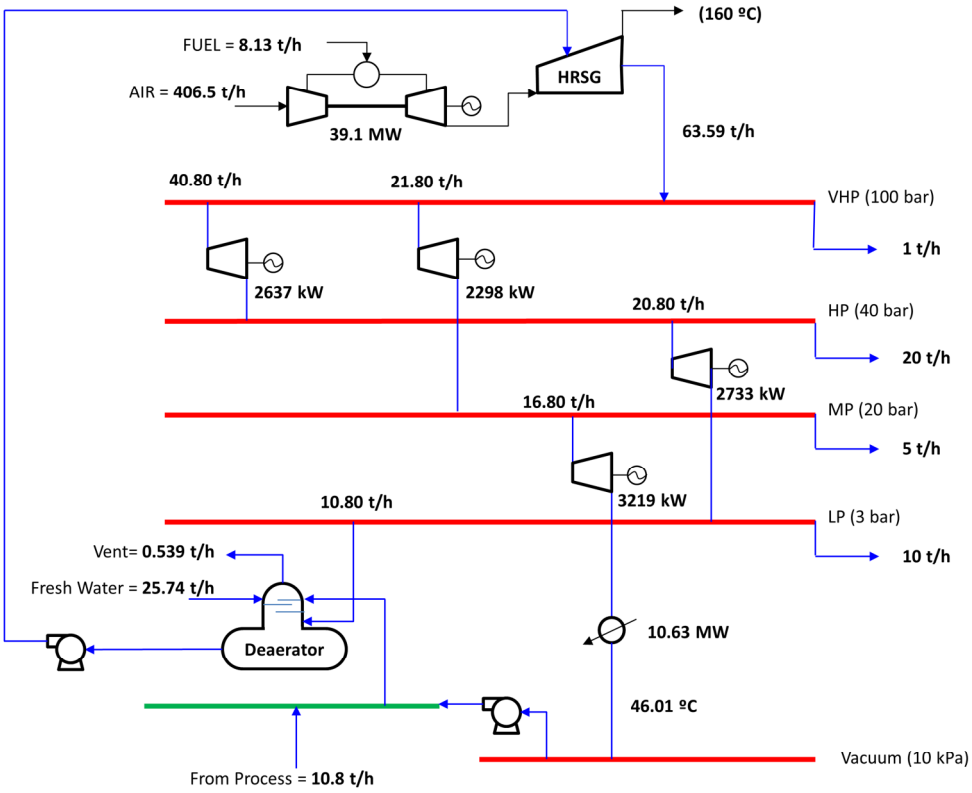

Figure 7. Optimal solution for case study, instance 1.

Notice the perfect steam balance in this example. The operating conditions of the gas turbine and HRSG are optimized in such a way that the steam produced in the HRSG is used, without

any excess, for supplying the steam demands (including the LP steam in the de-aerator) and the extra power demand not satisfied by the gas turbine. Table 4 shows with detail the results of this example.

From a computational point of view, the most complex non-linear problem, that in which all the units exist (used for initialization since it is a feasible solution, although of course not the optimal one) is solved in around 60 seconds of CPU time, although this time depends on the initial values. The remaining the NLP sub-problems are solved typically in CPU times that range between 5 and 20 seconds. (*Intel(R) Core(TM)2Quad CPU 2.4GHz 2.39 GHz. RAM 8 GB under Windows 7*). The MILPs where solved in all the cases in less than 1 second of CPU time. Table 5

All the numerical tests we have performed show that for a given problem with a fixed configuration, the optimal solution that is obtained is not too sensitive to the initial values. In other words, there is reasonable evidence that the solution is the global optimum (although of course we cannot guarantee it). However, in the master problem slack variables become active during the initial iterations, and as a consequence we cannot guarantee that the solution is indeed the global optimum. Nevertheless, numerical tests allowing a large number of major iterations (i.e. 25 major iterations) show that the solution obtained, if not optimal, is at least a good solution (error smaller than 10-15% in the worst cases).

In the second instance there are mechanical power demands (three rotating units of equipment requiring 2000, 2500 and 3500 kW, respectively) that must be satisfied by steam turbines, and a power (electricity) demand of 5 MW. Moreover, the process plant requires 10 t/h of HP steam and 5 t/h of LP steam.

The optimal solution includes a gas turbine, a HRSG, and three steam turbines to provide mechanical power (one of these is a condensing turbine).See Figure 8 and Table6. In contrast to the first example, the steam needed to drive the turbines and satisfy process demands, generates more electricity than is required, and can be fed in and sold to the power grid.

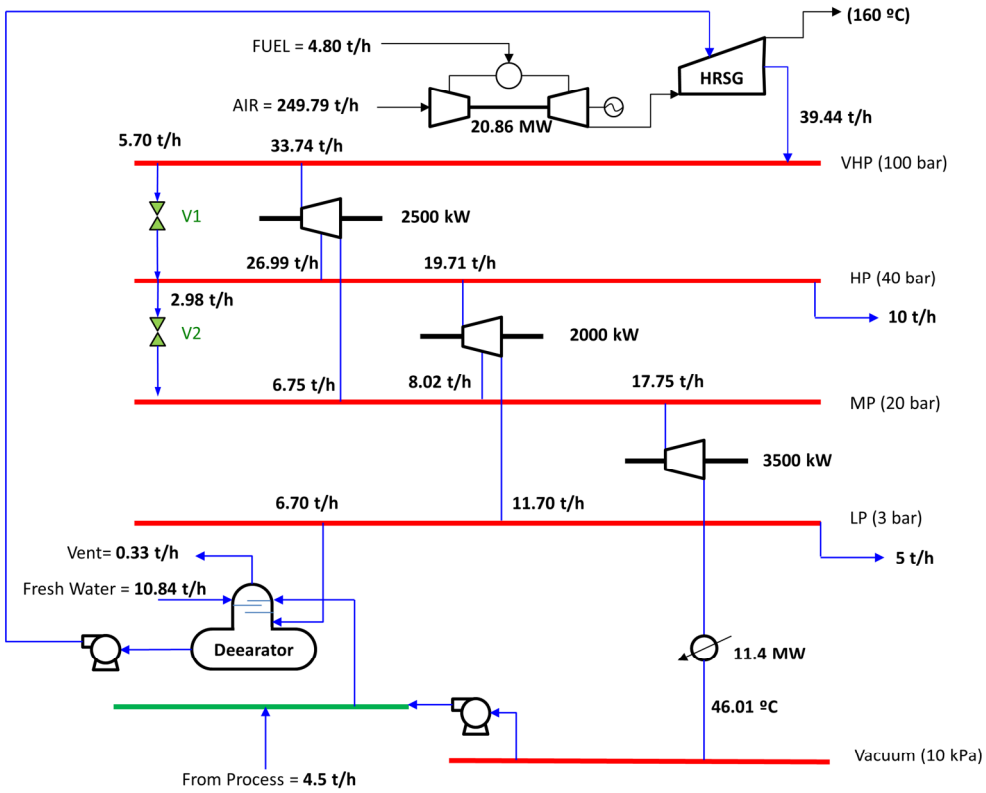

Figure 8. Optimal solution for case study, instance 2.

The computational performance in this case is similar to that observed in the first instance. Nevertheless, it is interesting to note that because of the initialization it is not practical to solve a set of feasible NLPs that includes all the alternatives at least once. Due to the assignment equations linking turbines to mechanical demands, the number of alternatives is too large. However, what we really need to initialize are the steam turbines. Therefore, to initialize the MILP master problem we simply assume that all the steam turbines exist subject to the condition that the power generated by all of them exceeds the total power (mechanical + electricity).

The third instance is similar to the second but the mechanical power demands (500 kW, 1000 kW) and electricity demand (600 kW) are lower. There are also demands for HP steam (2 t/h) and LP steam (5 t/h). The optimal solution includes a Boiler, 2 backpressure steam turbines, a condensing turbine, and the de-aerator. There are also two letdown valves to distribute the steam among the steam mains and the steam turbines. In this case, the reduced demand for

electricity does not justify the high gas turbine cost. See Figure 9 and Table 7 for a detailed summary of the results.

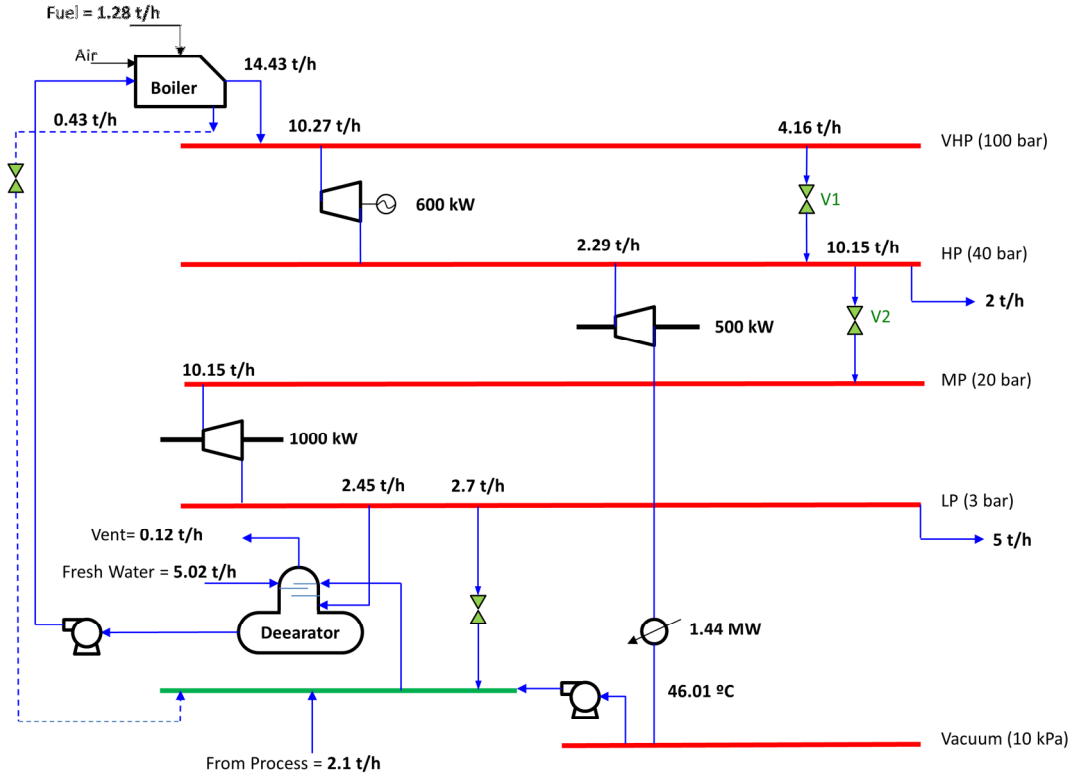

Figure 9. Optimal solution for case study, instance 3.

6. Conclusions

This paper describes how models from very different sources and with different numerical behavior can be incorporated as modules in a general synthesis framework that is able to capture their numerical performance and implement numerical approaches tailored to the need of each model. Some of the capabilities of the proposed approach are the following:

- Structural analysis of the input-output structure of each model for selecting design variables and limiting as much as possible the extent of the recycle structure of the problem.
- Determination of the sparsity pattern of each model to accelerate the calculation of the Jacobian matrix.

- Different approaches for calculating the derivative information that includes analytical, complex variables based differentiation, and finite differences.
- Possibility for dealing with noisy systems through adaptive surrogate models (kriging).
- Disjunctive modeling that facilitates the conceptual modeling.
- Soft and hard logic relations can be easily introduced in the model

The final result is a hybrid system that includes implicit models coming from different sources exhibiting different numerical behavior, and models or constraints in equation form. The basic idea is that the 'best available' model can be used in a rigorous synthesis framework when shortcut models are inadequate. Even though some of the previous characteristics are included in different algebraic modeling systems, as far as we know the combination of all of them in a Generalized Disjunctive Programming environment is new.

It has been shown that the synthesis problem can be written as a General Disjunctive Programming problem and solved without MINLP reformulation using the logic based outer approximation or LP/NLP based branch and bound. Conceptually, the GDP approach allows easy model formulation by the final user and at the same time encapsulates each sub-model, facilitating numerical study of its characteristics.

The synthesis of a utility system, in which each of the components has been treated as a sub-model of different characteristics, has been used as a case study. The objectives were twofold, first we have proved the efficiency and robustness of the proposed framework and second we develop a new tool to the preliminary design of utility systems. Future work will concentrate on producing a new design and synthesis tool that can be applied to many processes.

Acknowledgements.

The authors wish to acknowledge support from the Spanish Ministry of Science and Innovation (CTQ2012-37039-C02-02).

Appendix A.

Logic Based Algorithms with implicit sub-models

The specific disjunctive model we solve in this paper can be written as follows.

$$
\begin{aligned}
&\min : f(\mathbf{x_D}, \mathbf{x_I}) \\
&\quad s.t. \quad \mathbf{x_D} = \mathbf{r_I}(\mathbf{x_I}) \\
&\qquad\quad \mathbf{r_E}(\mathbf{x_D}, \mathbf{x_I}) = \mathbf{0} \\
&\qquad\quad \mathbf{s_E}(\mathbf{x_D}, \mathbf{x_I}) \leq \mathbf{0} \\
&\begin{bmatrix} Y_i \\ \mathbf{x_D} = \mathbf{h_{Ii}}(\mathbf{x_{Ii}}) \\ \mathbf{h_{Ei}}(\mathbf{x_D}, \mathbf{x_{Ii}}) = \mathbf{0} \\ \mathbf{g_{Ei}}(\mathbf{x_D}, \mathbf{x_{Ii}}) \leq \mathbf{0} \end{bmatrix} \veebar \begin{bmatrix} \neg Y_i \\ \mathbf{x_{Ii}} = \mathbf{0} \end{bmatrix} \quad \forall\, i \in D \qquad \text{(A.1)} \\
&\qquad\quad \Omega\left(\mathbf{Y}\right) = True \\
&\qquad\quad \mathbf{x_I} \in X \subseteq \Re^n \\
&\qquad\quad \mathbf{Y} \in \left\{True,\ False\right\}^p
\end{aligned}
$$

where $\mathbf{x_D}$ is a vector of dependent variables (e.g calculated by the implicit models) over which the designer has no direct control. $\mathbf{x_I}$ is a vector of independent variables over which the user has complete control. The index "$\mathbf{I}$" makes reference to implicit equations calculated by external modules (Process Simulator, property estimation modules, etc) and the index "$\mathbf{E}$" makes reference to explicit equations. Note that we have introduced dependent variables in explicit equations (for example in $\mathbf{r_E}(\mathbf{x_D}, \mathbf{x_I}) = \mathbf{0}$; or $\mathbf{s_E}(\mathbf{x_D}, \mathbf{x_I}) \leq \mathbf{0}$), associated with a given topology implicitly this formulation involves sequential function evaluation, first the implicit models and then the explicit equations. An alternative approach consists of adding a new subset of independent variables and explicitly including the relationship with the dependent variables : $\mathbf{x_I^j} = \mathbf{x_D^j}$. In this work we follow the first approach.

In the model given by equation (A.1) we only allow two term disjunctions which is the case of process networks and synthesis problems. This is not a major limitation because an N term disjunction can be reformulated as N disjunctions with 2 terms.

Initial and primal subproblems

To solve the problem given by equation (A.1) we use a version of the logic based outer approximation algorithm [43] or a logic version of the LP-NLP based branch and bound algorithm [50]. The first step consists of initializing all the units (sub-models) inside the

disjunctions. Here there are different alternatives. The first one consists of selecting a basic feasible flow sheet, optimizing it and then performing a sub-lagrangian optimization of the non-existing units in that initial flow sheet. This constitutes the Modeling and Decomposition (MD) strategy [19]. The second approach consists of selecting a minimum set of feasible flow sheets in such a way that all the disjunctions are true at least once. This sub-set can be selected by solving a set covering problem [43] with the constraints given in form of logic relationships in equation (A.1). In some situations the sub-lagrangian optimization cannot be easily performed. This is the case, for example, of distillation columns in a process simulator, in which the optimization of non-existing trays is equivalent to optimizing the complete column. In that case, Caballero et al [23] proposed performing a simulation of the non-existing configurations using the optimal values of the initial base case, and adding an extra term in the master problem that takes into account the eventual existence of those units. Brunet et al [30, 31, 78] extended this approach to other systems but not distillation columns.

During iteration k only a feasible NLP problem is solved, one that corresponds to fixed values of boolean variables given by the master problem. The major difference between the logic versions of the Outer Approximation and LP-NLP based BB is that in the first the master is solved to optimality, however, in the second the NLP is solved when an integer solution is found. In the latter case, the tree generated by the master problem is updated with new linearizations from the last NLP.

It is worth remarking that when a given configuration (a set of Boolean variables that produce a feasible solution) is selected the NLP involves only the common variables and the variables inside the existing disjunctions. The remaining variables are ignored. Mathematically the non-existing variables are set to zero, but that is done a posteriori, and therefore the solver is only viewing a reduced set of variables, which increases the robustness of the optimization.

Master subproblem.

The objective of the master problem is to provide a new set of Boolean variables that likely produce better results than a previous solution. Here we present a tailored master linear disjunctive problem.

We define the following index set for iteration *k* in the algorithm:

$$\Gamma\left\{m \,|\, m \text{ is a feasible configuration already visited by the algorithm}\right\}$$

$$
\begin{aligned}
&\min : \alpha + \Pi\left(\mathbf{u}_{\mathbf{E}}^{1} + \mathbf{u}_{\mathbf{E}}^{2} + \sum_{i\in D}(\mathbf{u}_{\mathbf{E}i}^{1} + \mathbf{u}_{\mathbf{E}i}^{2})\right)\\
&s.t. \quad \alpha \geq f(\mathbf{x}_{\mathbf{I}}^{k},\mathbf{x}_{\mathbf{D}}^{k}) + \nabla_{\mathbf{x}_{\mathbf{I}}} f(\mathbf{x}_{\mathbf{I}}^{k},\mathbf{x}_{\mathbf{D}}^{k})^{T}(\mathbf{x}_{\mathbf{I}} - \mathbf{x}_{\mathbf{I}}^{k}) + \sum_{m\in\Gamma}\Delta obj_{m}^{k}\\
&\quad sign(\mathbf{r}_{\mathbf{E}}^{k})\left[\mathbf{r}_{\mathbf{E}}(\mathbf{x}_{\mathbf{I}}^{k},\mathbf{x}_{\mathbf{D}}^{k}) + \nabla_{\mathbf{x}_{\mathbf{I}}}\mathbf{r}_{\mathbf{E}}(\mathbf{x}_{\mathbf{I}}^{k},\mathbf{x}_{\mathbf{D}}^{k})^{T}(\mathbf{x}_{\mathbf{I}} - \mathbf{x}_{\mathbf{I}}^{k}) + \sum_{m\in\Gamma}\Delta\mathbf{r}_{\mathbf{E}m}^{k}\right] \leq \mathbf{u}_{\mathbf{E}}^{1}\\
&\quad \mathbf{s}_{\mathbf{E}}(\mathbf{x}_{\mathbf{I}}^{k},\mathbf{x}_{\mathbf{D}}^{k}) + \nabla_{\mathbf{x}_{\mathbf{I}}}\mathbf{s}_{\mathbf{E}}(\mathbf{x}_{\mathbf{I}}^{k},\mathbf{x}_{\mathbf{D}}^{k})^{T}(\mathbf{x}_{\mathbf{I}} - \mathbf{x}_{\mathbf{I}}^{k}) + \sum_{m\in\Gamma}\Delta\mathbf{s}_{\mathbf{E}m}^{k} \leq \mathbf{u}_{\mathbf{E}}^{2}\\
&\left[\begin{array}{c} Y_i \\ sign(\mathbf{h}_{\mathbf{E}}^{k})\left[\mathbf{h}_{\mathbf{E}i}(\mathbf{x}_{\mathbf{I}i}^{k},\mathbf{x}_{\mathbf{D}}^{k}) + \nabla_{\mathbf{x}_{\mathbf{I}}}\mathbf{h}_{\mathbf{E}}(\mathbf{x}_{\mathbf{I}i}^{k},\mathbf{x}_{\mathbf{D}}^{k})^{T}(\mathbf{x}_{\mathbf{I}i} - \mathbf{x}_{\mathbf{I}i}^{k}) + \sum_{m\in\Gamma}\Delta\mathbf{h}_{\mathbf{E}i,m}^{k}\right] \leq \mathbf{u}_{\mathbf{E}i}^{k} \\ \mathbf{g}_{\mathbf{E}i}(\mathbf{x}_{\mathbf{I}i}^{k},\mathbf{x}_{\mathbf{D}}^{k}) + \nabla_{\mathbf{x}_{\mathbf{I}}}\mathbf{g}_{\mathbf{E}i}(\mathbf{x}_{\mathbf{I}i}^{k},\mathbf{x}_{\mathbf{D}}^{k})^{T}(\mathbf{x}_{\mathbf{I}i} - \mathbf{x}_{\mathbf{I}i}^{k}) + \sum_{m\in\Gamma}\Delta\mathbf{g}_{\mathbf{E}i,m}^{k} \leq \mathbf{0} \end{array}\right] \underline{\vee}\left[\begin{array}{c} \neg Y_i \\ \mathbf{x}_{\mathbf{I}i} = \mathbf{0} \end{array}\right] \ \forall\, i \in D\\
&\quad \Omega(\mathbf{Y}) = True\\
&\quad \mathbf{u}_{\mathbf{E}}^{1}, \mathbf{u}_{\mathbf{E}}^{2}, \mathbf{u}_{\mathbf{E}i}^{k} \geq \mathbf{0}\\
&\quad \mathbf{x}_{\mathbf{I}} \in X \subseteq \Re^{n}\\
&\quad \mathbf{Y} \in (True,\, False)^{p}
\end{aligned}
\Bigg\} k = 1...K
$$

(A.2)

The disjunction model (A.2) is obtained by linearization in terms of independent variables in the optimal solution provided by the NLP solver, either in the initialization or in a given iteration k. The term Δobj_{m}^{k} corresponds to the difference between the objective function at a given iteration k of the NLP and the objective function associated with a given topology. The terms $\Delta\mathbf{r}_{\mathbf{E}m}^{k}; \Delta\mathbf{s}_{\mathbf{E}m}^{k};\ \ \Delta\mathbf{h}_{\mathbf{E}i,m}^{k}; \Delta\mathbf{g}_{\mathbf{E}i,m}^{k}$ are the differences between the values of a given constraint for the new topology and their value in the original NLP^{k} problem. These terms are not needed (fixed to zero) if the modeling and decomposition or the logic based outer approximation are used. Only in the third alternative, when no sub-optimization of non-existing disjunctions is used, must these terms be included.

Variables '**u**' are positive slack variables, that appear as penalties in the objective function using an exact penalty (Π is the penalty factor). These variables are introduced to ensure that the an

infeasible master problem could only be the result of logical relationships, and at the same time is a heuristic to try to minimize the effect of non-convexities [25].

The disjunctive master problem in (A.2) can be solved as a MILP problem using the convex hull reformulation [45].

If the original problem is convex, the master problem in the MD and logic OA base versions yields a lower bound to the optimal solution. Therefore, the optimal solution is found when at a given iteration, the optimal solution of the primal and master problems are within a given tolerance of each other (or when they cross each other if a canonical cut is added in each master problem [43]). In non-convex problems, like those solved in this work, we cannot guarantee that the master problem will yield a lower bound. Therefore, the search is terminated when after two consecutive iterations the NLP does not obtain an improvement [25]. This is just a heuristic based on experience. To improve the quality of the solution in non-convex problems, one could perform a fixed number of iterations (e.g. 10) but at the price of increasing the computational time to solve the problem.

References

1. Shacham, M., et al., *Equation oriented approach to process flowsheeting.* Computers and Chemical Engineering, 1982. **6**(2): p. 79-95.
2. Grossmann, I.E., J.A. Caballero, and H. Yeomans, *Mathematical programming approaches to the synthesis of chemical process systems.* Korean Journal of Chemical Engineering, 1999. **16**(4): p. 407-426.
3. Rosenthal, R.E., *GAMS - A User's Guide*. 2012: GAMS Development Corporation, Washington, DC, USA.
4. Fourer, R., D.M. Gay, and B.W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*. 2003, United States: Duxbury Press, Brooks, Cole Publishing Company.
5. *ASCEND*. [cited 2013; Available from: http://ascend4.org/.
6. *Process Systems Enterprise, gPROMS, www.psenterprise.com/gproms,*. 1997-2009.
7. Marquardt, W., *Trends in computer-aided process modeling.* Computers and Chemical Engineering, 1996. **20**(6-7): p. 591-609.
8. Biegler, L.T., I.E. Grossmann, and A.W. Westerberg, *Systematic methods of chemical process design*. Prentice Hall international series in the physical and chemical engineering sciences. 1997, Upper Saddle River, N.J.: Prentice Hall PTR. xviii, 796 p.

9. Grossmann, I.E., *Review of Nonlinear Mixed-Integer and Disjunctive Programming Techniques.* Optimization and Engineering, 2002(3): p. 227-252.

10. Lee, S. and I.E. Grossmann, *Logic-based modeling and solution of nonlinear discrete/continuous optimization problems.* Annals Of Operations Research, 2005. **139**(1): p. 267-288.

11. Dimian, A.C., *Integrated design and simulation of chemical processes*. Computer Aided Chemical Engineering 13. 2003, Amsterdam, The Netherlands: Elsevier.

12. Smith, R., *Chemical Process Design and Integration*. 2005, Chichester: John wiley & Sons, Ltd. 687.

13. Turton, R., et al., *Analysis Synthesis and Design of Chemical Processes*. 2003, New York: McGraw-Hill.

14. Luyben, W.L., *Principles and Case Studies of Simultaneous Design.* 2011, Hoboken, New Yersey, USA.: John Wiley & Sons.

15. Caballero, J.A. and I.E. Grossmann, *Aggregated models for integrated distillation systems.* Industrial and Engineering Chemistry Research, 1999. **38**(6): p. 2330-2344.

16. Braunschweig, B.L., et al., *Process modeling: The promise of open software architectures.* Chemical Engineering Progress, 2000. **96**(9): p. 65-76.

17. Harsh, M.G., P. Saderne, and L.T. Biegler, *A mixed integer flowsheet optimization strategy for process retrofits—the debottlenecking problem.* Computers & Chemical Engineering, 1989. **13**(8): p. 947-957.

18. Diwekar, U.M., I.E. Grossmann, and E.S. Rubin, *An MINLP Process Synthesizer For A Sequential Modular Simulator.* Industrial & Engineering Chemistry Research, 1992. **31**(1): p. 313-322.

19. Kocis, G.R. and I.E. Grossmann, *A Modeling And Decomposition Strategy For The Minlp Optimization Of Process Flowsheets.* Computers & Chemical Engineering, 1989. **13**(7): p. 797-819.

20. Reneaume, J.M.F., B.M. Koehert, and X.L. Joulia, *Optimal process synthesis in a modular simulator environment: new formulation on the mixed integer nonlinear programming problem.* Industrial & Engineering Chemistry Research, 1995. **34**: p. 4378-4394.

21. Díaz, M.S. and J.A. Bandoni, *A mixed integer optimization strategy for a large scale chemical plant in operation.* Computers & Chemical Engineering, 1996. **20**(5): p. 531-545.

22. Ilacqua, A., et al., *Simulador a medida de una plnata de etileno para estudios de sensibilidad paramétrica y optimización*, in *XVI Congreso Interamericano de Ingeniería Química.* . 1991: Buenos Aires, Argentina.

23. Caballero, J.A., D. Milan-Yanez, and I.E. Grossmann, *Rigorous Design of Distillation Columns: Integration of Disjunctive Programming and Process Simulators.* Industrial & Engineering Chemistry Research, 2005. **44**(17): p. 6760-6775.

24. Duran, M.A. and I.E. Grossmann, *An Outer-Approximation Algorithm For A Class Of Mixed-Integer Nonlinear Programs.* Mathematical Programming, 1986. **36**(3): p. 307-339.

25. Viswanathan, J. and I.E. Grossmann, *A combined penalty function and outer-approximation method for MINLP optimization.* Computers & Chemical Engineering, 1990. **14**(7): p. 769-782.

26. Kocis, G.R. and I.E. Grossmann, *Relaxation Strategy For The Structural Optimization Of Process Flow Sheets.* Industrial & Engineering Chemistry Research, 1987. **26**(9): p. 1869-1880.

27. Brunet, R., et al., *Minimization of the LCA impact of thermodynamic cycles using a combined simulation-optimization approach.* Applied Thermal Engineering, 2012. **48**(0): p. 367-377.

28. Caballero, J.A., et al., *Design of Hybrid Distillation−Vapor Membrane Separation Systems.* Industrial & Engineering Chemistry Research, 2009. **48**(20): p. 9151-9162.

29. Caballero, J.A., A. Odjo, and I.E. Grossmann, *Flowsheet optimization with complex cost and size functions using process simulators.* AIChE Journal, 2007. **53**(9): p. 2351-2366.

30. Brunet, R., G. Guillén-Gosálbez, and L. Jiménez, *Cleaner Design of Single-Product Biotechnological Facilities through the Integration of Process Simulation, Multiobjective Optimization, Life Cycle Assessment, and Principal Component Analysis.* Industrial & Engineering Chemistry Research, 2011. **51**(1): p. 410-424.

31. Brunet, R., et al., *Hybrid simulation-optimization based approach for the optimal design of single-product biotechnological processes.* Computers & Chemical Engineering, 2012. **37**(0): p. 125-135.

32. Dantus, M.M. and K.A. High, *Evaluation of waste minimization alternatives under uncertainty: a multiobjective optimization approach.* Computers & Chemical Engineering, 1999. **23**(10): p. 1493-1508.

33. Leboreiro, J. and J. Acevedo, *Processes synthesis and design of distillation sequences using modular simulators: a genetic algorithm framework.* Computers & Chemical Engineering, 2004. **28**(8): p. 1223-1236.

34. Gutiérrez-Antonio, C. and A. Briones-Ramírez, *Pareto front of ideal Petlyuk sequences using a multiobjective genetic algorithm with constraints.* Computers & Chemical Engineering, 2009. **33**(2): p. 454-464.

35. Gutérrez-Antonio, C., A. Briones-Ramírez, and A. Jiménez-Gutiérrez, *Optimization of Petlyuk sequences using a multi objective genetic algorithm with constraints.* Computers & Chemical Engineering, 2011. **35**(2): p. 236-244.

36. Vazquez–Castillo, J.A., et al., *Design and optimization, using genetic algorithms, of intensified distillation systems for a class of quaternary mixtures.* Computers & Chemical Engineering, 2009. **33**(11): p. 1841-1850.

37. Eslick, J.C. and D.C. Miller, *A multi-objective analysis for the retrofit of a pulverized coal power plant with a CO2 capture and compression process.* Computers & Chemical Engineering, 2011. **35**(8): p. 1488-1500.

38. Torres, C.M., et al., *Evaluation Tool for the Environmental Design of Chemical Processes.* Industrial & Engineering Chemistry Research, 2011. **50**(23): p. 13466-13474.

39. Torres, C.M., et al., *An automated environmental and economic evaluation methodology for the optimization of a sour water stripping plant.* Journal of Cleaner Production, 2013. **44**(0): p. 56-68.

40. Lazzaretto, A., et al., *Criteria for the decomposition of energy systems in local/global optimizations.* Energy, 2010. **35**(2): p. 1157-1163.

41. Muñoz, J.R. and M.R. von Spakovsky, *A decomposition approach for the large scale synthesis design optimization of highly coupled, highly dynamic energy systems.* Internationa Journal of Thermodynamics. , 2010. **4**(1): p. 1-17.

42. Rancruel, D.F. and M.R. von Spakovsky, *A decomposition strategy based on thermoeconomic isolation applied to the optimal synthesis/design and operation of an advanced tactical aircraft system.* Energy, 2006. **31**: p. 3327-3341.

43. Turkay, M. and I.E. Grossmann, *Logic-based MINLP algorithms for the optimal synthesis of process networks.* Computers & Chemical Engineering, 1996. **20**(8): p. 959-978.

44. Hooker, J.N. and M.A. Osorio, *Mixed logical-linear programming.* Discrete Applied Mathematics, 1999. **97**: p. 395-442.

45. Raman, R. and I.E. Grossmann, *Modeling And Computational Techniques For Logic-Based Integer Programming.* Computers & Chemical Engineering, 1994. **18**(7): p. 563-578.

46. Nemhauser, G. and L. Wolsey, *Integer and combinatorial optimization*. 1999: John Wiley & Sons.

47. Williams, H.P., *Model building in mathematical programming.* . 1990: Wiley (Chichester England and New York)

48. MATLAB., *The Language of Technical Computing.* 2006., The Mathworks Inc.

49. Holmström, K., *The Tomlab Optimization Environment in Matlab.* Adv. Model Optim., 1999. **1**: p. 47-69.

50. Quesada, I. and I.E. Grossmann, *An LP/NLP based branch and bound algorithm for convex MINLP optimization problems.* Computers & Chemical Engineering, 1992. **16**(10-11): p. 937-947.

51. Westerberg, A.W., et al., *Process flowsheeting* 1979, Cambridge. : Cambridge University Press.

52. Tolsma, J.E., J.A. Clabaugh, and P.I. Barton, *Symbolic Incorporation of External Procedures into Process Modeling Environments.* Industrial & Engineering Chemistry Research, 2002. **41**(16): p. 3867-3876.

53. Squire, W. and G. Trapp, *Using Complex Variables to Estimate Derivatives of Real Functions.* SIAM Rev., 1998. **40**(1): p. 110-112.

54. Krige, D.G., *A Statistical Approach to Some Mine Valuations and Allied Problems at the Witwatersrand.* 1951, University of Witwatersrand.

55. Jones, D.R., *A taxonomy of global optimization methods based on response surfaces.* Journal Of Global Optimization, 2001. **21**(4): p. 345-383.

56. Jones, D.R., M. Schonlau, and W.J. Welch, *Efficient global optimization of expensive black-box functions.* Journal Of Global Optimization, 1998. **13**(4): p. 455-492.

57. Sasena, M.J., *Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations.* 2002, University of Michigan: Michigan. p. 237.

58. Davis, E. and M. Ierapetritou, *A kriging method for the solution of nonlinear programs with black-box functions.* AIChE Journal., 2007. **53**(8): p. 2001-2012.

59. Henao, C.A. and C.T. Maravelias, *Surrogate-based superstructure optimization framework.* AIChE Journal, 2011. **57**(5): p. 1216-1232.

60. Caballero, J.A. and I.E. Grossmann, *An algorithm for the use of surrogate models in modular flowsheet optimization.* AIChE Journal, 2008. **54**(10): p. 2633-2650.

61. Nishio, M., et al., *A Thermodynamic Approach to Steam-Power System Design.* Industrial & Engineering Chemistry Process Design and Development, 1980. **19**(2): p. 306-312.

62. Papoulias, S.A. and I.E. Grossmann, *A structural optimization approach in process synthesis—I: Utility systems.* Computers & Chemical Engineering, 1983. **7**(6): p. 695-706.

63. Dhole, V.R. and B. Linnhoff, *Total site targets for fuel, co-generation, emissions, and cooling.* Computers & Chemical Engineering, 1993. **17, Supplement 1**(0): p. S101-S109.

64. Iyer, R.R. and I.E. Grossmann, *Synthesis and operational planning of utility systems for multiperiod operation.* Computers & Chemical Engineering, 1998. **22**(7–8): p. 979-993.

65. Godoy, E., S.J. Benz, and N.J. Scenna, *A strategy for the economic optimization of combined cycle gas turbine power plants by taking advantage of useful thermodynamic relationships.* Applied Thermal Engineering, 2011. **31**(5): p. 852-871.

66. Siefert, N.S. and S. Litster, *Exergy and economic analyses of advanced IGCC–CCS and IGFC–CCS power plants.* Applied Energy, 2013. **107**(0): p. 315-328.

67. Tică, A., et al., *Design of a combined cycle power plant model for optimization.* Applied Energy, 2012. **98**(0): p. 256-265.

68. Alobaid, F., et al., *Modeling and investigation start-up procedures of a combined cycle power plant.* Applied Energy, 2008. **85**(12): p. 1173-1189.

69. Luo, X., et al., *Operational planning optimization of steam power plants considering equipment failure in petrochemical complex.* Applied Energy, 2013. **112**(0): p. 1247-1264.

70. Mavromatis, S.P. and A.C. Kokossis, *Conceptual optimisation of utility networks for operational variations—I. targets and level optimisation.* Chemical Engineering Science, 1998. **53**(8): p. 1585-1608.

71. Mavromatis, S.P. and A.C. Kokossis, *Conceptual optimisation of utility networks for operational variations—II. Network development and optimisation.* Chemical Engineering Science, 1998. **53**(8): p. 1609-1630.

72. Bruno, J.C., et al., *A Rigorous MINLP Model for the Optimal Synthesis and Operation of Utility Plants.* Chemical Engineering Research and Design, 1998. **76**(3): p. 246-258.

73. Manninen, J. and X.X. Zhu, *Level-by-level flowsheet synthesis methodology for thermal system design.* AIChE Journal, 2001. **47**(1): p. 142-159.

74. Varbanov, P.S., S. Doyle, and R. Smith, *Modelling and Optimization of Utility Systems.* Chemical Engineering Research and Design, 2004. **82**(5): p. 561-578.

75. Yeomans, H. and I.E. Grossmann, *A systematic modeling framework of superstructure optimization in process synthesis.* Computers & Chemical Engineering, 1999. **23**(6): p. 709-731.

76. Brooks, F.J., *GE Gas Turbine Performance Characteristic.* General Electric Reference Library, GER-3576H. Available at www.gepower.com=publications=gers=GER3567H.pdf, 2001.

77. *Nye Thermodynamics Corporation.* http://www.gas-turbines.com/trader/outprice.htm.

78. Brunet, R., et al., *Combined simulation–optimization methodology for the design of environmental conscious absorption systems.* Computers & Chemical Engineering, 2012. **46**(0): p. 205-216.

Table 1. Summary of operating parameters

Unit		Operating Parameters
Headers	VHP Pressure	100 bar
	HP Pressure	40 bar
	MP Pressure	20 bar
	LP Pressure	3 bar
	Condenser Pressure	1.01325 bar (1 atm)
	Vacuum Pressure	0.1 bar (10 kPa)
	Vacuum Temperature	46.01 ºC
Gas Turbine	Compression ratio (*r*)	$10 \leq r \leq 15$
	Combustion temperature	$T^{out}_{combustion} \leq 1200^{o} C$
	Exhaust gas temperature	$T_{Exhaust} \leq 600^{o} C$
	Thermodynamics	Peng Robinson (combustion chamber) ASME Steam for steam lines.
HRSG	Min approach temperature	30 ºC
	Stack temperature	160 ºC
Boilers	Efficiency	90 %
	Approach temperature	30 ºC
	Blowdown rate	3 %
	Excess combustion air	10 %
	Thermodynamics	Peng Robinson Hysys default (combustion chamber) ASME Steam for steam lines.
Steam Turbines	Efficiency	$\eta_{is} = c\left(\frac{a - W_t}{b + W_t}\right)0.95$ (W in kW)
	Backpressure	$P_{in} \geq 40\,bar;\ a = -427.0992;\ b = 865.5034;\ c = -0.8217;$ $P_{in} \geq 20\,bar;\ a = -378.0419;\ b = 758.8181;\ c = -0.8223;$ $P_{in} \geq 1\ bar;\ a = -181.9821;\ b = 381.1312;\ c = -0.8150;$
	Condensation	$P_{in} \geq 40\,bar;\ a = -313.3561;\ b = 648.2201;\ c = -0.7714;$ $P_{in} \geq 20\,bar;\ a = 244.8585;\ b = 528.1410;\ c = 0.7769;$ $P_{in} \geq 1\ bar;\ a = -142.4190;\ b = 323.1106;\ c = -0.7700;$
Deaerator	Pressure	1.01325 bar (1 atm)
	Vent	5% LP introduced

Table 2. Utilities Data.

Utility		
Fuel	Natural Gas	
	Composition (wt fraction)	
	Methane	0.8405
	Ethane	0.1278
	Propane	0.0203
	n-Butane	0.0033
	Ethylene	0.0081
	Temperature	25 ºC
	Pressure	101.325 kPa (1 atm)
	Cost	0.23 \$/(m^3-std) = 2.5792 M\$ /(year·tone fuel)[+]
Cooling Water	Temperature	25 - 35 ºC
	Pressure	101.325 kPa (1 atm)
	Cost	19.1952 10^{-6} M\$/(kW·year)
Demineralized Water	Temperature	25
	Pressure	101.325 kPa (1 atm)
	Cost	0.02 M\$/(t/h year)

[+]Calculated for 8000 h of operation year ($\rho_{Fuel} = 0.7134\, kg\ /\ (m^3{}_{std})$).

Table 3. Data for examples

	Instance 1	**Instance 2**	**Instance 3**
VHP Steam Demand (t/h)	1	---	---
HP Steam Demand (t/h)	20	10	2
MP Steam Demand (t/h)	5	---	---
LP Steam Demand (t/h)	10	5	5
Water Returned from process (t/h)	10.8	4.5	2.1
Electricity demand (MW)	50	5	0.6
Drivers demands (kW)	---	2000 2500 3500	500 1000

Table 4. Results for example 1

Equipment		
Gas Turbine	Power (MW)	39.1
	Fuel (t/h)	8.13
	Compression ratio	1.5
	Air to fuel ratio	50.0
	Exhaust gas temperature (ºC)	592.9
	Installed cost (M$)	41.5
HRSG	Water inlet temperature (ºC)	80
	Steam mass flow rate (t/h)	63.59
	Installed cost (M$)	3.23
Backpressure turbine 1	PIn – Pout (bar)	VHP(100) – HP (40)
	Work (kW)	2637
	Efficiency	0.781
	Steam Flow (t/h)	40.8
	Installed cost (M$)	1.81
Backpressure turbine 2	PIn – Pout (bar)	VHP(100) – MP (20)
	Work (kW)	2298
	Efficiency	0.780
	Steam Flow (t/h)	21.80
	Installed cost (M$)	1.76
Backpressure turbine 3	PIn – Pout (bar)	HP (40) – LP(3)
	Work (kW)	2733
	Efficiency	0.78
	Steam Flow (t/h)	20.80
	Installed cost (M$)	1.83
Condensing Turbine	PIn – Pout (bar)	MP (20) – Vacuum (0.1)
	Work (kW)	3219
	Efficiency	0.738
	Steam Flow (t/h)	16.80
	Installed cost (M$)	1.89
	Condenser heat load (MW)	10.63
	Condenser Installed cost (M$)	0.33
Deaerator	LP Steam flow rate (t/h)	10.80
	Fresh water makeup (t/h)	25.74
	Vent (t/h)	0.539
	Water from atm. header (t/h)	16.80
	Installed cost (M$)	0.067
Costs	**Total installed Cost (M$)**	**52.45**
	Annualizing factor (f) (8 years 10% interest)	0.1874
	Fuel Cost (M$/year)	20.97
	Fresh water cost (M$/year)	0.515
	Cooling water cost (M$/year)	0.108
	Total utilities cost (M$/year)	**21.59**
	TAC (M$/year)	**31.42**

Note: cost of pipes, pumps and valves is not included.

Table 5. Numerical statistics for the three instance.

	Instance 1	**Instance 2**	**Instance 3**
Nº of Boolean variables	15	48	37
Nº of independent variables	25	25	25
Nº of linear explicit equations[(1)]	25	25	25
Nº of non-linear explicit equations[(2)]	8	41	30
Nº of implicit blocks	24	24	24
Detailed iterations for instance 1[(3)]			
Iterations	Objective	CPU time (seconds)	Solver
Initialization (all units exist). NLP	37.99	55.3	CONOPT
MILP Master	*29.14*	*0.047*	*CEPLEX*
Iteration 2: NLP	31.85	29.6	CONOPT
Iteration 2: Master-MILP	*30.21*	*0.28*	*CEPLEX*
Iteration 3: NLP	33.66	7.7	CONOPT
Iteration 3: Master-MILP	*34.62*	*0.14*	*CEPLEX*
Iteration 4: NLP	***31.42***	14.1	CONOPT
Iteration 4: Master-MILP	*35.01*	*0.29*	*CEPLEX*
Iteration 5: NLP	33.24	6.1	CONOPT
Iteration 5: Master-MILP	*35.07*	*0.16*	*CEPLEX*
Iteration 6: NLP	54.73	24.9	CONOPT
Iteration 6: Master-MILP	*96.11*	*0.34*	*CEPLEX*
Iteration 7: NLP	32.08	22.6	CONOPT
Iteration 7: Master-MILP	*189.06*	*0.16*	*CEPLEX*
Iteration 8: NLP	32.58	8.4	CONOPT

(1) Logical relationships not included
(2) Includes both independent and dependent variables.
(3) Numerical performance for instances 2 and 3 is similar.
(4) Fixed number of iterations to 8. Note also that the Master is not providing lower bounds due to slack variables.
(Intel(R) Core(TM)2Quad CPU 2.4GHz 2.39 GHz. RAM 8 GB under Windows 7)

Table 6. Results for example 2

Equipment		
Gas Turbine	Power (MW)	20.86
	Fuel (t/h)	4.80
	Compression ratio	10
	Air to fuel ratio	52.04
	Exhaust gas temperature (ºC)	600
	Installed cost (M$)	26.09
HRSG	Water inlet temperature (ºC)	80
	Steam mass flow rate (t/h)	39.44
	Installed cost (M$)	2.15
Extraction turbine 1	PIn – Pout (bar)	VHP(100) – HP(40) – MP (20)
	Work (kW)	2500
	Steam Flow In(t/h)	33.74
	Stem Flow Out (t/h)	26.99 – 6.75
	Efficiency	0.781 – 0.784
	Installed cost (M$)	2.50
Extraction turbine 2	PIn – Pout (bar)	HP(40) – MP(20) – LP(3)
	Work (kW)	2000
	Steam Flow In (t/h)	19.71
	Stem Flow Out (t/h)	8.02-11.70
	Efficiency	0.781 - 0.782
	Installed cost (M$)	2.70
Condensing Turbine	PIn – Pout (bar)	MP(20) – Vacuum (0.1)
	Work (kW)	3500
	Steam Flow (t/h)	17.75
	Efficiency	0.738
	Installed cost (M$)	1.92
	Condenser heat load (MW)	11.40
	Condenser Installed cost (M$)	0.348
Deaerator	LP Steam flow rate (t/h)	6.70
	Fresh water makeup (t/h)	10.84
	Vent (t/h)	0.33
	Water from atm. header (t/h)	22.25
	Installed cost (M$)	0.052
Costs	**Total installed Cost (M$)**	**35.77**
	Annualizing factor (f) (8 years 10% interest)	0.1874
	Fuel Cost (M$/year)	10.53
	Fresh water cost (M$/year)	0.216
	Cooling water cost (M$/year)	0.116
	Total utilities cost (M$/year)	**10.862**
	TAC (M$/year)	**19.43**

Table 7. Results for example 3

Equipment		
Boiler	Fuel (t/h)	1.28
	Temperature steam (ºC)	502.6
	Heat load (MW)	13.47
	Steam flow rate (t/h)	14.43
	Blowdown flow rate (t/h)	0.43
	Installed Cost	0.942
Back pressure turbine 1	PIn – Pout (bar)	VHP(100) – HP (40)
	Work (kW)	600
	Steam Flow In(t/h)	10.27
	Efficiency	0.782
	Installed cost (M$)	1.08
Back pressure turbine 2	PIn – Pout (bar)	MP(20) – LP(3)
	Work (kW)	1000
	Steam Flow In (t/h)	10.15
	Efficiency	0.782
	Installed cost (M$)	1.36
Condensing Turbine	PIn – Pout (bar)	HP(40) – vacuum (0.1)
	Work (kW)	500
	Steam Flow (t/h)	2.29
	Efficiency	0.734
	Installed cost (M$)	1.01
	Condenser heat load (MW)	1.44
	Condenser Installed cost (M$)	0.124
Deaerator	LP Steam flow rate (t/h)	2.45
	Fresh water makeup (t/h)	5.02
	Vent (t/h)	0.12
	Water from atm. header (t/h)	7.09
	Installed cost (M$)	0.0317
Costs	**Total installed Cost (M$)**	**4.56**
	Annualizing factor (f) (8 years 10% interest)	0.1874
	Fuel Cost (M$/year)	3.30
	Fresh water cost (M$/year)	0.101
	Cooling water cost (M$/year)	0.015
	Total utilities cost (M$/year)	**3.42**
	TAC (M$/year)	**4.27**