Flowsheet optimization with implicit models and complex cost and size functions using Chemical Process Simulators.

José A. Caballero[*], Andrew Odjo and Ignacio E. Grossmann[+].

[*] Department of Chemical Engineering, University of Alicante. Apartado de Correos 99. E-03080 Alicante Spain. E-mail Caballer@ua.es

[+] Department of Chemical Engineering. Carnegie Mellon University. Pittsburgh, PA.

December 2006

Abstract.

In this paper we address the synthesis and design of chemical processes using Chemical Modular Process Simulators – that include state of the art models- including discontinuous cost and sizing equations. Equations are divided into 'implicit' ones which include all the equations in the process simulators with an input-output black box structure, and other third party equations (i.e. sizing and costing correlations for any database) and 'explicit' constraints in form of equalities or inequalities like in any regular equation based optimization environment.

Using this modular framework the problem is formulated as a Generalized Disjunctive Programming problem and reformulated and solved as an Mixed-Integer Nonlinear Programming Problem. Different algorithms (Branch and Bound, Outer Approximation and LP/NLP based Branch and Bound) have been adapted to deal with implicit equations and their capabilities have been studied. Several examples are presented in order to illustrate the performance of the algorithms.

TOPICAL HEADING : Process Systems Engineering

KEYWORDS: MINLP, Process Simulators; Process Synthesis; Process Design; Disjunctive Programming.

* Author to whom all correspondence should be addressed.

Jose A. Caballero. University of Alicante. Chemical Engineering Department.
Apartado de Correos 99. 03080. Alicante Spain. E-mail: caballer@ua.es

Introduction

Optimal Process Synthesis is one of the most challenging problems in Chemical Engineering[1]. The goal of conceptual design (process synthesis) is the identification of the best flowsheet structure (process system) that must carry out a specific task, such as conversion of raw material into a product, or separation of a multi-component mixture. In order to accomplish this goal many alternative designs must be considered. Usually the flowsheet is divided into subsystems such as reaction, separation and heat integration. These subsystems can, in turn, be divided into smaller ones, either at the level of the paths between unit operations, or choices of different technologies among them. Along the history of process system engineering there has been different alternatives approaches to the Process Synthesis problem. Some of them are: a) Total enumeration that is limited to problems with a reduced number of alternatives. b) Graphical methods through physical insights c) Heuristic evolutionary search[2] that uses heuristics to generate a good base case design, and then modifies the flowsheet until no improvement is possible. However, the two more widely used approaches are d) Hierarchical decomposition developed by Douglas[3],[4], and superstructure optimization[5] In the first, the problem is addressed through hierarchical decisions and short cut models at various levels: batch versus continuous, input output structure of the flowsheet, recycle structure of the flowsheet, general structure of the separation system, (vapor and liquid recovery), and heat integration. If the process becomes unprofitable as the design proceeds, the search is terminated. In the case of superstructure optimization, a systematic representation is postulated in which all the alternatives of interest are embedded. The problem is then modeled and solved as a Generalized Disjunctive Programming Problem (GDP) or as a Mixed Integer (non)Linear Programming Problem (MINLP).

While hierarchical decomposition can address complex problems, it cannot guarantee to obtain the best solution because it is a sequential decomposition strategy, and therefore it does not take into account the interactions between the different levels of decomposition. As an example, Duran and Grossmann[6] and Lang et all[7] have shown that the simultaneous optimization and heat integration of process flowsheets generally produces improvements compared to the sequential approach. The GDP or MINLP techniques have shown to be powerful in the synthesis of subsystems: heat exchanger networks, mass exchanger networks, distillation sequencing, utility systems, etc[8]. Nowadays, however, process synthesis does not use a single strategy but a combination of hierarchical decomposition together with graphical techniques based on physical insights, and everything assisted by mathematical programming tools.

When applied to Process Synthesis, GDP and MINLP are limited to moderated size problems. The reasons are that the number of equations implied in chemical process models can be very large with hundreds or even thousands of integer (binary, Boolean) variables and with a large number of nonlinear and non convex equations that can prevent not only to finding the optimal solution but even finding a feasible point. The advances in the development of algorithms for MI(N)LP and GDP, global optimization, software and hardware have significantly increased the size of the problems and reduced the CPU time required in the solution and have produced interesting tools that allow the synthesis of chemical process plants using mathematical programming tools[9],[10],[11]. However, a rigorous modeling approach requires further advances in fields like global optimization, GDP algorithms and so on. In order to mitigate those problems, in conceptual design it is common to use shortcut or aggregated models that capture the essentials of the process with moderate numerical complexity. Also in some cases specially tailored algorithms have been developed to solve some specific problems[12]. The drawback of the shortcut models is that they have limited accuracy, and hence may predict unreliable results.

On the other hand, modular chemical process simulators include state of the art models for the most important units in chemical process industry with numerical methods especially tailored for each one of the unit operations together with large databases of physico-chemical thermodynamic and transport properties. Process simulators are also robust and reliable tools that are used extensively in process engineering.

Although nowadays most process simulators have optimization capabilities, they are able to deal only with problems involving continuous variables and smooth constraints with continuous domains. Optimization capabilities involving integer variables or discontinuous domains for the equations are, if any, very limited. Therefore, complex cost models or detailed size models included in some simulators can only be used 'a posteriori' after the simulation has been converged. In other words, the flowsheet was synthesized using approximate size and cost models as well as shortcut or aggregated models, and the further optimization of the operational conditions does not use the rigorous sizing or cost models due to their complex discontinuous nature.

In this paper we investigate different algorithms to integrate GDP and MINLP algorithms with existing process simulators in order to include complex cost and/or size functions, or in general complex equations defined over discontinuous domains. These functions can be in the form of explicit equations or implicit blocks (input-output black box relations). The structural optimization of process flowsheets –topology optimization- will be developed in future papers. Some works in this area can be found in references[13],[14],[15], although the most important theoretical aspects related with structural optimization will be addressed in next sections as well.

GDP formulation with implicit models. Application to a Modular Process Simulator

To get a clear representation of the different solution strategies it is convenient to differentiate the distinct kinds of variables present in an optimization problem in which there are implicit equations involved:

Design or independent variables (x_I). In a chemical process simulator these are the variables that must be specified to converge the flowsheet. The number of such variables matches the degrees of freedom in the flowsheet.

Variables calculated by the simulator (x_D) (or in general by any implicit model). The user has no direct control over these variables. In some process simulators it is possible to force some of these variables to take specific values through "auxiliary calculation blocks" –the name changes depending on the simulator-. These calculation blocks change some of the design variables until the specification is met. However, if the system is optimized, it is faster and usually numerically more reliable to introduce these specifications as constraints to the model.

Variables that must be fixed in a given topology of the flowsheet (u), for example number of trays in a distillation column, integer, binary (or Boolean) variables, etc. It is worth mentioning that we are referring to a subset of variables that must be fixed in a given iteration when solving a NLP problem with a given topology and a given set of fixed binary (Boolean or integer) variables. However, these variables can change from an iteration to another. The ways in which these variables are modified depend on the algorithm used to solve the problem. Due to the characteristics of some equipment, in some cases specially tailored algorithms are required as it is the case in distillation columns. Special algorithms for design distillation columns using process simulators were previously presented by Lang and Biegler[16] and by Caballero and Grossmann[17]

Variables that do not appear at the flowsheet level (or in other implicit block of equations) (z) but appear in explicit external constraints. No special treatment of these variables is required.

In a similar way, we can differentiate two classes of equations:

Implicit equations, These are all the equations solved by each of the modules in the process simulator, or any other third party module added to the model. These equations are usually considered “black box input-output” relationships because we have no access to the explicit equations.

A well known danger hidden in the implicit equations introduced in a gradient based optimization environment is that can have points in which some of these equations are non differentiable. Therefore, we must have a general knowledge of the system of equations in order to anticipate this behavior and correctly model it. For instance, a module developed to calculate the cost of a vessel could use different correlations depending on the value of the pressure design. This is not a problem in ‘a posteriori’ cost estimation, but in a gradient based optimization algorithm introduces discontinuities and therefore, unpredictable numerical behavior. The model must explicitly capture this behavior and correctly model the cost equations.

External or explicit equations, These are equations over which we have complete control. These equations can include dependent and independent variables. When the equations involve independent variables exact derivatives can be obtained, When they involve variables calculated by the simulator finite differences must be used which are not exact.

The disjunctive formulation of the problem can then be written as follows:

$$
\begin{aligned}
&\min: \ f(x_I, x_D, u, z) \\
&s.t.\ \ r_I(x_I, x_D, u) = 0 \\
&\quad\ s_E(x_I, x_D, u, z) = 0 \\
&\quad\ s_E(x_I, x_D, u, z) \le 0
\end{aligned}
$$

$$
\underset{i \in D_j}{\veebar} \begin{bmatrix} Y_{i,j} \\ h_I(x_I, x_D, u) = 0 \\ h_E(x_I, x_D, u, z) = 0 \\ g_E(x_I, x_D, u, z) \le 0 \end{bmatrix} \qquad \forall\, j \in J \tag{1}
$$

$$\begin{aligned}&\Omega(Y)=True\\&x_I \in X \subseteq \Re^n\\&Y \in \{True, False\}^m\end{aligned}$$

In equation (1) the index 'I' makes reference to the implicit equations, either in the process simulator or in a third party program; and the index 'E' to the explicit ones. J is a set of disjunctions and i makes reference to each one of the terms in the disjunction D_j.

The general disjunctive formulation given by equation (1) includes some particular cases of interest by themselves, or because they appear as subproblems in some solution algorithms.

If there are no disjunctions, or the value of the binary variables is fixed, the problem becomes in a non-linear programming problem with implicit equations. This is the case of optimizing a flowsheet with a fixed topology and smooth functions. Even though, the problem of solving a nonlinear optimization problem in a process simulator has been successfully addressed by different researchers [18], [19],[20],[21],[22],[23],[24], it is worth mentioning some relevant aspects. In modular simulators the equations are grouped in modules according to the physical process they represent; then these modules are solved sequentially, usually in the same way material flows through the process (HYSYS.Plant©[25] is a remarkable exception in which information is propagated through the systems as soon as it is generated). To solve the optimization problem the complete system is converged before calculating the constraints and objective function. However, because the flowsheet consists of black-box modules, simulation is usually performed by slow convergence techniques. Moreover, for optimization gradients can only be computed for independent variables in explicit equations while numerical approximations are used for the calculated variables.

In the three most common process simulation codes (ASPEN, PRO/II and HYSYS) the optimization problem is solved first calculating the process models before evaluating the constraints and objective function . The optimization problem is solved in an outer loop while the model equations are converged in an inner loop. Therefore, at least a single process model evaluation is required every time the objective and constraint functions are evaluated for optimization.

If there are no implicit equations inside the disjunctions, then the problem can be reformulated as a MINLP using a big M or a convex hull reformulation. If a pure branch and bound algorithm is used then no especial care is needed. However, if a decomposition algorithm, like outer approximation[26] or the LP-NLP based branch and bound[27] (an intermediate situation between pure BB and outer approximation), is used then generating the master MILP problem is not trivial.

If there are implicit equations inside the disjunctions we can differentiate two cases. First, the implicit equation makes reference to some block (unit operation) that could eventually produce numerical problems or to implicit equations, calculated by the process simulator or by another third party program. A small example will illustrate this point: A typical disjunction in process synthesis is:

$$\begin{bmatrix} Y_{unit} \\ Unit\ equations \end{bmatrix} \underline{\vee} \begin{bmatrix} \neg Y_{unit} \\ Relevant\ vaiables = 0 \end{bmatrix} \quad (2)$$

The left term in the previous disjunction states that if the unit operation is selected, then we must calculate all the equations associated with it. The right term states that if the unit operation is not selected a subset of variables (we have called them the *relevant variables*) must be set to zero. Among these variables are the inlet and outlet flows. However, if we try to set to zero the inlet flows in a unit operation the process simulator is likely to stop with warning messages due to convergence failure. Fixing the variables to a small value can work in some situations, but again depending on the simulator and on the unit operation, unexpected results can be obtained. Even more, fixing the output flows to zero can affect parts of the rest of the flowsheet. Therefore, this situation must be anticipated and zero flows avoided. As a second point, if the implicit equations make reference to blocks that will not produce numerical problems then no special care is needed.

Adapted Algorithms.

In this work we have adapted some algorithms that represent the state of the art in solving MINLP problems: A Regular and a Disjunctive Branch and Bound (BB) [28] ,[29]; Outer Approximation (OA) with MINLP reformulation;,[30],[31] LP-NLP based Branch and Bound (LP/NLP-BB).

In all the branch and bound based algorithms we have to reformulate the problem (or parts) as a regular MI(N)LP using a big M or a convex hull reformulation. In any case, we have to solve a series of NLP problems in which a subset of binary (Boolean) variables are fixed and the rest are relaxed to be between 0 and 1. The NLPs solved take the general form given by the next equation:

$$\begin{aligned}
&\min: f(x_I, x_D(x_I), u, z) \\
&s.t.\ r_I(x_I, x_D(x_I), u) = 0 && (a) \\
&\quad r_E(x_I, x_D(x_I), u, z) = 0 && (b) \\
&\quad s_E(x_I, x_D(x_I), u, z) \le 0 && (c) \\
&\quad h_E^*\left(x_I, x_D(x_I), u, z, y_{i,j}\right) = 0 && (d) \\
&\quad g_E^*\left(x_I, x_D(x_I), u, z,\ y_{i,j}\right) \le 0 && (e) \\
&-M(1-y_{i,j}) \le h_I^*\left(x_I, x_D(x_I), u, y_{i,j}\right) \le M(1-y_{i,j}) && (f) \\
&\quad A y - b \le 0 \\
&\quad 0 \le y_{ij} \le 1 \quad \{i,j\} \in B
\end{aligned} \qquad (3)$$

Where equation 3.a refers to the implicit equations (i.e. at the level of process simulators). Usually, but not always, this equations come in the form $x_D = \Theta(x_I, u)$ and can be explicitly removed –recycles in a flowsheet is an example in which those equations cannot be removed-. Equations 3.b and 3.c are the rest of explicit equations that do not depend on the disjunction. Equations 3.d and 3.e are the convex hull, big M or any other valid MINLP reformulation.

Equation 3.f refers to the big M reformulation for implicit equations inside the disjunctions, if those equations cannot be explicitly removed by $x_D = \Theta_I(x_I, u)$. The set B makes reference to those binaries that are not fixed in a given iteration.

The OA and LP/NLP-BB algorithms iterate between two different problems, an NLP like that in equation (3) with fixed $y_{i,j}$ and a Master (MILP) problem that is obtained from the linearizations of the constraints and the objective function[30, 31]:

$$
\min_{\alpha, x_I, s} \; \alpha + \Pi^T (s_1 + s_2 + s_3 + s_4)
$$

$$
\left.
\begin{aligned}
s.t.\;\; \alpha \ge \; & f\left[x_I^k, x_D(x_I^k), u, z^k\right] + \nabla f\left[x_I^k, x_D(x_I^k), u, z^k\right] \begin{pmatrix} x_I - x_I^k \\ z - z^k \end{pmatrix} \\
& \lambda_1^T \left\{ \nabla r_E[x_I^k, x_D(x_I^k), u, z^k] \begin{pmatrix} x_I - x_I^k \\ z - z^k \end{pmatrix} \right\} \le s_1 \\
& s_E[x_I, x_D(x_I), u, z^k] + \nabla s_E[x_I, x_D(x_I), u, z^k] \begin{pmatrix} x_I - x_I^k \\ z - z^k \end{pmatrix} \le s_2 \\
& \lambda_2^T \left\{ \nabla h_E^*[x_I^k, x_D(x_I^k), u, z^k, y_{i,j}^k] \begin{pmatrix} x_I - x_I^k \\ y_{i,j} - y_{i,j}^k \\ z - z^k \end{pmatrix} \right\} \le s_3 \\
& g_E^*[x_I^k, x_D(x_I^k), u, z^k, y_{i,j}^k] + \nabla g_E^*[x_I^k, x_D(x_I^k), u, z^k, y_{i,j}^k] \begin{pmatrix} x_I - x_I^k \\ y_{i,j} - y_{i,j}^k \\ z - z^k \end{pmatrix} \le s
\end{aligned}
\right\} \; k = 1..K
$$

$$
Ay - b \le 0; \;\; (s_1,\; s_2,\; s_3) \ge 0 \quad ; \quad y \in \{0,1\}^m
$$

(4)

where λ is a vector formed by the signs of the Lagrange multipliers of equality constrains in the solution of last NLP subproblem. s_1, s_2 and s_3 are positive vectors of slacks variables introduced to minimize the effect of non-convexities, and Π is a vector of penalty parameters.

It is worth mentioning that in the previous formulation all implicit equations have been removed since the Master problem does not depends on dependent variables. Linearizations of dependent variables can be done applying the chain rule through the entire flowsheet or the implicit equations and therefore all the problem depends only on independent, slacks, variables that do not appear in the flowsheet (z) and the variable α, used to transfer the objective function to the constraints.

$$
\nabla_{x_I} f = \frac{\partial f}{\partial x_I} + \frac{\partial f}{\partial x_D} \frac{d x_D}{d x_I} \qquad (5)
$$

As commented in a previous section a critical step is to obtain accurate derivatives and all precautions previously commented must also be taken into account here.

Both, the OA and LP/NLP-BB algorithms start by transforming the problem into a MINLP using a Big M or a convex hull reformulations and solving an initial NLP problem in which the binary variables are relaxed to continuous variables that lie between 0 and 1. This approach can eventually produce numerical problems with the implicit blocks inside disjunctions if the user does not have complete control over those blocks of equations (i.e. a unit operation disappearing from the superstructure). In this case logic based algorithms must be adapted. However, in this paper, as commented before, we focus only on the case in which there are no implicit equations inside the disjunctions (i.e. disappearing units) or in the case in which we have control over those equations and then eventual numerical problems are avoided (discontinuous size and cost correlations). From the solution of this first NLP a Master problem is generated.

OA solves the Master to optimality (or to a given pre-specify tolerance). Then, a new NLP is solved with the values of the binaries obtained from the last Master fixed, that is a solution to the problem and then an upper bound to the optimal solution. A new Master is generated by accumulation of linearizations (constraints of previous Masters plus the linearizations of the last NLP). The procedure continues until in two consecutive iterations there is no improvements in the upper bound .

In the LP/NLP-BB algorithm the idea is no re-starting the MILP from the beginning. Therefore, when an integer solution is found a NLP, with the binaries fixed to the integer solution, is solved. The solution of this NLP is an upper bound to the optimal one. All the open nodes in the MILP-Master are updated with new linearizations obtained from the last NLP and the branch and bound continues without re-starting the tree search. The trade-off, however, is that the number of NLP problems may increase, but computational experience indicates that the number of NLP problems remains unchanged

Implementation Details

In this work we have used the public version of HYSYS.Plant for performing the simulations. The NLP subproblems were solved using external to HYSYS, state of the art NLP, solvers (CONOPT[32], SNOPT[33]) through an activeX client-server application. All the process is controlled by MATLAB[34]. Cost and size models were also developed in MATLAB as third party implicit models or through explicit equations. Although it would be possible to use the HYSYS internal NLP solvers we obtain a large degree of flexibility dealing with explicit constraints, master problem generation, etc, when using external solvers.

Figure 1 shows an scheme of the implementation. The first step is at the level of process simulator. Here we have to set up the flowsheet (or the superstructure if it was the case), determine the degrees of freedom and decide which are the independent variables in the flowsheet among the options available. In this point HYSYS has an important advantage over other process simulators due to the way in which the flowsheet is calculated. In general, modular simulators implement a rigid input-output structure, in other words, the user must provide information of the inputs to a unit operation and internal specifications and the simulator calculates the outputs. However, in HYSYS, as soon as all the degrees of freedom of a unit are satisfied that unit is calculated, and the information propagated forward and backward . So it is possible calculate inputs in terms of outputs, or any feasible input-output combination. Of course, there are exceptions like distillation columns where the usual approach is followed. It is convenient to start with a converged flowsheet, even though the other external constrains were initially violated. We let the optimizer to converge all those constraints as the same time that is searching for the optimum.

The second step consists of writing the mathematical programming model, that includes explicit constraints (sizing and cost models), third party implicit models (other input-output models not included in the process simulator), etc. these new constraints could be only in terms of the independent and/or dependent variables that previously appear in the flowsheet and also in terms of new external variables. The second case is the usual one. In order to deal with those new variables there are two approaches: divide the variables into dependent and independent, like in the variables at the flowsheet level, or simply let the optimizer to deal with the variables like in any other optimization problem. The latter approach is usually more efficient because it avoids eliminating constraints. If the problem has also third party blocks of implicit equations that cannot be removed – a subset of variables, equal to the number of equations, written in terms of the rest $x_D = \Theta_I(x_I, u)$ - then the same approach followed with equations at process flowsheet level must be used.

If the model is solved using an MINLP solver, as is the case in this paper, a valid reformulation using a big M or a convex hull must be used. For linear equations a convex hull reformulation is used, in the case of nonlinear equations, that include all the implicit equations, we use a big M reformulation. Although it would be possible to use a non-linear convex hull reformulation it introduces numerical difficulties when some of the binaries take zero values. A correct implementation of the non-linear convex hull was introduced by Sawaya & Grossmann[35] but it is not easy to apply in the case of implicit blocks of equations. Therefore, to avoid numerical problems in nonlinear equations or implicit blocks we implement only a big M reformulation. The equations below show a general disjunction an valid MINLP reformulation:

$$
\bigvee_{i \in D} \begin{bmatrix} Y_i \\ A_i x_I^L + B_i z^L + b_i \le 0 \\ \Phi_{E,i}(x_I^N, x_D, u, z^N) \le 0 \\ \Phi_{I,i}(x_I^N, x_D, u) = 0 \\ \left(x_I^L\right)_i^{LO} \le x_I^L \le \left(x_I^L\right)_i^{UP} \\ \left(z^L\right)_i^{LO} \le z^L \le \left(z^L\right)_i^{UP} \end{bmatrix}
\qquad
\begin{array}{l}
\sum_{i \in D} y_i = 1 \\
x_I^L = \sum_{i \in D} v_i \\
z^L = \sum_{i \in D} w_i \\
A_i v_i + B_i w_i + b_i y_i \le 0 \\
y_i \left(x_I^L\right)_i^{LO} \le v_i \le y_i \left(x_I^L\right)_i^{UP} \\
y_i \left(z^L\right)_i^{LO} \le w_i \le y_i \left(z^L\right)_i^{UP} \\
\Phi_{E,i}(x_I^N, x_D, u, z^N) \le M(1 - y_i) \\
\Phi_{I,i}(x_I^N, x_D, u,) \le M(1 - y_i) \\
\Phi_{I,i}(x_I^N, x_D, u) \ge -M(1 - y_i) \\
y_i \in \{0,1\}
\end{array}
\qquad (6)
$$

In the equations above the superscript ‘L’ refers to the subset of independent and/or z variables that are linear inside the disjunction. The superscript ‘N’ refers to the subset of independent and/or z variables that are nonlinear in the disjunction. Note that if there is no further information all dependent variables should be considered nonlinear because they are calculated through a black box relation. Superscripts ‘LO’ and ‘UP’ make reference to the lower and upper bounds of the variables. Note that the convex hull reformulation for the linear constraints requires bounds on the linear variables inside the disjunction but it is not necessary for the big M reformulation. Finally ‘v’ and ‘w’ are the disaggregated variables.

The two first steps, model formulation at the level of flowsheet and model formulation of the explicit external constraints and other implicit blocks, are the only that are not completely automated. The rest have been implemented in a way that the user only has to select among the available options.

Once the model has been specified, the next step is connecting the process simulator with the rest of the model. In this work we used a client-server application through the windows Component Object Model (COM) interface. All the process is controlled from MATLAB where the different algorithms were implemented. Derivatives calculation, master generation, etc are performed by MATLAB automatically in our current implementation. NLP subproblems and Master problems are solved using TOMLAB-MATLAB[36] –an interface for accessing state of the art NLP or MILP solvers. In this paper we have used Conopt and Snopt as NLP solvers and CPLEX as MILP solver.

The most time consuming task in all the process is the convergence of the flowsheet each time that the objective function and constraints are evaluated. In order to speed up the calculations it is necessary to optimize the number of calls to the flowsheet. The major number of flowsheet evaluations is required when derivatives are calculated. There are two aspects to take into account: separating variables that affect the flowsheet from those that only appear in explicit constraints to avoid unnecessary calls to the flowsheet, and second if the perturbation parameter to calculate derivatives numerically are compatible (same order of magnitude) try to take advantage of the sparsity pattern of the model and calculate some columns of the Jacobian matrix simultaneously. A good option is the CPR algorithm [37] .

Examples

Example 1. Three heat exchangers network.

The first example is an adaptation of an example by Turkay and Grossmann[38]. It consists of a small heat exchangers network. There are two streams in the network: a hot stream to be cooled from 500 to 430 K and a cold stream to be heated from 350 to 560 K. Cooling water and High pressure steam at 600 K are available as cooling and heating utilities. All necessary data for the example is given in Table 1. As shown in Figure 2 the network consist of three heat exchangers; the first one (E-101) exchanges heat between the hot and cold streams, the second one (E-102) cools the hot stream with cooling water, and the third one (E-103) heats the cold stream with steam to the exit temperature. Since the network structure is fixed, the variables to be determined are heat loads, areas of the heat exchangers and unknown temperatures (HotStream T1 and ColdStream T2, in Figure 2). The objective function includes both the investment and utility costs. The cost of each heat exchanger is given by a discontinuous cost function in terms of the heat transfer area of each heat exchanger.

The first step is to do a degree of freedom analysis at the level of flowsheet. Although in most situations this analysis is not easy, in general is very straightforward when using a process simulator. Once all the data from the problem has been introduced, the degrees of freedom are equal to the number of extra specifications we need to introduce to converge the flowsheet. Process simulators usually include tools to detect over-specifications, inconsistencies and so forth which facilitates a lot this stage. In this first small example there is only one degree of freedom, and we decided that the heat exchange area in the first heat exchanger to be the independent variable. A disjunctive formulation of this problem is then as follows

$$
\begin{aligned}
&\min: \quad \sum_{j \in HE} IC_j + Cost_{steam} \cdot W_{steam} + Cost_{water} \cdot W_{water} \\
&s.t. \\
&\begin{bmatrix} Y_{j,1} \\ IC_j = 2750\ A_j^{0.6} + 3000 \\ 1 \le A_j \le 10 \end{bmatrix} \veebar \begin{bmatrix} Y_{j,2} \\ IC_j = 1500\ A_j^{0.6} + 15000 \\ 10 \le A_j \le 25 \end{bmatrix} \veebar \begin{bmatrix} Y_{j,3} \\ IC_j = 600\ A_j^{0.6} + 46500 \\ 25 \le A_j \le 50 \end{bmatrix} j \in HE \\
&\qquad T_{HotStreamT1} \ge T_{out\_water} + 10 \\
&\qquad \begin{pmatrix} T_{HotStreamT1} \\ T_{out\_water} \\ A_2 \\ A_3 \\ W_{steam} \\ W_{water} \end{pmatrix} = \Theta(A_1) \qquad (7) \\
&HE = \{ j \mid j \ is\ a\ heat\ exchanger \}; \quad Y \in \{True, False\}^9
\end{aligned}
$$

In equations (7), W is the heat load needed of steam or water (kW). IC is the annualized investment cost of each heat exchanger and □(·) makes reference to the flowsheet equations to calculate all the other variables needed in the model (Areas, heat loads, and unknown stream temperatures).

In the disjunctive model given in equation (7) there are no implicit equations inside the disjunctions. The problem was transformed into an MINLP using a big M reformulation and solved using BB, OA and LP/NLP-BB algorithms. In all cases the final results were the same, but the performance of the algorithms was very different. Since the bottleneck is the time spent calculating NLP subproblems, the BB algorithms produces the worst results in terms of CPU time (49.2 s for solving 37 NLPs –Nodes-). Both OA and LP/NLP-BB have similar performance OA take 5.2 seconds and 4 major iterations, while LP/NLP-BB takes 4 seconds and solved three NLP subproblems. One characteristic of this problem, that is also common to problems with discontinuous size regions, is that there is a relatively large number of combinations for the binary variables that produce infeasible solutions, even more if the relaxed solution is not very tight. Therefore, the master can predict combinations of binary variables that produce infeasible NLPs. –This is the case in the second major iteration in OA. Curiously, In the LP/NLP-BB algorithm this effect is not observed, maybe because the continuous update of the master as soon as an integer solution is found tend to minimize this effect because the Master approximate faster than with OA the feasible region. Table 2 gives some statistics of the problem using the different algorithms.

The optimal solution yields a total annual cost of 150322 $/year. Table 3 shows a summary of the optimal results. The initial relaxed NLP gave an initial objective function of 49259 $/year. In this case the relaxation gap is not very tight (67.2%) -although the fact that the three algorithms produce the same result indicates that it is likely the global optimal solution-. In general, improvements in the model formulation that reduce the relaxation gap tend to reduce the CPU time and, if the initial point is good enough, it increases the probability of obtaining the global optimum. In this example, since the cost equations are separable, it is possible to reformulate them using a piecewise linear approximation. Approximating each term in each disjunction by three linear terms is then possible to use a convex hull of the linear approximations. With this approach the initial relaxed NLP gives a total cost of 126779 $/year (gap 15.6%), much better that with the big M reformulation. However, the total CPU time was greater than with the original formulation. The reason is because the number of variables is considerably larger due to the piecewise linear approximation and the disaggregated variables in the convex hull reformulation, and also because the initial problem is small and relatively easy to solve.

Example 2

This example is a modification of a problem proposed by Seider et al[39] and consists of the design of a natural gas plant. It is required to process a natural gas stream at 5000 kmol/h, 20ºC and 1000 kPa. The gaseous product is required at 1500 kPa with at least 4900 kmol/h of nC_4 and lighter products and a combined mole percentage of at least 99.5%. The flowsheet is shown in Figure 3. The feed initially at 1000 kPa is compressed. The final pressure is one of the optimization variables. In the compressor we can choose between an electric engine or a combustion engine using fuel oil. The compressed stream (S0) is cooled in two stages using Coolers 1 and 2. Cooler 1 could be an air cooler or a shell and tubes heat exchanger using either water or cool water (see Table 4). In Cooler 2 the stream is cooled at temperatures under 0ºC and we can choose between three different refrigeration systems (R1, R2 or R3 in Table 4). The stream is flashed in Flash1 and the vapor and liquid streams heated using Heaters 1 and 2 respectively. In Heater 1 we can use hot water or Low Pressure steam. In Heater 2 we use hot water. This last stream (S5) is flashed again in the Flash 2 unit. The liquid stream exiting from the flash is sent to a distillation column. In the condenser of the distillation column we can use water, or refrigerants R1, R2 or R3. In the reboiler we can choose between medium pressure or high pressure steam. The distillate is mixed with the vapor streams from the flash units (streams S6 and S8) to form the final product.

Again, in this example the process simulator, HYSYS.Plant©, performs the basic calculations at the flowsheet level, including all mass and energy balances and properties estimation. However, size and cost calculations, that depend on the type of equipment, are calculated as implicit external functions developed in Matlab©, but with all basic data extracted from HYSYS through its COM communication capability.

Note, that although in the process simulator some equipments are represented by a general unit operation (i.e. heat exchanger), the cost and size of those equipments depends on the actual equipment; an air cooler is different from a floating head tube and shell exchanger. Therefore there are two kinds of implicit equations over which we have different control. The implicit equations associated to the basic flowsheet and solved by the process simulator and the size and cost equations over which we have full control. The reasons of using these equations as implicit are : a) They decrease the dimensionality of the problem at the optimization level, and b) the numerical behavior is better when the model is solved with a decomposition algorithm because linearizations are constrained to the input-output variables and not to all the intermediate non-convex equations reducing the possible effects of cutting parts of the feasible region due to linearizations.

Reducing the number of equations in a Master problem has always been an issue in MINLP optimization. Therefore, to take advantage of the physical structure of the problem, defining it in terms only of the independent variables for each unit operation is a natural way of reducing the dimensionality of the master while it assures that the reduced master is a valid one, without taking into account any other mathematical consideration**.**

Table 5 shows all the data needed in the problem specification. The objective in this example is minimize the total annualized cost that includes the annualized investment and the utilities costs. A disjunctive conceptual representation of the model showing the different alternatives is as follow:

Objective function

$$\min: TAC$$

$$TAC = \frac{i\,(i+1)^{PL}}{(i+1)^{PL}-1} Investment\ Cost\ +\ Annual\,Utilities\,Cost$$

$$\begin{aligned} Investment\ Cost = {} & Cost\_Compressor + Cost\_Engine + Cost\_Ref1 + \\ & Cost\_Ref2 + Cost\_Heat1 + Cost\_Heat2 + Cost\_Flash1 + \\ & Cost\_Flash2 + Cost\_Heat2 + Cost\_Flash1 + Cost\_Flash2 + \\ & Cost\_Cond + Cost\_Reb + Cost\_Column\_Vessel + \\ & Cost\_Column\_Internals. \end{aligned} \tag{8.OBJ}$$

External specifications to the problem:

$$\begin{aligned} & T_{S1} + 10 \le T_{S0} \\ & T_{L1} \ge -60 \\ & T_{V2} \le 240 \\ & Flow_{GAS} \ge 4900 \\ & \sum_{i \in I} Gas_{molar\_fraction\ i} \ge 0.995 \\ & T_{GAS} \ge 20 \\ & \quad I = \{nC_4\ and\ lighter\ \ products\} \end{aligned} \tag{8.ESP}$$

where $interest\ rate\ i = 0.08;\ \ plant\ life\ PL = 8$

In equation 8.ESP T is the temperature in Celsius. Stream L1 is the liquid stream leaving the condenser in the distillation column. V2 is the vapor stream leaving the reboiler in the distillation column. GAS is the final product stream. The previous equations comes from problem specifications or constraints to assure feasible heat exchange.

The implicit blocks of equations that are not at the level of process simulator are calculated as a MATLAB functions. All physical properties are extracted from the process simulator:

$$\begin{aligned}
&Cost\_Compressor = C_{comp}(W\_Compressor)\\
&Cost\_Flash1 = C_{Flash}(VFS_3, VFS_4, \rho_3, \rho_4)\\
&Cost\_Flash2 = C_{Flash}(VFS_6, VFS_7, \rho_7, \rho_7)\\
&Cost\_Column\_Vessel = C_{Vessel}(H, D)\\
&Cost\_Column\_Internals = C_{Int}(D, N_t)
\end{aligned} \qquad \text{(8-IMP)}$$

Equations calculated by the process simulator:

$$x_D = \Theta(x_I) \qquad \text{(8.I)}$$

In equation 8.I x_D makes reference to all the dependent variables calculated by the process simulators. This includes physical properties of streams, heat loads, some pressures, some temperatures, compositions, etc. In this example there are 7 independent variables at flowsheet level: Pressure in stream S0 (PS0); Temperatures in streams S1, S2, S8 and S5 (TS1, TS2, TS8, TS5) and recoveries of key components in distillation column (REC1, REC2)

Disjunctions related with each one of the discrete decisions. Inside the disjunctions there are implicit equations calculated by blocks of equations in MATLAB. The models have been written to allow zero flows without numerical errors. However, since cost correlations are only valid for some size intervals, if any variable is out of bounds the module produces a warning message. It is possible to introduce explicitly a constrain in order to assure that all the variables are inside the valid limits, but in our examples it has not been necessary.

$$\begin{bmatrix} Y_{Electric\ Engine} \\ Cost_{Eng} = C_{Elc}(W) \\ Cost_{Utility\_Eng} = Cost_{Electricity} \end{bmatrix} \underline{\vee} \begin{bmatrix} Y_{Combustion\ Engine} \\ Cost_{Eng} = C_{comb}(W) \\ Cost_{Utility\_Eng} = Cost_{Fuel} \end{bmatrix} \qquad \text{(8.D1)}$$

$$\begin{bmatrix} Y_{Cooler1\_Air} \\ Area_{Cooler1} = A_{Cooler1\_Air}(TS0, TS1, W_{Ref1}) \\ Cost_{Ref1} = C_{Cooler\_Air}(Area_{Cooler1}) \\ Cost_{Utility\_Cooler1} = Cost_{Air} \\ TS0 \geq 45^{o}C;\ TS1 \geq 40^{o}C \end{bmatrix} \veebar \begin{bmatrix} Y_{Cooler1\_Water} \\ Area_{Cooler1} = A_{Cooler1}(TS0, TS1, W_{Cooler1}) \\ Cost_{Utility\_Cooler1} = Cost_{Water} \\ TS0 \geq 35^{o}C;\ TS1 \geq 30^{o}C \end{bmatrix} \veebar$$

$$\begin{bmatrix} Y_{Coole1\_ColdWater} \\ Area_{Cooler1} = A_{Cooler1}(TS0, TS1, W_{Cooler1}) \\ Cost_{Utility\_Cooler1} = Cost_{Cold\_Water} \\ TS0 \geq 20^{o}C;\ TS1 \geq 15^{o}C \end{bmatrix}$$

(8.D2)

$$\begin{bmatrix} Y_{Cooler2\_R1} \\ Area_{Cooler2} = A_{Cooler2}(TS1, TS2, W_{Cooler2}) \\ Cost_{Cooler2} = C_{Cooler2}(Area_{Cooler2}) \\ TS1 \geq -21^{o}C;\ TS2 \geq -22^{o}C \end{bmatrix} \veebar \begin{bmatrix} Y_{Cooler2\_R2} \\ Area_{Cooler2} = A_{Cooler2}(TS1, TS2, W_{Cooler2}) \\ Cost_{Cooler2} = C_{Cooler2}(Area_{Cooler2}) \\ TS1 \geq -36^{o}C;\ TS2 \geq -37^{o}C \end{bmatrix} \veebar$$

$$\begin{bmatrix} Y_{Cooler2\_R3} \\ Area_{Cooler2} = A_{Cooler2}(TS1, TS2, W_{Cooler2}) \\ Cost_{Cooler2} = C_{Cooler2}(Area_{Cooler2}) \\ TS1 \geq -61^{o}C;\ TS2 \geq -62^{o}C \end{bmatrix}$$

(8.D3)

$$\begin{bmatrix} Y_{Heat1\_LP} \\ Area_{Heat1} = A_{Heat1}(TS3, TS8, W_{Heat1}) \\ Cost_{Heat1} = C_{Heat1}(Area_{Heat1}) \\ Cost_{Utility\_Heat1} = Cost_{LP} \end{bmatrix} \veebar \begin{bmatrix} Y_{Heat1\_HotWater} \\ Area_{Heat1} = A_{Heat1}(TS3, TS8, W_{Heat1}) \\ Cost_{Heat1} = C_{Heat1}(Area_{Heat1}) \\ Cost_{Utility\_Heat1} = Cost_{HotWater} \end{bmatrix}$$

(8.D4)

$$\begin{bmatrix} Y_{CondWater} \\ Area_{Cond} = A_{Cond}(TV1, TL1, W_{Cond}) \\ Cost_{Cond} = C_{cond}(Area_{Cond}) \\ Cost_{UtilityCond} = Cost_{Water} \end{bmatrix} \veebar \begin{bmatrix} Y_{CondR1} \\ Area_{Cond} = A_{Cond}(TV1, TL1, W_{Cond}) \\ Cost_{Cond} = C_{cond}(Area_{Cond}) \\ Cost_{UtilityCond} = Cost_{R1} \end{bmatrix} \veebar$$

$$\begin{bmatrix} Y_{CondR2} \\ Area_{Cond} = A_{Cond}(TV1, TL1, W_{Cond}) \\ Cost_{Cond} = C_{cond}(Area_{Cond}) \\ Cost_{UtilityCond} = Cost_{R2} \end{bmatrix} \veebar \begin{bmatrix} Y_{CondR3} \\ Area_{Cond} = A_{Cond}(TV1, TL1, W_{Cond}) \\ Cost_{Cond} = C_{cond}(Area_{Cond}) \\ Cost_{UtilityCond} = Cost_{R3} \end{bmatrix}$$

(8.D5)

$$\begin{bmatrix} Y_{Reb\_MP} \\ Area_{Reb} = A_{Reb}(TV2, TL2, W_{Reb}) \\ Cost_{Reb} = C_{Reb}(Area_{Reb}) \\ Cost_{Utility\_Reb} = Cost_{MP} \end{bmatrix} \underline{\vee} \begin{bmatrix} Y_{Reb\_HP} \\ Area_{Reb} = A_{Reb}(TV2, TL2, W_{Reb}) \\ Cost_{Reb} = C_{Reb}(Area_{Reb}) \\ Cost_{Utility\_Reb} = Cost_{HP} \end{bmatrix} \tag{8.D6}$$

Disjunctions 8.D1-8.D6 make reference to the different alternatives considered : D1 compressor engine; D2 Cooler 1; D3 Cooler 2; D4 Heater 1: D5 Condenser and D6 Reboiler.

The previous equations were written as an MINLP problem using a big M reformulation, for nonlinear equations and implicit blocks, and convex hull for the linear equations inside the disjunctions.

As commented in example 1, the bottleneck in the solving procedure is the time consumed in communications between Matlab and Hysys. Therefore, to reduce the number of NLP subproblems, this example and the next one have been solved using only the OA and LP/NLP-BB algorithms.

Results for the best obtained solutions are shown in Table 6. Some statistics about the reformulated problem are given in Table 7. It is worth mentioning that besides all the equations solved by the process simulator there are 40 implicit blocks of equations, most of them inside the disjunctions.

The most remarkable result is that even though the problem is reformulated using a big M approach the relaxation gap is only about 13% (Initial relaxed NLP 1015 10^3 \$/year and optimal solution 1170.5 10^3 \$/year). A similar behavior appears in example 3 that we will comment in next paragraphs. This is a very interesting result. A possible explanation could be the following: In a pure equation oriented environment, when a problem is reformulated using a big M approach, the effect of the relaxation is that some mass and/or energy balances, equilibrium equations, etc can be violated. However, in our examples all the mass balances, equilibrium equations, cost and size correlations, etc must hold. In other words, we relax blocks of equations but inside each block (and this includes the entire flowsheet) all the equations are satisfied. Although a couple of examples are not significant, and further studies are needed, some previous results show that previous explanation could be correct. i.e. Karuppiah and Grossmann[40] noted that simply adding a global mass balance in a superstructure optimization the relaxation gap is significantly reduced. In our examples mass and energy balances are always satisfied.

Example 3

This example is for the synthesis of Dimethylether (DME). DME is produced by dehydrogenation over a catalytic zeolite. The main reaction is:

$$2CH_3OH \longrightarrow (CH_3)_2O + H_2O \tag{9}$$

The reaction temperature can vary between 225 and 400 ºC and is carried out in an adiabatic reactor. Methanol is introduced to the system at 25 ºC mixed with some water (Table 8 shows all data for the example), compressed to 1500 kPa and then mixed with the recycle stream coming from the separation. This mixture is vaporized in the heater (See Figure 4) and pre-heated in the heat exchanger (HE) before coming into the reactor. The reactor exit stream is cooled (Cooler), partially decompressed and introduced in the separation train. The DME is obtained in the distillate of the first column with a purity higher than 99.5% (molar basis). The bottoms of the first column are decompressed again and introduced in a second distillation column that separates water from methanol. Water is sent to a treatment section to remove traces of organic compounds (not represented in the flowsheet) and the methanol recycled.

The discrete options considered in this example are the following:

- For the Heater, the heat exchanger (HE) and the condensers of both columns we can choose between a double pipe heat exchanger, a multiple pipe and a floating head. Costs and sizing equations are both implicit.
- For the Cooler there are two options using an Air Cooler or a Tube and pipes heat exchangers.
- For the Reboilers of both columns it is possible to choose between a Kettle reboiler and a floating head heat exchanger.
- The columns can be packed or with sieve trays.

The objective is to minimize the total annualized cost. Data of utilities are the same as in example 2 (Table 4). The independent variables at the flowsheet level are inlet temperature to the Heat Exchanger (HE), inlet temperature to the reactor; Pressures at the exists of valves; Recoveries of DME by head and methanol in bottoms in column 1; Recoveries of Methanol and Water in column 2; Temperature, Flow and compositions in stream S2 to converge the recycle.

An important difference with the previous example is related with the recycle stream. In example 2 all dependent variables at the flowsheet level could be calculated in a single flowsheet evaluation. In other words it was possible to write $x_D = \Theta(x_I)$. However, with the variables associated with the recycle it is not possible to do this variable elimination. To deal with recycle streams there are two possible approaches: a) Let the process simulators converge the flowsheet including the recycles, or b) Let the optimizer converge the recycles while solving the rest of the NLP problem. The first approach has the advantage of a smaller number of variables (all those variables related with the recycle) but has two important drawbacks. Each flowsheet evaluation is much slower due to the convergence of the recycles and the recycle streams increases the noise in the variables inside the cycle when estimating derivatives, with the undesirable effect of increase the CPU calculation time. In the second approach the number of variables and constraints increases in the NLP problem, but avoids the drawbacks of the previous approach. In general, the second alternative has better numerical performance and it is the alternative we used in this example.

A Disjunctive conceptual representation of the model showing the different alternatives is as follow:

$$\min: TAC$$

$$TAC = \frac{i\,(i+1)^{PL}}{(i+1)^{PL}-1} Investment\ Cost + Annual\,Utilities\,Cost$$

$$\begin{aligned} Investment\ Cost = {} & Cost\_HE + Cost\_Heater + Cost\_Cooler + \\ & Cost\_Pump1 + Cost\_Pump2 + Cost\_Reactor + \\ & Cost\_Condenser1 + Cost\_Condenser2 + Cost\_Reboiler1 + \\ & Cost\_\mathrm{Re}boiler2 + Cost\_Column\_Vessel1 + Cost\_Column\_Vessel2 + \\ & Cost\_Column\_Internals1 + Cost\_Column\_Internals2 \end{aligned} \quad (10.OBJ)$$

External specifications and other explicit equations:

$$\begin{aligned} & x_{DME} \geq 0.995 \\ & x_{Water} \geq 0.995 \\ & T_{S2} = T_{Recycle} \\ & Flow_{S2} = Flow_{Recycle} \\ & \left.\begin{aligned} & x^{i}_{S2} = x^{i}_{Recycle} \\ & \sum_i x^{i}_{S2} = 1 \end{aligned}\right\} \quad i = \{DME,\ MeOH,\ Water) \end{aligned} \quad (10.ESP)$$

Where x refers to mol fractions and T to temperatures, and $interest\ rate\ i = 0.08;\ \ plant\ life\ PL = 8$. The last four equations in (10.ESP) are used to converge the recycle stream.

Implicit blocks of equation of fixed equipment which are not calculated by the process simulator:

$$\begin{aligned} & Cost\_Pump\_1 = C_{Pump1}\,(Power\_P1, P_{S1}) \\ & Cost\_Pump\_2 = C_{Pump2}\,(Power\_P2, P_{\mathrm{Re}cycle}) \\ & [Diameter, Height]_{reactor} = Size_i\,(P_{reactor}) \\ & Cost\_Reactor = Cost\_Vessel\,(Diameter, Height) \end{aligned} \quad (10.IMP)$$

Equations calculated by the process simulator:

$$x_D = \Theta(x_I) \quad (10.I)$$

Note that in equation (10.I) the convergence of the recycle streams are excluded because those equations are written in explicit form in equation (10.ESP).

The disjunctions for the discrete decisions are:

$$\begin{bmatrix} Y^{i}_{Double\ Pipe} \\ A_i = f\left(T_H^{in}, T_H^{out}, T_C^{in}, T_C^{out}\right) \\ Cost_{DP} = C_{DP}(A) \\ 1 \le A_i \le 10 \end{bmatrix} \veebar \begin{bmatrix} Y^{i}_{Multiple\_Pipe} \\ A_i = f\left(T_H^{in}, T_H^{out}, T_C^{in}, T_C^{out}\right) \\ Cost_{MP} = C_{MP}(A) \\ 10 \le A_i \le 100 \end{bmatrix} \veebar \begin{bmatrix} Y^{i}_{Floating\ Head} \\ A_i = f\left(T_H^{in}, T_H^{out}, T_C^{in}, T_C^{out}\right) \\ Cost_{FH} = C_{FH}(A) \\ 10 \le A_i \le 1000 \end{bmatrix}$$

$$i = \{Heater,\ Heat\_Exchanger(HE), Condenser_{Column1}, Condenser_{Column2}\}$$

$$\begin{bmatrix} Y_{Air\ Cooler} \\ A_{Cooler} = A\left(T_{in}^{s}, T_{out}^{s}, T_{in}^{Air}, T_{out}^{Air}, U_{Air}\right) \\ Cost_{Cooler} = C_{Air}(A_{Cooler}) \\ Cost_{Utility\ Cooler} = 0 \end{bmatrix} \veebar \begin{bmatrix} Y_{Cooler\ MultiplePipe} \\ A_{Cooler} = A\left(T_{in}^{s}, T_{out}^{s}, T_{in}^{W}, T_{out}^{W}, U_{W}\right) \\ Cost_{Cooler} = C_{TS}(A_{Cooler}) \\ Cost_{Utility\ Cooler} = Cost\_Water \end{bmatrix}$$

$$\begin{bmatrix} Y^{j}_{Kettle} \\ A_j = A_{Kettle}\left(T_{in}^{s}, T_{out}^{s}, T_{in}^{V}, T_{out}^{R}, U_{Air}\right) \\ Cost_j = C_{Kettle}(A) \\ Cost\ utylity_j = Cost Vapor \end{bmatrix} \veebar \begin{bmatrix} Y^{j}_{Floating\ Head} \\ A_j = A_{FH}\left(T_{in}^{s}, T_{out}^{s}, T_{in}^{V}, T_{out}^{R}, U_{W}\right) \\ Cost_j = C_{FH}(A) \\ Cost\ utylity_j = Cost\_Vapor \end{bmatrix}$$

$$j = \{Reboiler\ Column\ 1,\ Reboiler\ Column\ 2\}$$

$$\begin{bmatrix} Y^{k}_{Tray\ Column} \\ [H^k, D^k] = Size_{Vessel}(N, L, V, \rho ...) \\ Cost^{k}_{Vessel} = Cost_{Vessel}(H^k, D^k) \\ Cost^{k}_{Internals} = Cost_{Internals}(N^k, D^k) \\ D \ge 1m \end{bmatrix} \veebar \begin{bmatrix} Y^{k}_{Packed\ Column} \\ [H^k, D^k] = Size_{Vessel}(N, L, V, \rho ...) \\ Cost^{k}_{Vessel} = Cost_{Vessel}(H^k, D^k) \\ Cost_{Internals} = Cost_{Internals}(H^k, D^k, material) \\ D \le 1m \end{bmatrix}$$

$$k = \{Column\,1,\ Column\,2\}$$

(10.D)

The above model was, like in example 2, converted in a MINLP using a big M formulation for all the implicit blocks and a convex hull formulation for the linear equations that could appear inside the disjunctions.

Table 9 and 10 show the most significant results and some statistics related with the model. The most remarkable aspect is related with the small relaxation GAP only around 5.9% (0.942 10^6

\$/year in the initial relaxed NLP problem 1.002 10^6 \$/year in the best obtained solution), like in example 2, the simultaneous convergence of blocks of equations produce a better relaxation than it could be expected if each one of the equations was individually relaxed.

Conclusions and final Remarks

This paper has introduced a methodology for solving disjunctive programming problems in which most of the equations are given by blocks of equations with an input-output structure (implicit blocks of equations). It has been specialized to the optimization process flowsheets, using commercial simulators in which the sizing and cost functions are given by discontinuous relations or even the selection of different equipments given in a set of alternatives. However, the methodology is not constrained to this kind of systems and it can be directly applied to any system in where some relations are given in form of implicit blocks of equations.

Three different algorithms have bee studied and adapted using an MINLP reformulation of the original Disjunctive problem: BB, OA and LP/NLP-BB. The bottleneck of all the procedure is in the time spent by NLP solvers, that is directly related with two aspects, the time consumed in the communication between different programs (process simulator and external solver) and the time consumed in estimating accurate derivatives. However, the entire methodology can be eventually integrated in a process simulator and then derivatives estimated during the convergence and then all the procedure considerably speeded up. Note that the total number of major iterations performed by the NLP solvers does not increase (typically 10-20 major iterations).

One major contribution of this paper is showing that is possible to use process simulators in rigorous optimization involving discrete decisions, with all the advantages of using rigorous models (when these are needed) instead of the shortcut models that are usually used, and at almost no extra cost.

Two interesting results that are of great interest and that could extend the use of implicit models, are as follows:

- The size of the master problem using implicit equations (in the decomposition algorithms) is reduced in comparison with an equation oriented approach. The reason is that the Master problem is written in terms of independent variables. If the Master is a bottleneck, then merging blocks of equations following the physical meaning of the system (i.e. joining all the equations defining a unit operation) is a valid form of getting a valid reduced master problem without further mathematical considerations.
- Relaxing blocks of equations, instead of each equation individually, (i.e. by a big M reformulation) seems to produce better relaxation gaps. A possible explanation is that although we relax blocks of equations, those equations continue to be given a feasible solution (mass and energy balances, equilibrium, etc inside the block cannot be

violated) which is not true if we relax individual equations. However, more detailed study of this last point must be carried out.

Acknowledgments

The authors gratefully acknowledge financial support from Spanish "*Ministerio de Ciencia y Tecnología*" under project CTQ2005-05456/PPQ

References

[1] Grossmann, I.E.; Westerberg, A.W. Research Challenges in Process System Engineering. *AIChE J.* 2000; 46(9), 1700-1703.

[2] Stephanopoulos, G.; Westerberg, A. W. Studies in Process Synthesis II. Evolutionary Synthesis of Optimal Process Flowsheets. *Chem. Eng. Sci*. 1996; 31, 195.

[3] Douglas, J. M. *A* Hierarchical Decision Procedure for Process Synthesis. *AIChE. J*. 1985; 31, 353.

[4] Douglas, J. M. *Conceptual Design of Chemical Process*. McGrawHill, New York. 1988

[5] Grossmann, I.E.; Review of Nonlinear and Mixed Integer and Disjunctive Programming Techniques for Process System Engineering. *Optimization and Engineering*. 2002; 3; 227-252.

[6] Duran, M. A.; Grossmann, I. E. Simultaneous Optimization and Heat Integration of Chemical Processes. *AIChE J*. 1986; 32, 123.

[7] Lang, Y. D.; Biegler, L. T.; Grossmann, I. E. Simultaneous Optimization and Heat Integration with Process Simulators. *Comput. Chem. Engng.* 1988; 12, 311.

[8] Grossmann, I.E; Caballero, J.A.; Yeomans, H; Advances in Mathematical Programming for Automated Designs, Integration and Operation of Chemical Processes. Proceedings ot the International Conference on Process Integration (PI'99), Copenhagen, Denmark, (1999).

[9] Kravanja, Z. and Grossmann, I.E. "PROSYN: An MINLP Process Synthesizer," *Computers and Chemical Engineering,* 1990; 14, 1363.

[10] Kravanja Z. and I.E. Grossmann, "New developments and capabilities in PROSYN-an automated topology and parameter process synthesizer", *Computers Chem. Engng*.,1994; 18, 1097-1114.

[11] Kravanja, Z.; Grossmann, I. E., A Computational Approach for the Modeling/Decomposition Strategy in the MINLP Optimization of Process Flowsheets with Implicit Models. *Ind. Eng. Chem. Res*.1996; 35(6); 2065-2070.

[12] Caballero, J.A.; Grossmann, I.E.; Aggregated Models for Integrated Distillation Systems. *Ind. Eng. Chem Res*. 1999; 38(6). 2330-2344.

[13] Diwekar, U.M.; Grossmann, I.E.; Rubin, E.S.; A MINLP Process Synthesizer for Sequential Modular Simulator. *Ind. Eng. Chem. Res*. 1992; 31, 313-322.

[14] Reneaume, J.M.F.; Koehret, B.M.; Joulia, X.L.; Optimal Process Synthesis in a Modular Simulator Environment: New Formulation on the Mixed-Integer Nonlinear Programming Problem. *Ind. Eng. Chem. Res*. 1995; 34, 4378-4394.

[15] Diaz, M.S.; Bandoni, J.A. A Mixed Integer Optimization Strategy for a Large Scale Chemical Plant in Operation. *Computers Chem. Engng*. 1996; 20(5), 531-545.

[16] Lang, Y-D.; Biegler, L.T. 2002. Distributed Stream Method for Tray Optimization. *AIChE J*. 2002; *48*(3), 582-595.

[17] Caballero, J.A.; Milán-Yañez, D; Grossmann, I.E. Rigorous Design of Distillation Columns. Integration of Disjunctive Programming and Process Simulators. *Ind. Eng. Chem. Res*. 2005; 44, 6760-6775.

[18] Biegler, L.T.; Hughes, R.R. Infeasible Path Optimization with Sequential Modular Simulators. *AIChE J.* 1982; *28*(6), 994-1002.

[19] Biegler, L.T., "Simultaneous Modular Simulation and Optimization," in Foundations of Computer Aided Process Design-II, pp. 369-409, Westerberg and Chien (eds.), CACHE Committee. 1984.

[20] Biegler, L.T. and R.R. Hughes, "Feasible Path Optimization for Sequential Modular Simulators," *Computers Chem Engng*. 1985; 9, 4, p. 379.

[21] Biegler, L.T., "Improved Infeasible Path Optimization for Sequential Modular Simulators, Part I: The Interface," *Computers Chem. Engng*. 1985; 9, 3, p. 245.

[22] Biegler, L.T. and J.E. Cuthrell, "Improved Infeasible Path Optimization for Sequential Modular Simulators, Part II: The Optimization Algorithm," *Computers Chem. Engng*. 1985; 9, 3, p. 257.

[23] Lang, Y-D.; Biegler, L.T. A Unified Algorithm for Flowsheet Optimization. *Comput. Chem. Engng*. 1987; *11*(2) 143-158.

[24] Alkaya, D.; Vasantharajan, S.; Biegler, L.T.; Generalization of a Tailored Approach for Process Optimization. *Ind. Eng. Chem. Res*. 2000; 39, 1731-1742.

[25] HYSYS v 3.2; Hyprotech Ltd. 1995-2002.

[26] Duran, M.A. and Grossmann, I.E.; An Outer Approximation Algorithm for a Class of Mixed-Integer Nonlinear Programs. *Math. Programming*. 1986; 36, 307.

[27] Quesada, I.; Grossmann, I.E.; An LP/NLP Based Branch and Bound Algorithm for convex MINLP Optimization Problems. *Comput. Chem. Engng*. 1992; 16, 937-947.

[28] Gupta, O.K.; and Ravindran,V.; Branch and Bound Experiments in Convex Nonlinear Integer Programming. *Management Science*. 1985; 31(12), 1533-1546.

[29] Lee, S. and Grossmann, I.E. New Algorithms for Nonlinear Generalized Disjunctive Programming. *Comput. Chem. Eng*. 2000; 24, 2125-2141.

[30] Kocis. G.R. and Grossmann, I.E.; Relaxation Strategy for the Structural Optimization of Process Flowsheets. *Ind. Eng. Chem. Res*. 1987; 26, 1869.

[31] Viswanathan, J. and Grossmann, I.E.; A Combined Penalty Function and Outer Approximation Method for MINLP Optimization. *Comput. Chem. Engng*. 1990; 14, 769.

[32] Drud, A.S. CONOPT: A System for Large Scale Nonlinear Optimization, Reference Manual for CONOPT. Subroutine Library, 69p, ARKI Consulting and Development A/S, Bagsvaerd, Denmark, 1996.

[33] Gill, W. P.E.; Murray, W; Saunders, SNOPT: An SQP algorithm for large-scale constrained optimization, *SIAM J. Optim*. 2002; 12 pp. 979–1006.

[34] MATLAB The language of Technical Computing. The Mathworks INC. 2006.

[35] Sawaya, N.W.; Grossmann, I.E. Computational implementation of non-linear convex hull reformulation. Submitted to publication, *Computers, Chem Engng*. 2005.

[36] Holmström k. The Tomlab Optimization Environment in Matlab. *Advanced Modeling and Optimization*. 1999; 1, 47-69..

[37] Curtis, A.R.; Powell, M.J.D.; Reid, J.K. On the estimation of sparse Jacobian Matrices. *J. Inst. Maths. Applics*. 1974; 13, 117-120.

[38] Turkay, M.; Grossmann, I.E.; Disjunctive Optimization Techniques for the Optimization of Process Systems with Discontinuous Investment Cost. Multiple Size Regions. *Ind. Eng. Chem. Res*. 1996; 35, 2611-2623.

[39] Seider, W.D.; Seader, J.D.; Lewin, D.R.; Process Design Principles. Synthesis Analysis and Evaluation. John Willey & Sons, Inc. 1999.

[40] Karuppiah, R.; Grossmann, I.E.; Global Optimization of Multiscenario Mixed Integer Nonlinear Programming Models Arising in the Synthesis of Integrated Water Networks under Uncertainty. 16th European Symposium on Computer Aided Engineering. W. Marquardt, C. Pantelides Eds. In Computer Aided Chemical Engineering, 21B. 1747-1752. 2006.

Table 1. Data for example 1.

Stream	Composition	Flow	T in	T out	Cost
		(kmol / h)	(K)	(K)	($/kW-year)
Hot	DiPhenylC3	120	500	340	
Cold	Glycerol	100	350	560	
Cooling Utility	Water		323	363	20
Heating Utility	Steam		557	557	80

Heat Exchangers	
Name	U (W / m^2 K)
E-101	500
E-102	1500
E-103	1000

Nominal Pressure in all streams = 1 atm

Thermodynamics: Extended-NRLT (liquid); ideal (vapor)

Table 2: Computational results for the solution of example 1.

Algorithm	**Branch and Bound**	**Outer Approximation**	**LP/NLP-BB**
Best Objective ($/year)	150322	150322	150322
Objective in initial relaxed NLP	49259	49259	49259
Total NLP nodes or NLP sub-problems	37	4	3
CPU time (s)	49.2	5.18	4.01
Solver(s)	SNOPT7	SNOPT7/CPLEX	SNOPT7 + proprietary BB.

Table 3. Results for example 1.

	Area (m^2)	Investment Cost ($/year)
Heat Exchanger (E-101)	25.0	25347
Heater (E-102)	21.1	24339
Cooler (E-103)	28.9	51018
	Power (kW)	**Cost ($/year)**
Heat utility (Steam)	374.3	29940
Cold Utility (Water)	983.8	19676
	Total annualized cost	150322
	Temperature (ºC)	
Hot_Stream_T1	150.0	
Cold_Stream_T2	220.2	

Table 4. Utility data for examples 2 and 3.

Cooling	**Tin (°C)**	**Tout (°C)**	**ΔTmin (°C)**	**U (W/m^2°C)**	**Cost ($/kW·year)**
Air	30	35	10	100	0
Water	20	25	10	800	6.7
Cold Water	8	15	5	800	15.0
R1	-25	-24	3	300	86.3
R2	-40	-39	3	300	106.1
R3	-65	-64	3	300	185.3
Heating					
Hot Water	80	60	10	800	15.0
Vapor LP	125	124	10	1600	59.9
Vapor MP	175	174	10	1600	69.4
Vapor HP	250	249	10	1600	78.8
Other					
Electricity					480.0
Fuel					115.2

Table 5. Data for example 2, except data related with utilities that are reported in Table 4.

Feed Stream	**Composition** (molar fraction)	
	Nitrogen	0.0211
	Methane	0.8276
	Ethane	0.0871
	Propane	0.0411
	n-Butane	0.0141
	n-Pentane	0.0057
	n-Hexane	0.0033
	Pressure	1000 kPa
	Temperature	20 ºC
	Flow	5000 kgmole / h
Product	**Composition**: Combined molar fractionf n_Butane and lighters $\geq$ 0.995	
	Flow $\geq$ 4900 kgmole/h	
	Temperature $\geq$ 20ºC	
Thermodynamics: Peng Robinson Equation of State		

Table 6. Results of example 2

Equipment	**Size Parameters**	**Utility Name / Power (kW)**	**Investment Cost ($)**	**Utilities Cost ($ / year)**
Compressor Engine	----	Fuel oil / 1842	421381	212176
Compressor	----	----	3222085	----
Cooler 1	Area = 156.4 m^2	Cold Water / 2322	61899	34823
Cooler 2	Area = 235.8 m^2	R1 / 1741	80924	150338

Flash 1	Diameter = 2.10 m
	Height = 6.30 m
		----	161253	----

Flash 2	Diameter = 0.66 m
	Height = 1.98 m
		----	27274	----

Heater 1	Area = 45.6 m^2	Hot Water / 1819	31028	27289
Heater 2	Area 0 1.72 m^2	Hot Water / 65.4	6351	980
Column: Condenser	Area = 8.78 m^2	R3 / 87.7	11467	16247
Column: Reboiler	Area = 6.17 m^2	MP Steam / 231.5	9934	16060

Column Shell	Diameter = 0.21 m
	Height = 7.30 m
		----	58713	----

Column Internals	Random polyethylene	----	2495	----
		Sub-total	4094805	4979155
		Total annual cost* =	1170472 $/year	

* Note that the total annualized cost is calculated as $TAC = i\,(i+1)^{PL} \big/ ((i+1)^{PL} - 1) \cdot Investment\_Cost + Utilities\,Cost$

Table 6. (Cont). Result of example 2. Flowsheet Independent variables and final products

Independent variables in Flowsheet		
Name	**Parameter (units)**	**Value**
PS0	Pressure Stream S0 (kPa)	1500
TS1	Temperature Stream S1 (ºC)	15.0
TS2	Temperature Stream S2 (ºC)	-10.7
TS8	Temperature Stream S8 (ºC)	21.0
TS5	Temperature Stream S5 (ºC)	28.9
REC1	Butane Recovery (%)	90.01
REC2	n-Pentane Recovery (%)	99.44
Final Product (GAS Stream)		
Flow (kgmol/h)	4972.98	
Temperature (ºC)	20.02	
Composition (molar fraction)	Nitrogen = 0.021215	
	Methane = 0.832096	
	Ethane = 0.087573	
	Propane = 0.041214	
	n-Butane = 0.012902	
	n-Pentane = 0.003849	
	n-Hexane = 0.001151	

Table 7. Computational results for the solution of example 2.

Algorithm	**Outer Approximation**	**LP/NLP-BB**
Best Objective ($/year)	1170472 $/year	1170472 $/year
Objective in initial relaxed NLP	1015334 $/year	1015334 $/year
Total NLP sub-problems	3	2 (23 LP nodes)
CPU time (s)	708	300
Solvers	SNOPT/CPLEX	SNOPT/ proprietary BB
Explicit Linear equations[1]	12	
Explicit Non-linear equations[1]	22	
Binary variables	16	
Independent variables (flowsheet level) [1]	7	
Other explicit variables [1]	12	
Implicit blocks of equations excluding flowsheet[1]	40	

[1] The number of equations and variables make reference to the initial MINLP problem formulation, in Master problems the number of variables and constraints change in each iteration

Table 8. Data for example 3, except data related with utilities that are reported in Table 4.

	Composition (molar fraction)	
Feed Stream	Methanol	0.8
	Water	0.2
	Pressure	101.3 kPa
	Temperature	25ºC
	Flow	261.5 kgmole / h
DME Stream	Molar fraction (DME) $\geq$ 0.995	
Water Stream	Molar fraction (Water) $\geq$ 0.995	
Thermodynamics: Liquid UNIQUAC; Vapor Ideal.		

Table 9. Results of example 3.

Equipment	Investment Cost ($)	Utility Cost ($/year)	Power / Duty (kW)	Other
Pump 1	18398	3045.2	6.344	Centrifuge
Pump 2	15251	1439.4	2.999	Centrifuge
Heater	15818	413260	4549	Multi-pipe HP Steam Area = 16.6 m^2
Heat exchanger	16590	----	----	Multi-pipe Area = 18.2 m^2
Cooler	139829	0.000	4668	Air Cooler Area = 374.7 m^2

Reactor	118295	0.000	0 (adiabatic)	Diam. = 0.72 m
				Length = 10 m

Column 1				
Condenser	32248	6325	944	Tubes and Shell Floating Head Water Area = 46.4 m^2
Reboiler	58048	120360	1735	Kettle Reboiler HP Steam Area = 54.7 m^2

Vessel	104270	----	----	D = 0.93 m
				H = 8.4 m

Internals	5374	----	----	Packed
Column 2				
Condenser	28263	17481	2609	Tubes and Shell Floating Head Area = 37.8 m^2
Reboiler	12543	280690	3090	Tubes and Shell HP Steam Area = 10.78 m^2

Vessel	262738	----	----	D = 2.9 m
				H = 11.0 m

Internals	23710	----	----	Tray 21 Trays
Total cost	1002202 $/year			

Table 9. (Cont). Result of example 3. Flowsheet Independent variables and final products

Independent variables in Flowsheet		
Name	**Parameter (units)**	**Value**
TS4	Temperature Stream S4 (ºC)	100.00
TS5	Temperature Stream S5 (ºC)	246.25
PS10	Pressure Stream S10 (kPa)	469.2
RECL1	Recovery DME Column 1 (%)	99.633
RECH1	Recovery MeOH Column 1 (%)	98.995
RECL2	Recovery MeOH Column 2 (%)	99.507
RECH2	Recovery Water Column 2 (%)	98.087
TS2	Temperature Stream S2 (ºC)	108.68
XS2DME	Molar fraction DME Stream S2	0.0039
XS2MeOH	Molar fraction MeOH Stream S2	0.9774
XS2Water	Molar fraction Water Stream S2	0.0187
FS2	Molar Flow Stream S2 (kgmole/h)	130.28
Final Product (DME Stream)		
Flow (kgmol/h)	135.3	
Temperature (ºC)	47.14	

Composition (molar fraction)	DME = 0.995
	MeOH = 0.005
	Water = 0.000	

Final Byproduct (Water Stream)		
Flow (kgmol/h)	126.19	
Temperature (ºC)	148.5	

Composition (molar fraction)	DME = 0.000
	MeOH = 0.005
	Water = 0.995	

Table 10. Computational results for the solution of example 2.

Algorithm	**Outer Approximation**	**LP/NLP-BB**
Best Objective ($/year)	$1.002\ 10^6$	$1.002\ 10^6$
Objective in initial relaxed NLP	$0.945\ 10^6$	$0.945\ 10^6$
Total NLP sub-problems	4	9 (111 LP nodes)
CPU time (s)	2219	2423
Solvers	SNOPT/CPLEX	SNOPT/ proprietary BB
Explicit Linear equations[1]	12	
Explicit Non-linear equations[1]	63	
Binary variables	22	
Independent variables (flowsheet level) [1]	12	
Other explicit variables [1]	38	
Implicit blocks of equations excluding flowsheet[1]	39	

[1] The number of equations and variables make reference to the initial MINLP problem formulation, in Master problems the number of variables and constraints change in each iteration

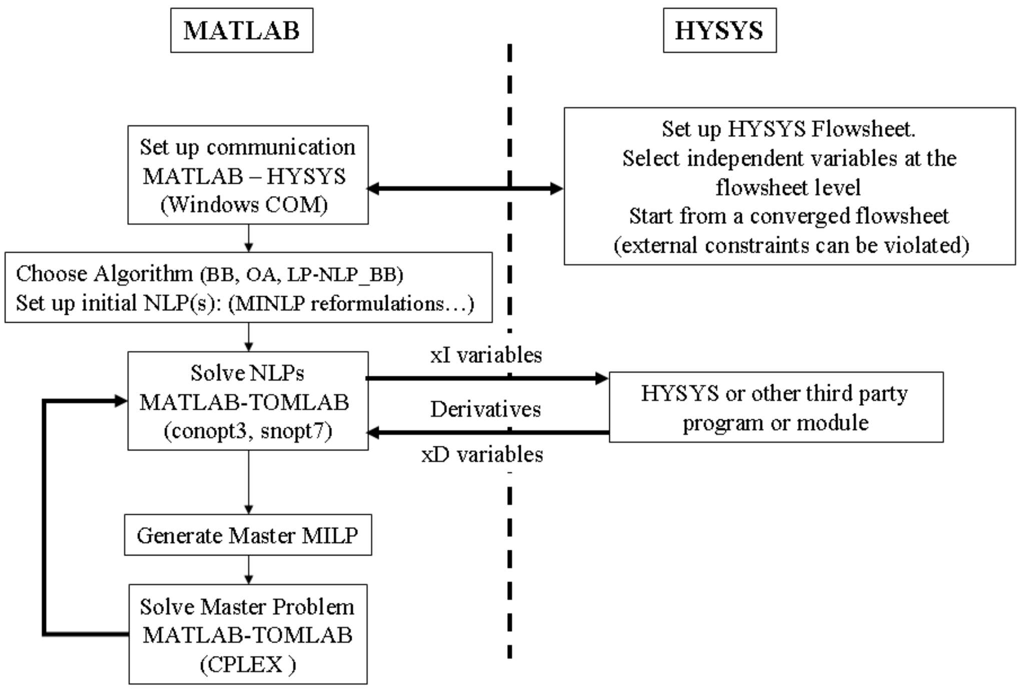
MATLAB
HYSYS
Set up communication MATLAB – HYSYS (Windows COM)
Set up HYSYS Flowsheet. Select independent variables at the flowsheet level Start from a converged flowsheet (external constraints can be violated)
Choose Algorithm (BB, OA, LP-NLP_BB) Set up initial NLP(s): (MINLP reformulations…)
Solve NLPs MATLAB-TOMLAB (conopt3, snopt7)
xI variables
Derivatives
xD variables
HYSYS or other third party program or module
Generate Master MILP
Solve Master Problem MATLAB-TOMLAB (CPLEX)

Figure 1. Scheme of the actual implementation of the algorithms using Matlab- Hysys.

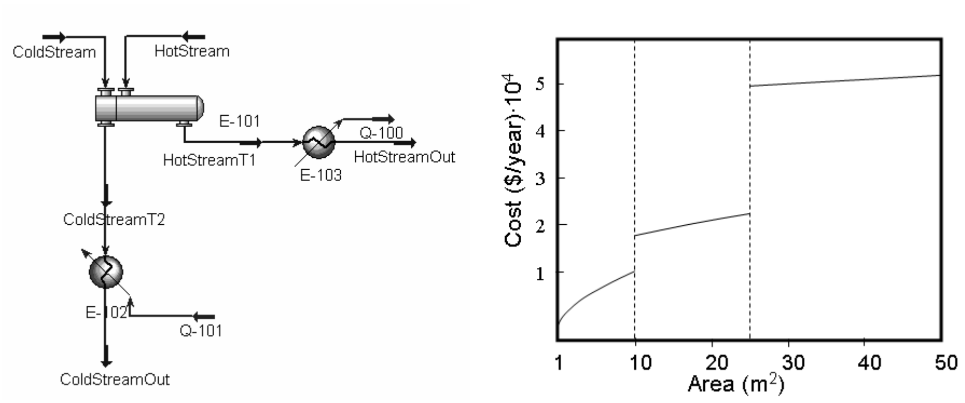

Figure 2. Flowsheet for example 1. The graphic shows the cost regions considered for the example in terms of the heat exchanger area.

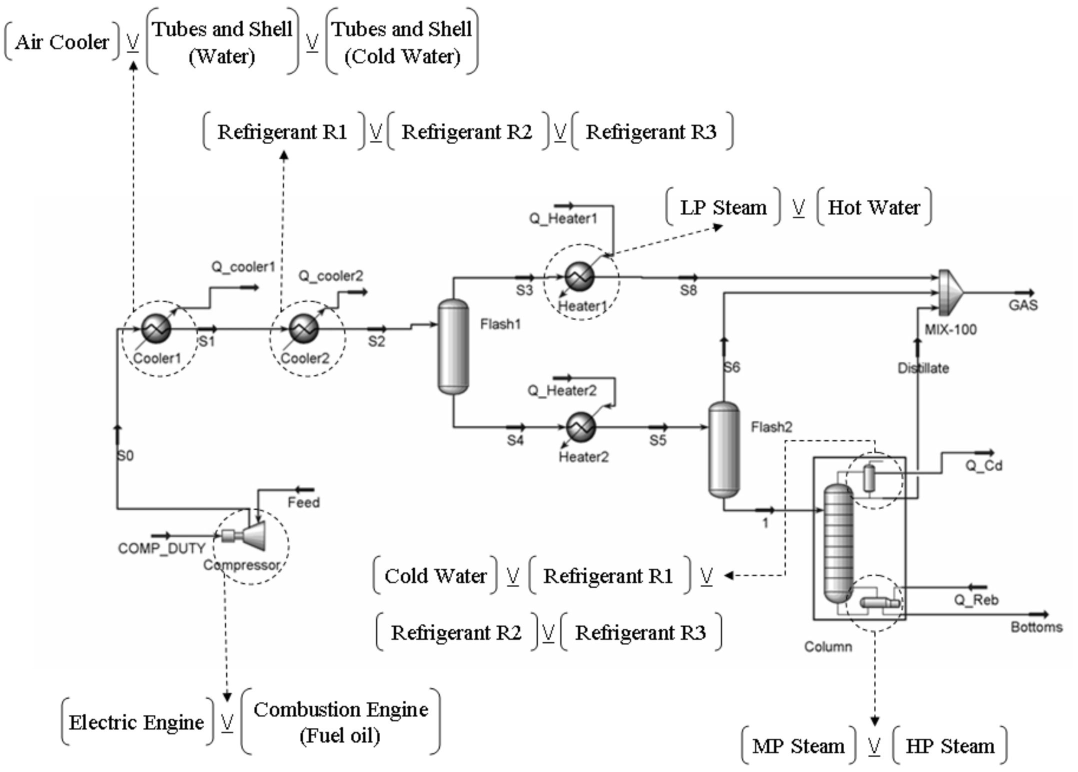

Figure 3. Flowsheet and alternatives for example 2.

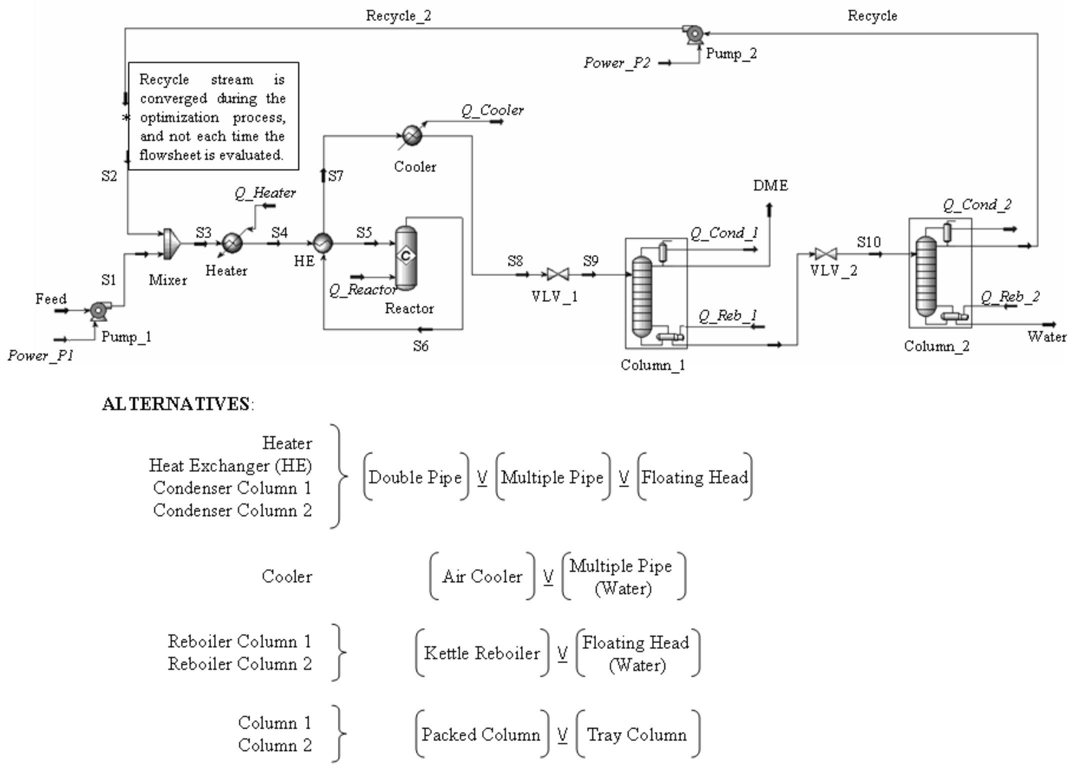
Recycle stream is converged during the optimization process, and not each time the flowsheet is evaluated.

Figure 4. Flowsheet and alternatives for example 3.