Flowsheet Optimization with Complex Cost and Size Functions Using Process Simulators

José A. Caballero and Andrew Odjo
Dept. of Chemical Engineering, University of Alicante, E-03080 Alicante, Spain

Ignacio E. Grossmann
Dept. of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213

DOI 10.1002/aic.11262
Published online August 1, 2007 in Wiley InterScience (www.interscience.wiley.com).

In this article we address the design and optimization 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 the 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 a 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. © 2007 American Institute of Chemical Engineers *AIChE J,* 53: 2351–2366, 2007
Keywords: MINLP, process simulators, process synthesis, process design, disjunctive programming

Introduction

GDP or MINLP techniques have shown to be powerful in the synthesis and design of subsystems such as heat exchanger networks, mass exchanger networks, distillation sequencing, and utility systems.[1–3] However, these techniques are limited to moderately sized 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 nonconvex equations that can prevent not only in finding the optimal solution but even in finding a feasible point. The advances in the development of algorithms for MI(N)LP and GDP,[4] global optimization,[5] software, and hardware have significantly increased the size of the problems that can be solved and have produced computational tools that allow the synthesis of chemical process plants using mathematical programming tools.[6–8] However, a rigorous modeling approach requires further advances in fields like global optimization and GDP algorithms. In order to mitigate those problems, it is common to use shortcut or aggregated models of moderate size and numerical complexity. Also in some cases specially tailored algorithms have been developed to solve some specific problems.[9,10] 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 espe-

Correspondence concerning this article should be addressed to J. A. Caballero at caballer@ua.es.

cially tailored for each one of the unit operations together with large databases of physicochemical thermodynamic and transport properties. Process simulators are also robust and reliable tools that are extensively used in process engineering.

Although nowadays most process simulators have optimization capabilities, they are restricted to deal only with 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 because of their complex discontinuous nature.

There have been several papers that have reported the implementation of MINLP techniques in process simulation.[11–13] These, however, have been limited to the applications of the outer approximation algorithm[14] for MINLP formulations, and therefore, have not involved generalized disjunctive programming formulations as well as other MINLP algorithms. Furthermore, they have not addressed the problems with discontinuous cost functions. The use of implicit equations in a disjunctive environment for MIDO problems was addressed by Oldenburg et al.[15] using a tailored version of the logic base outer approximation algorithm.

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—is not addressed here and will be reported in a future paper.

GDP Formulation with Implicit Models: Application to a Modular Process Simulator

It is convenient to classify the different types of variables that arise in an optimization problem in which there are implicit equations:

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)*. These are variables calculated by the simulator. 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"—that 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). These refer 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. The typical example is the number of trays in a distillation column. However, these variables can change from one iteration to another. The way in which these variables are modified depends on the algorithm used to solve the problem. Owing 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 Caballero et al.[16]

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 the equations that are 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 problem with the implicit equations when using gradient-based optimization methods is that they can create points in which some of these equations are nondifferentiable. 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 only independent variables exact derivatives can be obtained. When they involve dependent variables calculated by the simulator, finite differences of any other approximation method must be used. Based on the earlier definitions and classifications, the disjunctive formulation of the problem can be written as follows:

$$
\begin{aligned}
\min:\ & (x_{\mathrm{I}}, x_{\mathrm{D}}, u, z) \\
\text{s.t.}\ & r_{\mathrm{I}}(x_{\mathrm{I}}, x_{\mathrm{D}}, u) = 0 \\
& s_{\mathrm{E}}(x_{\mathrm{I}}, x_{\mathrm{D}}, u, z) = 0 \\
& s_{\mathrm{E}}(x_{\mathrm{I}}, x_{\mathrm{D}}, u, z) \le 0 \\
& \bigvee_{i \in D_j} \begin{bmatrix} Y_{i,j} \\ h_{\mathrm{I}_i}(x_{\mathrm{I}}, x_{\mathrm{D}}, u) = 0 \\ h_{\mathrm{E}_i}(x_{\mathrm{I}}, x_{\mathrm{D}}, u, z) = 0 \\ g_{\mathrm{E}_i}(x_{\mathrm{I}}, x_{\mathrm{D}}, u, z) \le 0 \end{bmatrix} \quad \forall j \in J \\
& \Omega(Y) = \text{True} \\
& x_{\mathrm{I}} \in X \subseteq \Re^n \\
& Y \in \{\text{True}, \text{False}\}^m
\end{aligned} \tag{1}
$$

In Eq. 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 Eq. 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 nonlinear 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[17–19] it is worth mentioning some relevant points. In process simulators the equations are grouped in modules according to the physical process they represent; these modules are then solved sequentially, usually in the same way material flows through the process (HYSYS.Plant©[20] 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 convergence techniques that are slow. Moreover, 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.[21] 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 rewritten as a MINLP using a big M or a convex hull reformulation.[4] If a pure branch and bound algorithm is used, then no special provision is needed. However, if a decomposition algorithm, like outer approximation[14] or the LP-NLP based branch and bound[22] (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 identify 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:

$$\begin{bmatrix} Y_{\text{unit}} \\ \text{Unit equations} \end{bmatrix} \underline{\vee} \begin{bmatrix} \neg\, Y_{\text{unit}} \\ \text{Relevant variables} = 0 \end{bmatrix} \tag{2}$$

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, then 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 because of 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. An effective way of avoiding this problem consists of removing those units from the NLPs with fixed topology (logic-based approach). In algorithms that use relaxed NLPs, instead of forcing a zero value, these variables can be set to a small tolerance (large enough to avoid the numerical problems and small enough for not affecting the final result). As a second point, if the implicit equations make reference to blocks that will not produce numerical problems, then no special provision is needed.

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)[23,24] outer approximation (OA) with MINLP reformulation[14,25,26] LP-NLP based branch and bound (LP/NLP-BB).[22]

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.[4] 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:

$$\begin{aligned}
\text{min: } & (x_{\text{I}}, x_{\text{D}}(x_{\text{I}}, u), u, z) \\
\text{s.t. } & r_{\text{I}}(x_{\text{I}}, x_{\text{D}}(x_{\text{I}}, u), u) = 0 && \text{(a)} \\
& r_{\text{E}}(x_{\text{I}}, x_{\text{D}}(x_{\text{I}}, u), u, z) = 0 && \text{(b)} \\
& s_{\text{E}}(x_{\text{I}}, x_{\text{D}}(x_{\text{I}}, u), u, z) \le 0 && \text{(c)} \\
& \left.\begin{aligned}
& h^*_{\text{E}_i}(x_{\text{I}}, x_{\text{D}}(x_{\text{I}}, u), u, z, y_{i,j}) = 0 && \text{(d)} \\
& g^*_{\text{E}_i}(x_{\text{I}}, x_{\text{D}}(x_{\text{I}}, u), u, z, y_{i,j}) \le 0 && \text{(e)} \\
& -M(1 - y_{i,j}) \le h^*_{\text{I}_i}(x_{\text{I}}, x_{\text{D}}(x_{\text{I}}, u), u, y_{i,j}) \le M(1 - y_{i,j}) && \text{(f)} \\
& Ay - b \le 0 \\
& 0 \le y_k \le 1 \quad k \in \text{NFB}_{i,j} \\
& y_k = \{0, 1\} \quad k \in \text{FB}_{i,j}
\end{aligned}\right\} \quad i \in D_j; \quad j \in J
\end{aligned} \tag{3}$$

where Eq. 3a refers to the implicit equations (i.e., at the level of process simulators). Usually, these 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 3b and 3c are the rest of explicit equations that do not depend on the disjunction. Equations 3d and 3e are the convex hull, big M or any other valid MINLP reformulation. Equation 3f 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 NFB makes reference to those binary variables that are not fixed in a given iteration and the set FB makes reference to the binary variables fixed in a given iteration. (NFB $\cup$ FB = B) and B is the set of all binaries.

The OA and LP/NLP-BB algorithms iterate between two different problems, an NLP like that in Eq. 3 with fixed $y_{i,j}$ and a Master (MILP) problem that is obtained from the linearizations of the constraints and the objective function[26]:

$$
\begin{aligned}
&\min_{\alpha,x_I,s} \quad \alpha + \Pi^T(s_1+s_2+s_3+s_4)\\
&\left.\begin{aligned}
&\text{s.t. } \alpha \geq f\left[x_I^k, x_D(x_I^k,u),u,z^k\right] + \nabla f\left[x_I^k, x_D(x_I^k,u),u,z^k\right]\begin{pmatrix} x_I - x_I^k \\ z - z^k \end{pmatrix}\\
&T_1^T\left\{\nabla r_E\left[x_I^k, x_D(x_I^k,u),u,z^k\right]\begin{pmatrix} x_I - x_I^k \\ z - z^k \end{pmatrix}\right\} \leq s_1\\
&s_E\left[x_I, x_D(x_I,u),u,z^k\right] + \nabla s_E\left[x_I, x_D(x_I,u),u,z^k\right]\begin{pmatrix} x_I - x_I^k \\ z - z^k \end{pmatrix} \leq s_2\\
&T_2^T\left\{\nabla h_E^*\left[x_I^k, x_D(x_I^k,u),u,z^k,y_{i,j}^k\right]\begin{pmatrix} x_I - x_I^k \\ y_{i,j} - y_{i,j}^k \\ z - z^k \end{pmatrix}\right\} \leq s_3\\
&g_E^*\left[x_I^k, x_D(x_I^k,u),u,z^k,y_{i,j}^k\right] + \nabla g_E^*\left[x_I^k, x_D(x_I^k,u),u,z^k,y_{i,j}^k\right]\begin{pmatrix} x_I - x_I^k \\ y_{i,j} - y_{i,j}^k \\ z - z^k \end{pmatrix} \leq s
\end{aligned}\right\}
\begin{array}{c} k = 1\ldots K \\ i \in D_j \\ j \in J \end{array}\\
&Ay - b \leq 0; \quad (s_1,s_2,s_3) \geq 0; \quad y \in \{0,1\}^m
\end{aligned} \tag{4}
$$

where T is a matrix defined 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 slack variables introduced to minimize the effect of nonconvexities, 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 is not defined by dependent variables. Linearizations of dependent variables can be performed applying the chain rule through the entire flowsheet or the implicit equations and, therefore, 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.

$$\nabla_{x_I} f = \frac{\partial f}{\partial x_I} + \frac{\partial f}{\partial x_D}\frac{dx_D}{dx_I} \tag{5}$$

As indicated in a previous section, a critical step is to obtain accurate derivatives. For the explicit equations, either automatic differentiation (AD) or complex arithmetic[27] can be used in both cases, with machine precision accuracy ($\sim 10^{-16}$). In the partial derivatives respect to the variables appearing in implicit blocks at the level of process simulator, it is necessary to perform a sensitivity analysis in order to determine the optimal perturbation value and minimize the noise introduced by the process simulator (a typical value is around 10^{-4}). Finally for those variables appearing in equations of implicit blocks, we have to control a mixed strategy using AD or complex arithmetic whenever possible, or using a perturbation ($h = x_0\sqrt{\text{eps}}$, where x_0 is a typical value of variable x_I). In this last case, all the equations solved inside the implicit blocks must converge with an accuracy of at least two orders of magnitude lower than the perturbation value.

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, 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 prespecify tolerance). Then, a new NLP is solved with the values of the binaries obtained from the last Master problem to provide 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 improvement in the upper bound.[26]

In the LP/NLP-BB algorithm, the idea is to use a branch and cut approach based on the MILP. When an integer solution is found, an NLP with the binaries fixed to the integer solution is solved. The solution of this problem 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 restarting 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 often remains unchanged.[22,28]

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,[29] SNOPT[30]) through an activeX client–server application. All the process is controlled by MATLAB©.[31] 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, when using external solvers.

Figure 1 shows a scheme of the implementation. The first step is at the level of process simulator. Here we have to set up the flowsheet, determine the degrees of freedom, and decide which the independent variables are in the flowsheet, among the available options. In this point, HYSYS has an important advantage over other process simulators because of 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. In this way it is possible to calculate inputs in terms of outputs, or any feasible input–output combination.

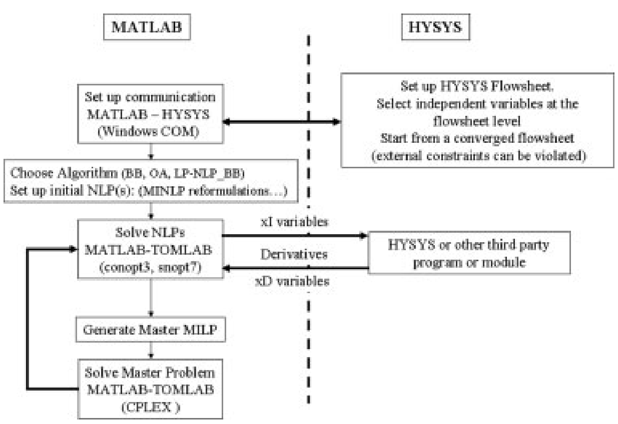

Figure 1. Scheme of the actual implementation of the algorithms using Matlab-Hysys.

Of course, there are exceptions like distillation columns. It is convenient to start with a converged flowsheet, even though the other external constraints were initially violated. We let the optimizer to converge all those constraints at 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). These new constraints can 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 common one. In order to deal with these 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 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, since it leads to tighter relaxations. 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 nonlinear convex hull reformulation[32] it introduces numerical difficulties when some of the binaries take zero values. A correct implementation of the nonlinear convex hull was introduced by Sawaya and Grossmann,[33] 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 given later show a general disjunction, a valid MINLP reformulation:

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

In the equations mentioned earlier, the superscript "L" refers to the subset of independent and/or z variables that are linear inside the disjunction. The superscript "N" refers to

Table 1. Data for Example 1

Stream	Composition	Flow, kmol /h	T_{in}, K	T_{out}, K	Cost, \$/(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).

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 steps 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 steps are controlled from MATLAB, where the different algorithms were implemented. Derivative calculation, master generation, etc. are performed automatically by MATLAB in our current implementation. NLP subproblems and Master problems are solved using TOMLAB-MATLAB[34]—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 largest 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.[35]

Examples

Example 1

The first example is an adaptation of an example by Turkay and Grossmann.[36] It consists of a small heat exchanger 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)

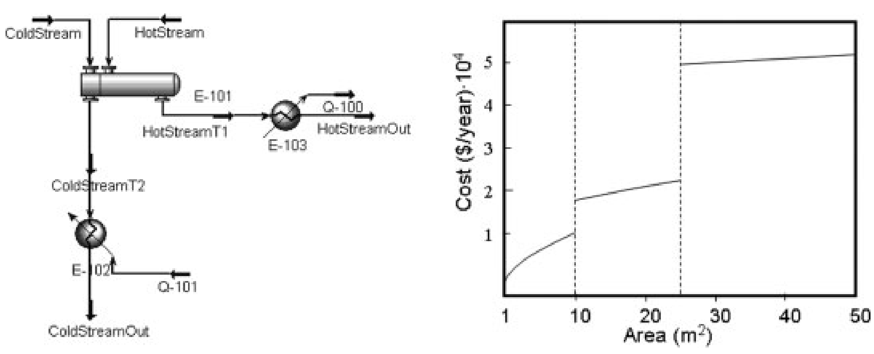

Figure 2. Flowsheet for Example 1.

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

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 perform a degree of freedom analysis at the level of flowsheet. Although in most situations this analysis is not easy, it is in general 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 overspecifications and inconsistencies, which facilitates this stage. In this first small example there is only one degree of freedom, and we selected the heat exchange area in the first heat exchanger to be the independent variable. A disjunctive formulation of this problem is then given as follows:

$$
\begin{aligned}
&\text{min: } \sum_{j\in \mathrm{HE}} \mathrm{IC}_j + \mathrm{Cost}_{\mathrm{steam}} \cdot W_{\mathrm{steam}} + \mathrm{Cost}_{\mathrm{water}} \cdot W_{\mathrm{water}} \\
&\text{s.t.} \\
&\begin{bmatrix} Y_{j,1} \\ \mathrm{IC}_j = 2750\, A_j^{0.6} + 3000 \\ 1 \le A_j \le 10 \end{bmatrix} \underline{\vee} \begin{bmatrix} Y_{j,2} \\ \mathrm{IC}_j = 1500\, A_j^{0.6} + 15,000 \\ 10 \le A_j \le 25 \end{bmatrix} \underline{\vee} \begin{bmatrix} Y_{j,3} \\ \mathrm{IC}_j = 600\, A_j^{0.6} + 46,500 \\ 25 \le A_j \le 50 \end{bmatrix} \quad j \in \mathrm{HE} \\
&T_{\mathrm{HotStream\ T1}} \ge T_{\mathrm{out\_water}} + 10 \\
&\begin{pmatrix} T_{\mathrm{HotStream\ T1}} \\ T_{\mathrm{out\_water}} \\ A_2 \\ A_3 \\ W_{\mathrm{steam}} \\ W_{\mathrm{water}} \end{pmatrix} = \Theta(A_1) \\
&\mathrm{HE} = \{j | j \text{ is a heat exchanger}\}; \quad Y \in \{\text{True, False}\}^9
\end{aligned} \tag{7}
$$

In Eqs. 7, W is the heat load supplied/removed by steam or water (kW). IC is the annualized investment cost of each heat exchanger and $\Theta(A_1)$ 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 Eq. 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 takes 5.2 s and four major iterations, while LP/NLP-BB takes 4 s 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 tends to minimize this effect because the Master approximates faster than with OA in the feasible region. Table 2 gives some statistics of the problem using the different algorithms.

The optimal solution yields a total annual cost of 150,322 \$/year. Table 3 shows a summary of the optimal results. The initial relaxed NLP gave an initial objective function of 49,259 \$/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.[37] 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 126,779 \$/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.

Table 2. Computational Results for the Solution of Example 1

Algorithm	Branch and Bound	Outer Approximation	LP/NLP-BB
Best objective, $/year	150,322	150,322	150,322
Objective in initial relaxed NLP	49,259	49,259	49,259
Total NLP nodes or NLP subproblems	37	4	3
CPU time, s	49.2	5.18	4.01
Solver(s)	SNOPT7	SNOPT7/CPLEX	SNOPT7 + proprietary BB

Example 2

This example is a modification of a problem proposed by Seider et al.[38] and consists of the design of a natural gas plant. It is required to process a natural gas stream at 5000 kmol/h, 20°C, 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 motor 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 0°C and we can choose between three different refrigeration systems (R1, R2, or R3 in Table 4). The stream is flashed in Flash 1 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.

Table 3. Results for Example 1

	Area, m^2	Investment Cost, $/year
Heat exchanger (E-101)	25.0	25,347
Heater (E-102)	21.1	24,339
Cooler (E-103)	28.9	51,018
	Power, kW	Cost, $/year
Heat utility (steam)	374.3	29,940
Cold utility (water)	983.8	19,676
Total annualized cost		*150,322*
	Temperature, °C	
Hot_Stream_T1	150.0	
Cold_Stream_T2	220.2	

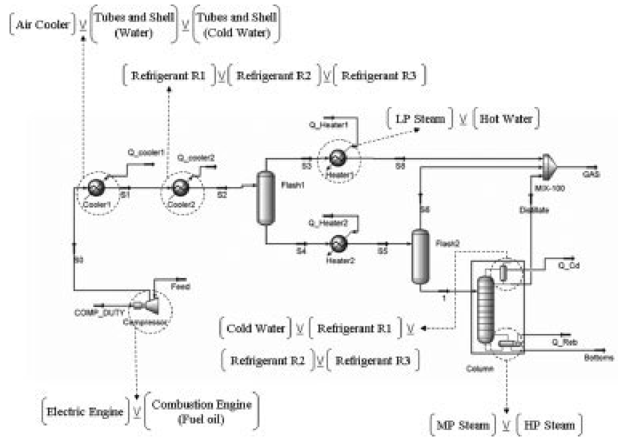

Figure 3. Flowsheet and alternatives for Example 2.

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. Capital cost estimation, in this example and in the next one, was done using the procedure presented in Turton et al.[39] Costs of utilities were obtained from the Distil™ Database.[40]

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 are associated to the basic flow sheet 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 nonconvex 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 to 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 follows:

Table 4. Utility Data for Examples 2 and 3

Cooling	T_{in}, °C	T_{out}, °C	ΔT_{min}, °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

Objective function:

$$\begin{aligned}
&\min:\ \text{TAC}\\
&\text{TAC} = \frac{i(i+1)^{\text{PL}}}{(i+1)^{\text{PL}}-1}\text{InvestmentCost}\\
&\quad + \text{Annual Utilities Cost}\\
&\text{Investment Cost} = \text{Cost_Compressor} + \text{Cost_Engine}\\
&\quad + \text{Cost_Ref1} + \text{Cost_Ref2} + \text{Cost_Heat1} + \text{Cost_Heat2}\\
&\quad + \text{Cost_Flash1} + \text{Cost_Flash2} + \text{Cost_Heat2}\\
&\quad + \text{Cost_Flash1} + \text{Cost_Flash2} + \text{Cost_Cond}\\
&\quad + \text{Cost_Reb} + \text{Cost_Column_Vessel}\\
&\quad + \text{Cost_Column_Internals}\\
&\text{Annual Utilities Cost} = \text{Cost}_{\text{Utility_Eng}} + \text{Cost}_{\text{Utility_Cooler1}}\\
&\quad + \text{Cost}_{\text{Utility_Cooler2}} + \text{Cost}_{\text{Utility_Heater1}} + \text{Cost}_{\text{Utility_Cond}}\\
&\quad + \text{Cost}_{\text{Utility_Reb}}\text{Cost}_{\text{Utility_Compressor}}
\end{aligned} \tag{8.OBJ}$$

External specifications to the problem:

$$\begin{aligned}
&T_{\text{S1}} + 10 \le T_{\text{S0}}\\
&T_{\text{L1}} \le -60\\
&T_{\text{V2}} \le 240\\
&\text{Flow}_{\text{GAS}} \ge 4900\\
&\sum_{i\in I}\text{Gas}_{\text{molar_fraction}\,i} \ge 0.995\\
&T_{\text{GAS}} \ge 20\\
&\quad I = \{n\text{C}_4 \text{ and lighter products}\}
\end{aligned} \tag{8.ESP}$$

where interest rate i = 0.08; plant life PL = 8.

In Eq. 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 come 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:

$$\begin{aligned}
&\text{Cost_Compressor} = C_{\text{comp}}(W\text{_Compressor})\\
&\text{Cost_Flash1} = C_{\text{Flash}}(\text{VFS}_3, \text{VFS}_4, \rho_3, \rho_4)\\
&\text{Cost_Flash2} = C_{\text{Flash}}(\text{VFS}_6, \text{VFS}_7, \rho_1, \rho_1)\\
&\text{Cost_Column_Vessel} = C_{\text{Vessel}}(H, D)\\
&\text{Cost_Column_Internals} = C_{\text{Int}}(D, N_t)
\end{aligned} \tag{8.IMP}$$

Equations calculated by the process simulator:

$$x_{\text{D}} = \Theta(x_{\text{I}}) \tag{8.I}$$

In Eq. 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 seven 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).

Table 5. Data for Example 2, Except Data Related with Utilities That Are Reported in Table 4

	Composition, molar fraction	
Feed stream	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°C
	Flow	5000 kg mol/h
Product	Composition: Combined molar fraction of n-butane and lighters ≥ 0.995	
	Flow ≥ 4900 kgmol/h	
	Temperature ≥ 20°C	
Thermodynamics: Peng Robinson equation of state		

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 constraint in order to ensure that all the variables are inside the valid limits, but in our examples it has not been necessary.

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

$$\begin{bmatrix} Y_{\text{Cooler1\_Air}} \\ \text{Area}_{\text{Cooler1}} = A_{\text{Cooler1\_Air}}(\text{TS0}, \text{TS1}, W_{\text{Ref1}}) \\ \text{Cost}_{\text{Ref1}} = C_{\text{Cooler\_Air}}(\text{Area}_{\text{Cooler1}}) \\ \text{Cost}_{\text{Utility\_Cooler1}} = \text{Cost}_{\text{Air}} \\ \text{TS0} \geq 45°\text{C}; \text{TS1} \geq 40°\text{C} \end{bmatrix} \underline{\vee} \begin{bmatrix} Y_{\text{Cooler1\_Water}} \\ \text{Area}_{\text{Cooler1}} = A_{\text{Cooler1}}(\text{TS0}, \text{TS1}, W_{\text{Cooler1}}) \\ \text{Cost}_{\text{Utility\_Cooler1}} = \text{Cost}_{\text{Water}} \\ \text{TS0} \geq 35°\text{C}; \text{TS1} \geq 30°\text{C} \end{bmatrix} \underline{\vee} \begin{bmatrix} Y_{\text{Cooler1\_ColdWater}} \\ \text{Area}_{\text{Cooler1}} = A_{\text{Cooler1}}(\text{TS0}, \text{TS1}, W_{\text{Cooler1}}) \\ \text{Cost}_{\text{Utility\_Cooler1}} = \text{Cost}_{\text{Cold\_Water}} \\ \text{TS0} \geq 20°\text{C}; \text{TS1} \geq 15°\text{C} \end{bmatrix} \tag{8.D2}$$

$$\begin{bmatrix} Y_{\text{Cooler2\_R1}} \\ \text{Area}_{\text{Cooler2}} = A_{\text{Cooler2}}(\text{TS1}, \text{TS2}, W_{\text{Cooler2}}) \\ \text{Cost}_{\text{Cooler2}} = C_{\text{Cooler2}}(\text{Area}_{\text{Cooler2}}) \\ \text{TS1} \geq -21°\text{C}; \text{TS2} \geq -22°\text{C} \end{bmatrix} \underline{\vee} \begin{bmatrix} Y_{\text{Cooler2\_R2}} \\ \text{Area}_{\text{Cooler2}} = A_{\text{Cooler2}}(\text{TS1}, \text{TS2}, W_{\text{Cooler2}}) \\ \text{Cost}_{\text{Cooler2}} = C_{\text{Cooler2}}(\text{Area}_{\text{Cooler2}}) \\ \text{TS1} \geq -36°\text{C}; \text{TS2} \geq -37°\text{C} \end{bmatrix} \underline{\vee} \begin{bmatrix} Y_{\text{Cooler2\_R3}} \\ \text{Area}_{\text{Cooler2}} = A_{\text{Cooler2}}(\text{TS1}, \text{TS2}, W_{\text{Cooler2}}) \\ \text{Cost}_{\text{Cooler2}} = C_{\text{Cooler2}}(\text{Area}_{\text{Cooler2}}) \\ \text{TS1} \geq -61°\text{C}; \text{TS2} \geq -62°\text{C} \end{bmatrix} \tag{8.D3}$$

$$\begin{bmatrix} Y_{\text{Heat1\_LP}} \\ \text{Area}_{\text{Heat1}} = A_{\text{Heat1}}(\text{TS3}, \text{TS8}, W_{\text{Heat1}}) \\ \text{Cost}_{\text{Heat1}} = C_{\text{Heat1}}(\text{Area}_{\text{Heat1}}) \\ \text{Cost}_{\text{Utility\_Heat1}} = \text{Cost}_{\text{LP}} \end{bmatrix} \underline{\vee} \begin{bmatrix} Y_{\text{Heat1\_HotWater}} \\ \text{Area}_{\text{Heat1}} = A_{\text{Heat1}}(\text{TS3}, \text{TS8}, W_{\text{Heat1}}) \\ \text{Cost}_{\text{Heat1}} = C_{\text{Heat1}}(\text{Area}_{\text{Heat1}}) \\ \text{Cost}_{\text{Utility\_Heat1}} = \text{Cost}_{\text{HotWater}} \end{bmatrix} \tag{8.D4}$$

$$\begin{bmatrix} Y_{\text{CondWater}} \\ \text{Area}_{\text{Cond}} = A_{\text{Cond}}(\text{TV1,TL1}, W_{\text{Cond}}) \\ \text{Cost}_{\text{Cond}} = C_{\text{cond}}(\text{Area}_{\text{Cond}}) \\ \text{Cost}_{\text{UtilityCond}} = \text{Cost}_{\text{Water}} \end{bmatrix} \underline{\vee} \begin{bmatrix} Y_{\text{CondR1}} \\ \text{Area}_{\text{Cond}} = A_{\text{Cond}}(\text{TV1,TL1}, W_{\text{Cond}}) \\ \text{Cost}_{\text{Cond}} = C_{\text{cond}}(\text{Area}_{\text{Cond}}) \\ \text{Cost}_{\text{UtilityCond}} = \text{Cost}_{\text{R1}} \end{bmatrix} \underline{\vee} \begin{bmatrix} Y_{\text{CondR2}} \\ \text{Area}_{\text{Cond}} = A_{\text{Cond}}(\text{TV1,TL1}, W_{\text{Cond}}) \\ \text{Cost}_{\text{Cond}} = C_{\text{cond}}(\text{Area}_{\text{Cond}}) \\ \text{Cost}_{\text{UtilityCond}} = \text{Cost}_{\text{R2}} \end{bmatrix} \underline{\vee} \begin{bmatrix} Y_{\text{CondR3}} \\ \text{Area}_{\text{Cond}} = A_{\text{Cond}}(\text{TV1,TL1}, W_{\text{Cond}}) \\ \text{Cost}_{\text{Cond}} = C_{\text{cond}}(\text{Area}_{\text{Cond}}) \\ \text{Cost}_{\text{UtilityCond}} = \text{Cost}_{\text{R3}} \end{bmatrix} \tag{8.D5}$$

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

Table 6. Results of Example 2

Equipment	Size Parameters	Utility Name/Power, kW	Investment Cost, $	Utilities Cost, $/year
Compressor engine	–	Fuel oil/1842	421,381	212,176
Compressor	–	–	3,222,085	–
Cooler 1	Area = 156.4 m^2	Cold water/2322	61,899	34,823
Cooler 2	Area = 235.8 m^2	R1/1741	80,924	150,338
Flash 1	Diameter = 2.10 m, Height = 6.30 m	–	161,253	–
Flash 2	Diameter = 0.66 m, Height = 1.98 m	–	27,274	–
Heater 1	Area = 45.6 m^2	Hot water/1819	31,028	27,289
Heater 2	Area = 1.72 m^2	Hot water/65.4	6351	980
Column: condenser	Area = 8.78 m^2	R3/87.7	11,467	16,247
Column: reboiler	Area = 6.17 m^2	MP steam/231.5	9934	16,060
Column shell	Diameter = 0.21 m, Height = 7.30 m	–	58,713	–
Column internals	Random polyethylene	–	2495	–
		Subtotal	*4,094,805*	*4,979,155*
		*Total annual cost**	*1,170,472 $/year*	

Independent Variables in Flowsheet

Name	Parameter (units)	Value
PS0	Pressure stream S0, kPa	1500
TS1	Temperature stream S1, °C	15.0
TS2	Temperature stream S2, °C	−10.7
TS8	Temperature stream S8, °C	21.0
TS5	Temperature stream S5, °C	28.9
REC1	Butane recovery, %	90.01
REC2	*n*-Pentane recovery, %	99.44

	Final Product (GAS Stream)
Flow, kgmol/h	4972.98
Temperature, °C	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

*Note that the total annualized cost is calculated as TAC = $i(i+1)E/((i+1)^{PL}-1)$ Investment_Cost + Utilities Cost.

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 indicated in Example 1, the bottleneck in the solution 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 1171 × 10^3 $/year). A similar behavior appears in Example 3, which we will describe in the following 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,

Table 7. Computational Results for the Solution of Example 2

Algorithm	Outer Approximation	LP/NLP-BB
Best Objective, $/year	1,170,472	1,170,472
Objective in initial relaxed NLP, $/year	1,015,334	1,015,334
Total NLP subproblems	3	2 (23 LP nodes)
CPU time, s	708	300
Solvers	SNOPT/ CPLEX	SNOPT/ proprietary BB
Explicit linear equations*	12	
Explicit nonlinear equations*	22	
Binary variables	16	
Independent variables (flowsheet level)*	7	
Other explicit variables*	12	
Implicit blocks of equations excluding flowsheet*	40	

*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	25°C
	Flow	261.5 kgmol/h
DME stream	Molar fraction (DME) 0.995	
Water stream	Molar fraction (Water) 0.995	
Thermodynamics: Liquid UNIQUAC; vapor ideal		

etc. can be violated. However, in our examples, all the mass balances, equilibrium equations, cost and size correlations 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.

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°C and is carried out in an adiabatic reactor. Methanol is introduced to the system at 25°C 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 preheated in the heat exchanger (HE) before entering 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 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 pipe 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 HE, inlet temperature to the reactor; Pressures at the exits 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 eval-

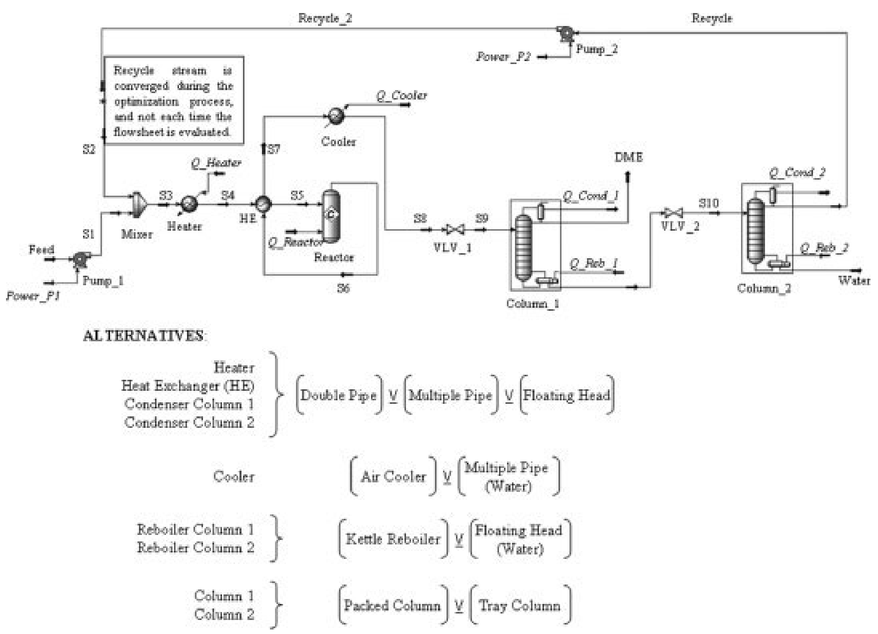

Figure 4. Flowsheet and alternatives for Example 3.

uation 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 of 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 follows:

$$
\begin{aligned}
&\min:\ \text{TAC}\\
&\text{TAC} = \frac{i(i+1)^{\text{PL}}}{(i+1)^{\text{PL}}-1}\ \text{Investment Cost} + \text{Annual Utilities Cost}\\
&\text{Investment Cost} = \text{Cost\_HE} + \text{Cost\_Heater} + \text{Cost\_Cooler}\\
&\quad + \text{Cost\_Pump1} + \text{Cost\_Pump2} + \text{Cost\_Reactor} +\\
&\quad \text{Cost\_Condenser1} + \text{Cost\_Condenser2} + \text{Cost\_Reboiler1}\\
&\quad + \text{Cost\_Reboiler2} + \text{Cost\_Column\_Vessel1}\\
&\quad + \text{Cost\_Column\_Vessel2} +\\
&\quad \text{Cost\_Column\_Internals1} + \text{Cost\_Column\_Internals2}\\
&\text{Annual Utilities Cost} = \text{Cost}_{\text{Utilities\_pump1}} + \text{Cost}_{\text{Utilities\_pump2}}\\
&\quad + \text{Cost}_{\text{Utility\_Heater}} + \text{Cost}_{\text{Utility\_HE}} + \text{Cost}_{\text{Utility\_Condenser1}}\\
&\quad + \text{Cost}_{\text{Utility\_Condenser2}} + \text{Cost}_{\text{Utility\_Reboiler1}} +\\
&\quad \text{Cost}_{\text{Utility\_Reboiler2}} + \text{Cost}_{\text{Utility\_Cooler}}
\end{aligned}
\tag{10.OBJ}
$$

External specifications and other explicit equations:

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

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 Eq. 10.ESP are used to converge the recycle stream.

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

$$
\begin{aligned}
&\text{Cost\_Pump\_1} = C_{\text{Pump1}}(\text{Power\_P1}, \text{P}_{\text{S1}})\\
&\text{Cost\_Pump\_2} = C_{\text{Pump2}}(\text{Power\_P2}, P_{\text{Recycle}})\\
&[\text{Diameter, Height}]_{\text{reactor}} = \text{Size}_i(P_{\text{reactor}})\\
&\text{Cost\_Reactor} = \text{Cost\_Vessel}(\text{Diameter, Height})
\end{aligned}
\tag{10. IMP}
$$

Equations calculated by the process simulator:

$$x_D = \Theta(x_1) \tag{10.I}$$

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

The disjunctions for the discrete decisions are

$$
\begin{bmatrix} Y^i_{\text{Double Pipe}}\\ A_i = f(T^{\text{in}}_{\text{H}}, T^{\text{out}}_{\text{H}}, T^{\text{in}}_{\text{C}}, T^{\text{out}}_{\text{C}})\\ \text{Cost}_{\text{DP}} = C_{\text{DP}}(A)\\ 1 \leq A_i \leq 10 \end{bmatrix} \underline{\vee} \begin{bmatrix} Y^i_{\text{Multiple\_Pipe}}\\ A_i = f(T^{\text{in}}_{\text{H}}, T^{\text{out}}_{\text{H}}, T^{\text{in}}_{\text{C}}, T^{\text{out}}_{\text{C}})\\ \text{Cost}_{\text{MP}} = C_{\text{MP}}(A)\\ 10 \leq A_i \leq 100 \end{bmatrix} \underline{\vee} \begin{bmatrix} Y^i_{\text{Floating Head}}\\ A_i = f(T^{\text{in}}_{\text{H}}, T^{\text{out}}_{\text{H}}, T^{\text{in}}_{\text{C}}, T^{\text{out}}_{\text{C}})\\ \text{Cost}_{\text{FH}} = C_{\text{FH}}(A)\\ 10 \leq A_i \leq 1000 \end{bmatrix}
$$
$$i = \{\text{Heater, Heat\_Exchanger (HE), Condenser}_{\text{Column1}}, \text{Condenser}_{\text{Column2}}\}$$

$$
\begin{bmatrix} Y_{\text{Air Cooler}}\\ A_{\text{Cooler}} = A(T^{\text{s}}_{\text{in}}, T^{\text{s}}_{\text{out}}, T^{\text{Air}}_{\text{in}}, T^{\text{Air}}_{\text{out}}, U_{\text{Air}})\\ \text{Cost}_{\text{Cooler}} = C_{\text{Air}}(A_{\text{Cooler}})\\ \text{Cost}_{\text{Utility Cooler}} = 0 \end{bmatrix} \underline{\vee} \begin{bmatrix} Y_{\text{Cooler MultiplePipe}}\\ A_{\text{Cooler}} = A(T^{\text{s}}_{\text{in}}, T^{\text{s}}_{\text{out}}, T^{\text{W}}_{\text{in}}, T^{\text{w}}_{\text{out}}, U_{\text{W}})\\ \text{Cost}_{\text{Cooler}} = C_{\text{TS}}(A_{\text{Cooler}})\\ \text{Cost}_{\text{Utility Cooler}} = \text{Cost\_Water} \end{bmatrix}
$$

$$
\begin{bmatrix} Y^j_{\text{Kettle}}\\ A_j = A_{\text{Kettle}}(T^{\text{s}}_{\text{in}}, T^{\text{s}}_{\text{out}}, T^{\text{V}}_{\text{in}}, T^{\text{R}}_{\text{out}}, U_{\text{Air}})\\ \text{Cost}_j = C_{\text{Kettle}}(A)\\ \text{Cost utility}_j = \text{Cost Vapor} \end{bmatrix} \underline{\vee} \begin{bmatrix} Y^j_{\text{Floating Head}}\\ A_j = A_{\text{FH}}(T^{\text{s}}_{\text{in}}, T^{\text{s}}_{\text{out}}, T^{\text{V}}_{\text{in}}, T^{\text{R}}_{\text{out}}, U_{\text{W}})\\ \text{Cost}_j = C_{\text{FH}}(A)\\ \text{Cost utility}_j = \text{Cost\_Vapor} \end{bmatrix}
$$
$$j = \{\text{Reboiler Column 1, Reboiler Column 2}\}$$

$$
\begin{bmatrix} Y^k_{\text{Tray Column}}\\ [H^k, D^k] = \text{Size}_{\text{Vessel}}(N, L, V, \rho \ldots)\\ \text{Cost}^k_{\text{Vessel}} = \text{Cost}_{\text{Vessel}}(H^k, D^k)\\ \text{Cost}^k_{\text{Internals}} = \text{Cost}_{\text{Internals}}(N^k, D^k)\\ D \geq 1\text{m} \end{bmatrix} \underline{\vee} \begin{bmatrix} Y^k_{\text{Packed Column}}\\ [H^k, D^k] = \text{Size}_{\text{Vessel}}(N, L, V, \rho \ldots)\\ \text{Cost}^k_{\text{Vessel}} = \text{Cost}_{\text{Vessel}}(H^k, D^k)\\ \text{Cost}_{\text{Internals}} = \text{Cost}_{\text{Internals}}(H^k, D^k, \text{material})\\ D \leq 1\text{m} \end{bmatrix}
\tag{10.D}
$$
$$k = \{\text{Column 1, Column 2}\}$$

Table 9. Results of Example 3

Equipment	Investment Cost, $	Utility Cost, $/year	Power/Duty, kW	Other
Pump 1	18,398	3045.2	6.344	Centrifuge
Pump 2	15,251	1439.4	2.999	Centrifuge
Heater	15,818	413,260	4549	Multipipe HP Steam, Area = 16.6 m^2
Heat exchanger	16,590	–	–	Multipipe, Area = 18.2 m^2
Cooler	139,829	0.000	4668	Air Cooler, Area = 374.7 m^2
Reactor	118,295	0.000	0 (adiabatic)	Diam. = 0.72 m, Length = 10 m
Column 1				
Condenser	32,248	6325	944	Tubes and shell floating head water, Area = 46.4 m^2
Reboiler	58,048	120,360	1735	Kettle reboiler HP steam, Area = 54.7 m^2
Vessel	10,4270	–	–	D = 0.93 m, H = 8.4 m
Internals	5374	–	–	Packed
Column 2				
Condenser	28,263	17,481	2609	Tubes and shell floating head, Area = 37.8 m^2
Reboiler	12,543	28,0690	3090	Tubes and shell HP steam, Area = 10.78 m^2
Vessel	262,738	–	–	D = 2.9 m, H = 11.0 m
Internals	23,710	–	–	Tray, 21 trays
Total cost	*1,002,202 $/year*			

Independent Variables in Flowsheet

Name	Parameter (units)	Value
TS4	Temperature stream S4, °C	100.00
TS5	Temperature stream S5, °C	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, °C	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, kg mol/h	130.28

	Final Product (DME Stream)	Final Byproduct (Water Stream)
Flow, kg mol/h	135.3	126.19
Temperature, °C	47.14	148.5
Composition, molar fraction	DME = 0.995	DME = 0.000
	MeOH = 0.005	MeOH = 0.005
	Water = 0.000	Water = 0.995

The earlier 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 appear inside the disjunctions.

Tables 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 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 systems where some relations are given in form of implicit blocks of equations.

Three different algorithms have been 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 because of the necessity of using perturbation schemes to minimize the noise introduced by the process simulator. However, the entire methodology can be eventually integrated in a process simulator and then deriva-

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 subproblems	4	9 (111 LP nodes)
CPU time, s	2219	2423
Solvers	SNOPT/ CPLEX	SNOPT/ proprietary BB
Explicit linear equations*	12	
Explicit nonlinear equations*	63	
Binary variables	22	
Independent variables (flowsheet level)*	12	
Other explicit variables*	38	
Implicit blocks of equations excluding flowsheet*	39	

*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.

tives estimated during the convergence, and then all the procedures are considerably speeded up. Note that the total number of major iterations performed by the NLP solvers does not increase (typically 10–20 major iterations).

An important contribution of this paper is demonstrating that it 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, inside the block cannot be violated) which is not true if we relax individual equations. However, a 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.

Literature Cited

1. Grossmann IE, Caballero JA, Yeomans H. Advances in mathematical programming for automated designs, integration and operation of chemical processes. In: *Proceedings of the International Conference on Process Integration (PI'99)*, Copenhagen, Denmark, 1999.
2. Barnicki SD, Siirola JJ. Process synthesis prospective. *Comput Chem Eng*. 2004;28:441–446.
3. Westerberg AW. A retrospective on design and process synthesis. *Comput Chem Eng*. 2004;28:448–458.
4. Grossmann IE. Review of nonlinear and mixed integer and disjunctive programming techniques for process system engineering. *Optim Eng*. 2002;3:227–252.
5. Floudas CA, Akrotiriankis IG, Caratzoulas S, Meyer S, Kallrath J. Global optimization in the 21st century: advances and challenges. Europena Symposium on Computer-Aided Process Engineering-14. Lisbon, Portugal. Published by Elsevier in: Barvosa-Povoa A, Matos H, editors. *Computer Aided Chemical Engineering*, Vol. 18. 2004;23–51.
6. Kravanja Z, Grossmann IE. PROSYN: an MINLP process synthesizer. *Comput Chem Eng*. 1990;14:1363.
7. Kravanja Z, Grossmann IE. New developments and capabilities in PROSYN-an automated topology and parameter process synthesizer. *Comput Chem Eng*. 1994;18:1097–1114.
8. Kravanja Z, Grossmann IE. A computational approach for the modeling/decomposition strategy in the MINLP optimization of process flowsheets with implicit models. *Ind Eng Chem Res*. 1996;35:2065–2070.
9. Caballero JA, Grossmann IE. Aggregated models for integrated distillation systems. *Ind Eng Chem Res*. 1999;38:2330–2344.
10. Papalexandri KP, Pistikopoulos EN. Generalized modular representation framework for process synthesis. *AIChE J*. 1996;42:1010.
11. Diwekar UM, Grossmann IE, Rubin ES. A MINLP process synthesizer for sequential modular simulator. *Ind Eng Chem Res*. 1992; 31:313–322.
12. Reneaume JMF, Koehret BM, Joulia XL. 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.
13. Diaz MS, Bandoni JA. A mixed integer optimization strategy for a large scale chemical plant in operation. *Comput Chem Eng*. 1996;20:531–545.
14. Duran MA, Grossmann, IE. An outer approximation algorithm for a class of mixed-integer nonlinear programs. *Math Program*. 1986; 36:307.
15. Oldenburg J, Marquardt W, Heinz D, Leineweber DB. Mixed-logic dynamic optimization applied to batch distillation process design. *AIChE J*. 2003;49:2900–2917.
16. Caballero JA, Milán-Yañez D, Grossmann IE. Rigorous design of distillation columns. Integration of disjunctive programming and process simulators. *Ind Eng Chem Res*. 2005;44:6760–6775.
17. Biegler LT, Hughes RR. Infeasible path optimization with sequential modular simulators. *AIChE J*. 1982;28:994–1002.
18. Biegler LT. Simultaneous modular simulation and optimization. Proceding on the second International Conference on Foundations of Computer Aided Process Design II, Snowmass, CO, USA. Published by Elsevier, Westerberg AW, Chien HH, editors. CACHE Committee 1984:369–409.
19. Lang YD, Biegler LT. A unified algorithm for flowsheet optimization. *Comput Chem Eng*. 1987;11:143–158.
20. HYSYS v 3.2. Hyprotech Ltd, 1995–2002.
21. Alkaya D, Vasantharajan S, Biegler LT. Generalization of a tailored approach for process optimization. *Ind Eng Chem Res*. 2000; 39:1731–1742.
22. Quesada I, Grossmann IE. An LP/NLP based branch and bound algorithm for convex MINLP optimization problems. *Comput Chem Eng*. 1992;16:937–947.
23. Gupta OK, Ravindran V. Branch and bound experiments in convex nonlinear integer programming. *Manage Sci*. 1985;31:1533–1546.
24. Lee S. Grossmann IE. New algorithms for nonlinear generalized disjunctive programming. *Comput Chem Eng*. 2000;24:2125–2141.
25. Kocis GR, Grossmann IE. Relaxation strategy for the structural optimization of process flowsheets. *Ind Eng Chem Res*. 1987;26:1869.
26. Viswanathan J, Grossmann IE. A combined penalty function and outer approximation method for MINLP optimization. *Comput Chem Eng*. 1990;14:769.
27. Martins JRRA, Kroo IM, Alonso JJ. An automated method for sensitivity analysis using complex variables. *Am Inst Aeronaut Astronaut*. 2000:0689.
28. Leyffer S. *Deterministic Methods for Mixed-Integer Nonlinear Programming*, PhD Thesis. Dundee: Department of Mathematics and Computer Science, University of Dundee, 1993.

29. Drud AS. *CONOPT: A System for Large Scale Nonlinear Optimization. Reference Manual for CONOPT. Subroutine Library*. Bagsvaerd, Denmark: ARKI Consulting and Development A/S, 1996, 69 pp.
30. Gill WPE, Murray W, Saunders. SNOPT: an SQP algorithm for large-scale constrained optimization. *SIAM J Optim*. 2002;979–1006.
31. MATLAB, *The Language of Technical Computing*. The Mathworks Inc., 2006
32. Stubbs R, Mehrotra S. A branch-and-cut method for 0–1 mixed convex programming. *Math Program*. 1999;86:515–532.
33. Sawaya NW, Grossmann IE. Computational implementation of non-linear convex hull reformulation. *Comput Chem Eng*. 2007;31:856–866.
34. Holmström K. The Tomlab Optimization Environment in Matlab. *Adv Model Optim*. 1999;1:47–69.
35. Curtis AR. Powell MJD, Reid JK. On the estimation of sparse Jacobian matrices. *J Inst Math Appl*. 1974;13:117–120.
36. Turkay M, Grossmann IE. Disjunctive optimization techniques for the optimization of process systems with discontinuous investment cost. Multiple size regions. *Ind Eng Chem Res*. 1996;35:2611–2623.
37. Nemhauser GL, Wolsey LA. *Integer and Combinatorial Optimization*. New York: Wiley Interscience, 1999.
38. Seider WD, Seader JD, Lewin DR. *Process Design Principles. Synthesis Analysis and Evaluation*. New York: Wiley, 1999.
39. Turton R, Bailie RC, Whiting WB, Shaeiwitz JA. *Analysis, Synthesis and Design of Chemical Processes*, 2nd ed. New Jersey: Prentice Hall PTR, 2003.
40. Distil v.6.2. Aspen Technology Inc., 2003.

Manuscript received Dec. 4, 2006, revision received Apr. 11, 2007, and final revision received Jun. 20, 2007.