

Integration of Generalized Disjunctive Programming with Modular Process Simulators

José A. Caballero^a, Andrew Odjo^a, Ignacio E. Grossmann.^b

^a *Departemen of Chemical Engineering, University of Alicante, Ap. Correos 99, Alicante 03080, Spain*

^b *Department of Chemical Engineering, Carnegie Mellon University, 5000 Forbes Avenue, 15213, Pittsburgh, PA.. USA.*

Abstract

This paper describes a computational approach for performing structural optimization in process flowsheets that are described by implicit models in a sequential modular process simulator or in a third party program. The key points are how to select the design variables, how to deal with implicit equations and variables, and how to generate a Master MILP problem including all the relevant information.

Keywords: Disyuntive Programming, Process Synthesis, Process Simulators.

1. Introduction

Generalized Disjunctive Programming is becoming increasingly popular as a solution technique for optimization problems involving discrete decisions, and it has been widely applied in design or retrofit of chemical processes both at the level of structural design (a given equipment appear or not in the final configuration) or at the level of equipment (i.e. discontinuous cost functions defined over the design variables sizes, etc).

In order to avoid solving very large problems, it is common to use shortcut or aggregated models, which try to capture the ‘main’ aspects of the process. However this is also a limitation that could produce important deviations from the actual process behavior.

On the other hand, Modular Process Simulators include reliable and robust models for almost any unit operation, with huge databases that include, physical properties, correlations in order to estimate unknown properties or other parameter like the transport coefficients and a large variety of thermodynamic models that cover almost any known application. But even more important, Modular Simulators include especially tailored numerical methods specifically developed for each unit operation instead of a generic solver, which greatly

improves the numerical robustness of these simulators (maybe at the expense of slowing down the convergence of the system).

Nowadays practically all modular process simulators include optimization capabilities; however, they are constrained to optimize only operational conditions for a fixed flowsheet topology with only smooth external constraints. Capabilities like structural optimization, or the possibility of using discontinuous external constraints like costs functions defined in terms of size variables or in general any conditional or discrete constraint are not included.

In this paper we show different algorithms to integrate GDP and Process simulators, not only at the level of structural decisions, but with any conditional constraint as for example discontinuous costs functions. The use of process simulators in a modular environment for solving MINLP has been addressed by Diwekar et al. (1992) Reneaume et al. (1995) and Díaz and Bandoni (1996). All these works are based on the augmented penalty/equality relaxation outer-approximation algorithm. Kravanja and Grossmann (1996) followed a similar approach, adapting the modeling/decomposition (M/D) strategy developed by Kocis and Grossmann (1987) that can be considered a precursor of generalized disjunctive programming.

2. GDP formulation in a Modular Process Simulator Environment.

When we defined an optimization model with conditional equations in a modular environment we can differentiate three kinds of variables: 1. the design or independent variables (x). These are equal to the degrees of freedom of the problem and form the set of variables over which the optimizer has full control. 2. Variables that are calculated by the simulator (u) and that in general can only be read. 3. Variables that must be fixed for a given topology in the flowsheet – for example number of trays in a distillation column, binary or integer variables, etc- but that can be manipulated by the solver in each iteration. In the same way, we can differentiate two classes of equations: 1. implicit equations that are all the equations solved by each of the modules in the process simulator (or any other external module added to the program). These equations are usually considered “black box input-output” relationships because we have not access either the equations or the way in which those equations are internally treated. However, there is an important danger hidden in the equations introduced in a gradient based optimization environment: They could include points in where some of these equations are non differentiable, therefore we cannot consider these systems like completely black boxes but we should have at least a general knowledge of the system of equations in order to anticipate this behavior and correctly model the process. 2. External or explicit equations over which we have a complete control. The disjunctive formulation of the problem can then be written as follows:

$$\begin{array}{ll}
\min : z = f(x, u, p) & \\
s.t. \quad r_I(x, u, p) = 0 & \\
\quad r_E(x, u, p) = 0 & \\
\quad s_E(x, u, p) \leq 0 &
\end{array}
\quad
\begin{array}{l}
\left[\begin{array}{c}
Y_{i,j} \\
h_{I_{i,j}}(x, u, p) = 0 \\
\bigvee_{i \in D} h_{E_{i,j}}(x, u, p) = 0 \\
g_{E_{i,j}}(x, u, p) \leq 0 \\
p = p_{i,j}
\end{array} \right] \quad \forall j \in J
\end{array}
\quad
\begin{array}{l}
\Omega(Y) = TRUE \\
x \in X \subseteq \mathfrak{R}^n; \\
Y \in \{True, False\}^m
\end{array}
\quad (1)$$

Where the index I makes reference to implicit equations and the index E makes reference to the explicit ones.

Starting with the problem formulated in equation (1) it is possible to follow different alternatives.

1. - If there are no implicit equations inside the disjunctions then the problem can be directly reformulated to an MINLP using a big-M or a convex hull reformulation, Grossmann, (2002). If the resulting MINLP (or even the original GDP) is solved using a branch and bound strategy then no major problems are expected because the problem is reduced to solve a set of NLPs with some subset of binary (Boolean) variables fixed at each branch of the tree during the search. However, this approach could be very time consuming due to the potential large number of NLPs that must be solved (remember that each NLP include the convergence of the flowsheet inside the process simulator and although it is 'acyclic' because all the recycles are specify as external constraints, it could take an important amount of time). If an algorithm like outer approximation is used then we must generate a Master MILP problem. In this case, it is necessary to perform equation linearizations in terms of independent variables. In the case of explicit equations it can be done analytically if the functional relationship between x and u variables is known, but usually that relationship is not known and then a numerical approach in where the value of the output variables (u) are calculated for an adequate perturbation of the input variables (x) must be used.

2.- If the implicit equations appear also in the disjunctions then the direct reformulation to an MI(N)LP problem has very bad performance. The reason is that zero flows in non-existing units could prevent the simulator to converge and then the entire optimization procedure fail as well. The best option in this case is developing an initialization strategy in order to get information for the first master problem. Two options are available: 1 solving a 'set covering' problem to get the minimum number of feasible flowsheets that include all the alternatives (Turkay and Grossmann, 1996) or develop a sub-Lagrangian optimization (Kravanja and Grossmann, (1996)).

In both alternatives, the Master problem is solved in the projection space of independent variables, where the relation between implicit and independent variables can be obtained directly from the simulator if the jacobian matrix is available or by an adequate numerical perturbation in other case.

3. Example

This example is a modification of the problem proposed by Seider et al, (1999) 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 nC₄ and lighter products and a combined mole percentage of at least 99.5%.

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. All the process is controlled from Matlab that contain the disjunctive formulation and controls all the solution procedure. The optimizer is also external to the process simulator and controlled by Matlab as well.

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 depend on 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 the size and cost equations over which we have full control. The reasons of using these equations as implicit are : 1. They decrease the dimensionality of the problem at the optimization level and 2. 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.

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

$$\min : TAC = 0.2(\text{investment cost}) + \text{Utilities_cost}$$

$$\left[\begin{array}{l} Y_electric_driver \\ cost_driver = f(power) \end{array} \right] \vee \left[\begin{array}{l} Y_combustion_driver \\ cost_driver = f(power) \end{array} \right]$$

$$\left[\begin{array}{l} Y_air_cooler1 \\ cost_cooler1 = f(A_1) \\ size_cooler1 = f(TS0, TS1, Tinair, \\ Toutair, Qc_1) \\ CUC_1 = 0 \end{array} \right] \vee \left[\begin{array}{l} Y_floating_head_cooler1 \\ cost_cooler1 = f(A_1) \\ size_cooler1 = f(TS0, TS1, Tinair, \\ Toutair, Qc_1) \\ \left[\begin{array}{l} Z_cool_water \\ CUC1 = Qc_1 \cdot Ccw \end{array} \right] \vee \left[\begin{array}{l} Z_water \\ CUC1 = Qc_1 \cdot Cw \end{array} \right] \end{array} \right]$$

$$\begin{aligned}
& \begin{bmatrix} Y_cooler2_R1 \\ CUc_2 = CR1 \cdot Qc_2 \\ cost_cooler2 = f(A_2) \\ size_cooler2 = f(TS1, \\ TS2, TinR1, ToutR1, Qc_2) \end{bmatrix} \vee \begin{bmatrix} Y_cooler2_R2 \\ CUc_2 = CR2 \cdot Qc_2 \\ cost_cooler2 = f(A_2) \\ size_cooler2 = f(TS1, \\ TS2, TinR2, ToutR2, Qc_2) \end{bmatrix} \vee \begin{bmatrix} Y_cooler2_R3 \\ CUc_2 = CR2 \cdot Qc_2 \\ cost_cooler2 = f(A_2) \\ size_cooler2 = f(TS1, \\ TS2, TinR3, ToutR3, Qc_2) \end{bmatrix} \\
& \begin{bmatrix} Y_heater1_LP \\ CUh_1 = CLP \cdot Qh_1 \\ cost_heater1 = f(A_{h1}) \\ size_heater1 = f(TS3, TS8, TLP, Qh_1) \end{bmatrix} \vee \begin{bmatrix} Y_heater1_HotWater \\ CUh_1 = CHW \cdot Qh_1 \\ cost_heater1 = f(A_{h1}) \\ size_heater1 = f(TS3, TS8, TinHW, \\ ToutHW, Qh_1) \end{bmatrix} \\
& \begin{bmatrix} Y_condenserW \\ CUcond = Cw \cdot Qcd \\ cost_Cd = f(A_{cond}) \\ size = f(TV1, TL1, \\ Twin, Twout, Qcd) \end{bmatrix} \vee \begin{bmatrix} Y_condenserR1 \\ CUcond = CR1 \cdot Qcd \\ cost_Cd = f(A_{cond}) \\ size = f(TV1, TL1, \\ TR1in, TR1out, Qcd) \end{bmatrix} \vee \begin{bmatrix} Y_condenserR2 \\ CUcond = CR2 \cdot Qcd \\ cost_Co = f(A_{cond}) \\ size = f(TV1, TL1, \\ TR2in, TR2out, Qcd) \end{bmatrix} \vee \begin{bmatrix} Y_condenserR3 \\ CUcond = CR3 \cdot Qcd \\ cost_Cd = f(A_{cond}) \\ size = f(TV1, TL1, \\ TR3in, TR3out, Qcd) \end{bmatrix} \\
& \begin{bmatrix} Y_reboilerMP \\ CUreb = CMP \cdot Qreb \\ cost_Reb = f(A_{reb}) \\ size_reb = f(TL2.TV2, T_{MP}, Qreb) \end{bmatrix} \vee \begin{bmatrix} Y_reboilerHP \\ CUreb = CMP \cdot Qreb \\ cost_Reb = f(A_{reb}) \\ size_reb = f(TL2.TV2, T_{HP}, Qreb) \end{bmatrix}
\end{aligned}$$

$$cost_compressor = f(power)$$

$$size_flash_i = f(Volume\ flow\ and\ density\ of\ output\ streams) \quad i = 1, 2$$

$$cost_flash = f(H_i, D_i) \quad i = 1, 2$$

$$cost_column = cost\ vessel(Hc, Dc) + cost\ internals$$

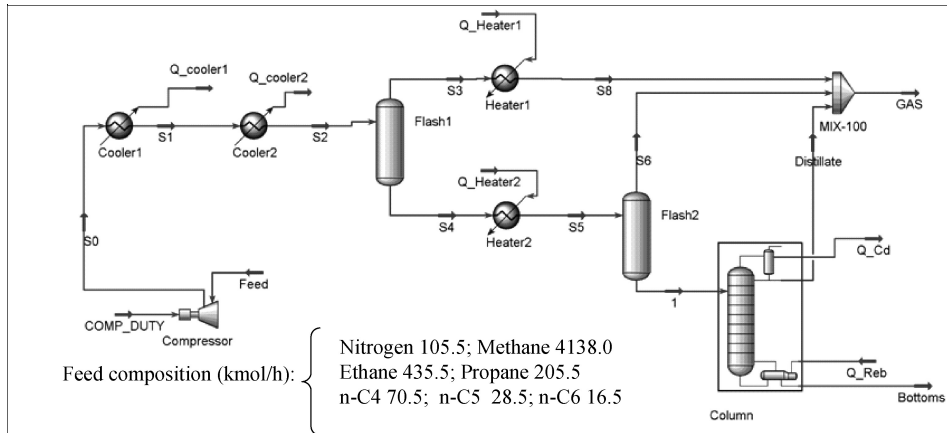


Figure 1. Basic flowsheet of the Gas Natural Plant in the example

Previous problem was solved using a mixed big-M convex hull (for the linear equations) reformulation to a MINLP problem. Cost correlations were taken from Turton et al (1998). Data for utility costs and heat transfer coefficients was taken from the Database of the program DISTILL[®].

The relaxed problem initial problem produce an objective value of $102.15 \cdot 10^4$ \$/year, only a 12.7% lower than the optimal value $117.01 \cdot 10^4$ \$/year. Convergence is obtained in only one major iteration.

The optimal solution include driver of the compressor must be a combustion engine. Refrigeration in coolers must be done with cool water and R1 refrigerant respectively. Hot water is used as heating media in heater 1. The condenser in the distillation column should use refrigerant R3 and the reboiler must be heated with medium pressure vapor steam.

References

1. Díaz, M. S.; Bandoni, J. A. A Mixed Integer Optimization Strategy for a Large Chemical Plant in Operation. *Comput. Chem. Eng.* 1996, 20 (5), 531-545.
2. Diwekar, U.M.; Grossmann, I.E.; Rubin, E.S. An MINLP Process Synthesizer for a Sequential Modular Simulator. *Ind. Eng. Chem. Res.* 1992, 31, 313-322.
3. Grossmann, I.E. Review of Nonlinear –Mixed Integer and Disjunctive Programming Techniques. *Optimization and Engineering*, 3, 227–252, 2002
4. Kocis, G. R.; Grossmann, I. E. Relaxation Strategy for the Structural Optimization of Process Flowsheets. *Ind. Eng. Chem. Res.* 1987, 26, 1869-1880.
5. 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, 2065-2070.
6. M. Turkay and I. E. Grossmann, "Alogic based outer-approximation algorithm for MINLP optimization of process flowsheets," *Computers and Chemical Engineering* vol. 20, pp. 959–978, 1996.
7. Reneaume, J.M.F.; Koehret, B.M.; Joulia, X.L. Optimal Process Synthesis in a Modular Simulation Environment: New Formulation of the Mixed Integer Nonlinear Programming Problem. *Ind. Eng. Chem. Res.* 1995, 34, 4378-4394.
8. Seider, W.D.; Seader, J.D.; Lewin, D.R.; *Process Design Principles. Analysis Synthesis and Evaluation.* Ed by Jhon Willey and Sons. 1999.
9. Turton, R.; Bailie, R. C.; Whiting, W. B.; Shaeiwitz, J. A. *Analysis, Synthesis and Design of Chemical Processes.* McGraw- Hill: New York, 1998.

Acknowledgements

The authors gratefully acknowledge financial support from Spanish "Ministerio de Ciencia y tecnología" under project CTQ2005-05456.