A Disjunctive Programming Model and a Rolling Horizon Algorithm for Optimal Multiperiod Capacity Expansion in a Multiproduct Batch Plant

Gabriela García-Ayala[1]

Department of Chemical Engineering, Tecnológico de Monterrey,

Eugenio Garza Sada 2501 Sur, Monterrey, NL 64849, Mexico

e-mail: *mg_garcia@yahoo.com*

Roger Z. Ríos-Mercado Óscar L. Chacón-Mondragón

Graduate Program in Systems Engineering, Universidad Autónoma de Nuevo León,

AP 111-F, Cd. Universitaria, San Nicolás de los Garza, NL 66450, Mexico

e-mail: { *roger*, *ochacon* }*@ yalma.fime.uanl.mx*

May 2011

[1] Corresponding author

Abstract: In this paper we address a multi-period mixed integer non-linear problem for the capacity expansion of multiproduct batch plants. In this problem, given a certain batch plant with its current configuration, product recipes, and growing production targets, modular expansions are wanted so that new demand can be met. Unlike most work for the batch retrofit problem found in literature, a multi-period disjunctive model is presented, so that long term investments and expansions can be planned out in advance. Although effective for short periods, the proposed model becomes computationally inefficient for long time horizons. To address this issue, we propose a rolling horizon algorithm that further exploits the advantages of a disjunctive programming model. A numerical example based on a case study from industry is presented that shows that the rolling horizon algorithm is very effective on finding near optimal solutions to large instances with a considerable number of time periods. Furthermore, empirical evidence shows how the solution found by the proposed algorithm can be used as a starting solution for the direct method for the original problem to deliver a global optimal solution to the problem.

Keywords: batch retrofit; multiproduct batch plants; multiperiod MINLP model; disjunctive programming; rolling horizon algorithm.

1. INTRODUCTION

Typically, batch production involves a general purpose facility where a wide variety of products can be produced with different processing recipes by sharing all available resources such as equipment, raw material, intermediates and utilities (Pinto, Barbosa-Póvoa, and Novais, 2005).

Determining the capacity of any plant depends on the amount of product that it is able to produce. However, in a batch plant such capacity not only depends on the capacity of the installed equipment but also on the production scheduling, the number of products and their recipes, the changeover times between products, and several other factors. To calculate such capacity is not trivial work. It needs a detailed analysis of all process times for each product, along with the available equipment. Even though equipment may never be out of work, it may be the case where the batch scheduling is not optimal leaving capacity in disuse. Sometimes after an optimal scheduling a plant expansion is not even necessary (Macchietto, 2005).

This work deals with the capacity expansion of a batch plant such that new production targets can be met. This problem is known as the retrofit problem. The retrofit is an optimization problem whose objective is to obtain a new plant layout starting with an actual plant configuration such that the benefits are maximized subject to a new demand. The solution is a plant configuration map where equipment that is not used is sold and new equipment is acquired and adjusted to work with existing equipment (Montagna, 2003). The proposed formulation differs from that in literature (Barbosa-Póvoa, 2007) in extending the retrofit problem to a long horizon in order to allow investment planning by using a multiperiod model. To the best of our knowledge, the most relevant model of a multiperiod

batch retrofit is due to Moreno, Montagna, and Iribarren (2007); however, their model is limited and does not allow for variations in the plant configuration during the time horizon in their model.

By extending the retrofit problem to a multiperiod model, the size of the problem increases drastically. In order to keep the problem tractable, a disjunctive programming model is introduced as an alternative model to the MINLP problem by using disjunctions and logic propositions (Raman and Grossmann, 1994). Disjunctive programming is based on the idea of expressing constraints (equalities and inequalities) in terms of global constraints that always should hold. These global constraints may be disjunctions that correspond to conditional constraints in the continuous space, and logic propositions in the discrete space. All these constraints are expressed in terms of Boolean and continuous variables, which are selected to optimize a given objective function subject to the various types of constraints (Lee and Grossmann, 2003). Disjunctive programming has been proven to be effective in terms of providing a qualitative and quantitative framework for modeling a number of applications ranging from desalting plants to distillation columns (Mussati et al., 2008; Caballero, Milán-Yañez, and Grossmann, 2005). Among the various applications it is shown that a disjunctive model representation provides a very flexible, intuitive and effective way to formulate discrete optimization problems (Oldenburg and Marquardt, 2008).

It has been observed that optimization algorithms for disjunctive programming formulations are in many cases more efficient than the ones developed for their regular full space models (Grossmann, 2004).

Even though disjunctive programming was used to keep the problem solvable for large time periods, it proved insufficient. A planning horizon for 20 years is to be considered, and

the disjunctive model could not find solution for such amount of time periods. For this reason, a rolling horizon algorithm is additionally proposed.

All the rolling horizon algorithms give approximations of the optimal solution with a significant decrease in their computational requirements. The algorithm provides a feasible solution for the original problem in reasonable time. Furthermore, such solution was used as a starting point to the direct method on the original model to find the global optimum.

Rolling horizon algorithms work by separating a problem into a sequence of iterations, each of which models only part of the horizon in detail (Dimitriadis, Shah, and Pantelides, 1997). The rest of the horizon is modeled with a relaxed model (Erdirik-Dogan and Grossmann, 2007a).

This paper has been motivated by a real-world problem in a local brewery, *Cervecería Cuauhtémoc Moctezuma*. The specific goal is to propose a mutiperiod model for the retrofit design of multiproduct batch plant over a long planning horizon. Taking into consideration the scale of the problem, a disjunctive programming was used to try and help solution times. We also investigate some solution strategies in the rolling horizon approach such as priority branching for reducing the computational effort.

The algorithm provides a feasible solution for the problem in reasonable time. Furthermore, in this specific case the problem was solved optimally since the RHA suboptimal solution was used as a starting point to the direct method on the original model to find the global optimum. This assesses that the RHA solution had an optimality gap of 8.6%, which is quite reasonable for industry standards.

The rest of the paper is organized as follows. First, in Section 2 the problem definition is given, including the notation, the disjunctive programming model, and numerical

examples that illustrates the usefulness of the proposed model. Then, in Section 3, we describe in detail the proposed rolling horizon algorithm for handling longer time periods of this problem. Section 4 shows the empirical work, where the model and solution approach is evaluated on some instances based on real-world data. This is followed by a discussion and conclusions in Section 5.

2. PROBLEM DESCRIPTION AND MODELING FRAMEWORK

Given a batch plant, with a series of equipment, products, and a growing demand, the goal is to find a program of staged expansions that allow the demand to be met at every time period of the horizon. The performance measure to be minimized is the expansion cost generated by the acquisition of new equipment. The general idea is to optimize the production rate of each product in the plant. The production rate is a function of the batch size and the cycle time, where each product has its own production rate. For this reason a production scheme must be considered. In the specific case of this work, the most convenient scheme is single product campaigns. For this specific application, a single product campaign is adopted.

Given a growing demand for a set of products and a plant configuration, the problem consists of deciding when and where new equipment must be added in order to meet the production targets. In each considered time period, the plant can grow in any of its production stages by adding new equipment. The outcome is a calendar of expansions giving equipment size and investment for each time period.

The assumptions by which the proposed model works correspond to those commonly used in the optimal design of multiproduct batch plants (Vaselenak, Grossmann, and

Westerberg, 1987) which are: the recipes for all products are given, fixed processing times are specified for each of the products in each type of equipment, the products are manufactured sequentially, a continuous range of equipment sizes is assumed to be available, and the number of batches is permitted to be non integer since this is usually a large number.

2.1 Notation

Sets and Indices:

I	Set of products; $i \in I$
J	Set of production stages; $j \in J$
K	Set of new unites per production stage; $k \in K$
T	Set of time periods; $t \in T$
M	Set of existing units in initial plant configuration; $m \in M$

Parameters:

N	The number of products manufactured
N_j^{old}	The number of existing units in stage j
V_{jm}^{old}	The volume of existing unit m in stage j
T_{ij}	The process time of product i in stage j
H	The operating time period
S_{ij}	The size factor of product i in stage j
K_{jt}	The annualized fixed charge of installing a new unit in stage j in period t
C_{jt}	The annualized cost coefficient of installing a new unit in stage j in period t
Q_{it}	The demand of product i in period t

V_j^L The minimum volume of new units in stage j

V_j^U The maximum volume of new units in stage j

Z_j The maximum number of units that can be added to stage j

Z^U The maximum number of units that can be added to the plant

Binary decision variables:

y_{jk} Selection of investment of unit k in stage j; (= 1) if unit k is chosen for investment in stage k; (= 0) otherwise

w_{jkt} Operation of unit k in stage j in period t; (= 1) if unit k is in operation in stage j in period t; (= 0) otherwise

w^B_{ijkmt} Operate new unit k in phase with existing unit m for product i in stage j in period t; (= 1) if unit k is operated in phase with existing unit m for product i in stage j in period t; (= 0) otherwise

w^C_{ijkt} Operate new unit k in sequence with existing units for product i in stage j in period t; (= 1) if new unit k is operated in sequence with existing units for product i in stage j in period t; (= 0) otherwise

z_{jkt} Expansion/installation of new unit k in stage j in period t; (= 1) if unit k is expanded in stage j in period t; (= 0) otherwise

Continuous decision variables:

N_{it} The number of batches of product i in period t

B_{it} The batch size of product i in period t

T^L_{it} The limiting cycle time of product i in period t

V_{jkt} The volume of new unit k in stage j in period t

E_{jkt} The expansion volume of new unit k in stage j in period t

V^{B}_{ijkmt} The volume required in new unit k in stage j for product i to use it in phase with existing unit m in period t

V^{C}_{ijkt} The volume required in new unit k in stage j for product i to use it in sequence with existing units in period t

CE_{jkt} Expansion/installation cost for new unit k in stage j in period t

2.2 Disjunctive Model

To handle the multiperiod aspect we are introducing a time-indexed model that extends the model by Fletcher, Hall, and Johns (1991). We use the same notation, except that some parameters and variables have in addition a time index. The expansions happen just once among the modeling horizon and are equivalent to installing a new unit. A convexified formulation of the feasible domain is used in order to guarantee a global optimum. The multiperiod batch retrofit problem is addressed with a disjunctive model, based on the general disjunctive multiperiod model proposed by Van den Heever, Grossmann, and Vasantharanjan (2000).

A multiproduct batch plant for manufacturing N products and consisting of M stages in sequence with parallel equipment in each stage is considered as shown in Figure 1.

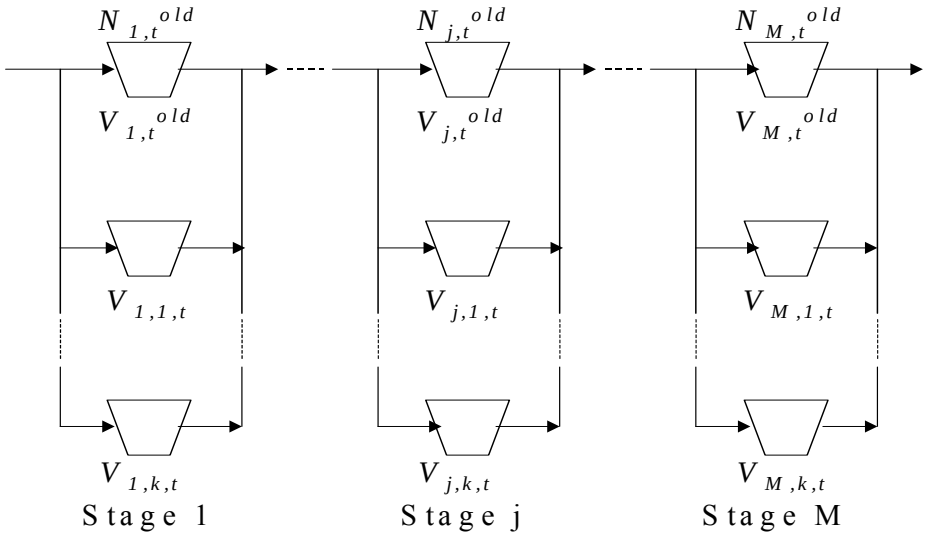

Figure 1: Superstructure for retrofit design of multiproduct batch plant (Vaselenak, Grossmann, and Westerberg, 1987)

We define the following variables to apply the exponential transformation (Vaselenak et al., 1987): $x_{it}^{(1)} = \ln N_{it}$, $x_{it}^{(2)} = \ln B_{it}$, $x_{it}^{(3)} = \ln T_{Lit}$. The multiperiod formulation, obtained by applying the general disjunctive model is then as follows:

Detailed Disjunctive Design (DDD) model

i) Objective function:

$$\min \sum_{t} \sum_{j} \sum_{k} CE_{tjk}$$

ii) Production targets:

$$N_{ti} B_{it} \geq \mathrm{Q}_{\mathrm{it}} \qquad \forall i,t \text{ with exponential transformation}$$

$$x_{it}^{(1)} + x_{it}^{(2)} \geq \ln \mathrm{Q}_{\mathrm{it}} \quad \forall i,t$$

iii) Limiting cycle time of product *i*:

$$\frac{\mathrm{T}_{\mathrm{ij}}}{T_{Lit}} - \sum_{k} w_{ijkt}^{B} \leq \mathrm{N}_{\mathrm{j}}^{\mathrm{old}} \qquad \forall i, j, t \text{ ; applying exponential transformation}$$

$$\mathrm{N}_{\mathrm{j}}^{\mathrm{old}} + \sum_{k} w_{ijkt}^{C} \geq \mathrm{T}_{\mathrm{ij}} \exp(-x_{it}^{(3)}) \qquad \forall i, j, t$$

iv) Yearly operating time:

$$\sum_{i} N_{it}\ T_{Lit} \leq \mathrm{H}_{\mathrm{t}} \qquad \forall t \text{ ; applying exponential transformation}$$

$$\sum_{i} \exp(x_{it}^{(1)} + x_{it}^{(3)}) \leq \mathrm{H}_{t} \qquad \forall t$$

v) Bound on total number of new units:

$$\sum_{j}\sum_{k} y_{jk} \leq \mathrm{Z}^{\mathrm{U}}$$

vi) Option B capacity constraints:

$$\sum_{k} V_{ijkmt}^{B} + \mathrm{V}_{\mathrm{jm}}^{\mathrm{old}} \geq \mathrm{S}_{\mathrm{ij}} B_{it} \qquad \forall i, j, m, t$$

vii) Distinct assignment of new units:

$$y_{jk} \geq y_{j,k+1} \qquad \forall j, k = 1 \ldots Z_{j} - 1$$

viii) Disjunction for every unit k added to stage j:

$$\left[\begin{array}{c} y_{jk} \\ V_{jkt} \leq V_{j}^{U} \\ V_{jkt} = V_{jk,t-1} + E_{jkt} \\ \left[\begin{array}{c} w_{jkt} \\ V_{jkt} \geq V_{j}^{L} \\ \left[\begin{array}{c} w_{ijk1t}^{B} \\ V_{ijk1t}^{B} \leq V_{jkt} \\ V_{ijk1t}^{B} \leq V_{j}^{U} \end{array}\right] \vee \ldots \vee \left[\begin{array}{c} w_{ijkmt}^{B} \\ V_{ijkmt}^{B} \leq V_{jkt} \\ V_{ijkmt}^{B} \leq V_{j}^{U} \end{array}\right] \vee \left[\begin{array}{c} w_{ijkt}^{C} \\ V_{ijkt}^{C} \leq V_{jkt} \\ V_{ijkt}^{C} \leq V_{j}^{U} \\ V_{ijkt}^{C} \geq B_{it} S_{ij} \end{array}\right] \forall i \\ \left[\begin{array}{c} z_{jkt} \\ CE_{jkt} = \mathrm{K}_{\mathrm{jt}} + \mathrm{C}_{\mathrm{jt}} E_{jkt} \\ \mathrm{V}_{\mathrm{j}}^{\mathrm{L}} \leq E_{jkt} \leq \mathrm{V}_{\mathrm{j}}^{\mathrm{U}} \end{array}\right] \vee \left[\begin{array}{c} \neg z_{jkt} \\ CE_{jkt} = 0 \\ E_{jkt} = 0 \end{array}\right] \end{array}\right] \vee \left[\begin{array}{c} \neg w_{jkt} \\ V_{jkt} \geq 0 \end{array}\right] \forall t \end{array}\right] \vee \left[\begin{array}{c} \neg y_{jk} \\ V_{jkt} = 0 \end{array}\right] \forall j, k$$

ix) Logic relationships:

$$\sum_m w^B_{ijkmt} + w^C_{ijkt} = w_{jkt} \quad \forall i, j, k, t$$
$$\sum_t z_{jkt} = y_{jk} \quad \forall j, k$$
$$y_{jk} \geq \sum_t w_{jkt} \quad \forall j, k$$
$$w_{jkt} \leq y_{jk} \quad \forall j, k, t$$
$$w_{jkt} \leq \sum_{\tau=1}^{t} z_{jk\tau} \quad \forall j, k, t$$
$$z_{jkt} \leq w_{jkt} \quad \forall j, k, t$$

x) Variables:

$$N_{it}, B_{it}, T_{Lit}, V_{jk}, E_{jkt}, V_{jkt}, V^B_{ijkmt}, V^C_{ijkt} \geq 0 \quad y_{jk}, w_{jkt}, w^B_{ijkmt}, w^C_{ijkt}, z_{jkt} = \{0,1\}$$
$$i \in I, \quad j \in J, \quad t \in T, \quad k \in Z, \quad m \in N^{old}_j$$

Convergence to the optimal solution is guaranteed in a finite number of iterations since the model is convex (Vaselenak, Grossmann, and Westerberg, 1987).

2.3 Numerical example

In order to show the advantages of using disjunctive programming, a small example of 10 time periods is solved in GAMS 22.5 using DICOPT as a solver on a Dell DXP051 with 3192Mhz and 2GB. CPLEX 11.0 and CONOPT were used respectively as the MIP and NLP solvers called upon by DICOPT. This modeling system, computer and solvers will be used for all examples throughout this paper.

The data for this example can be found in Section 4, using only the first 10 periods from the product demands. Table 1 has the results for the problem solved under different models.

Table 1. 10 period problem solved by different models.

Model	Discrete variables	Continuous variables	Number of equations	Solution time CPU sec.	Expansion Cost ($1000)
Full space non-convex model.	2,220	5,267	7,122	***	***
Full space convexified model*	2,220	4,097	5,932	623452	**1024.06
Model DDD	2,220	5322	8,958	41226	950.36
Model DDD with slack variables and priority branching	2,220	5,297	7,152	6328	950.36

*Convexified model proposed by Vaselenak et al., 1987.
** Best solution found while using all resources available.
*** Solver reports model as infeasible.

For the full space non-convex model, the solver reported the problem as infeasible. The convexified model did not find the optimal solution and reports the best integer solution found, with a value of $1024.06, taking 623,452 CPU seconds. The disjunctive model, model DDD finds the optimal solution in 41,226 seconds. Finally model DDD, with the addition of slack variables and using priority branching finds the same optimal solution in just 6,328 seconds.

Priority branching helps the solution time considerably because it takes advantage of the hierarchical structure of the problem in the disjunctions (see equation viii) by branching first on y_j, then on w_{jt} and then on z_{jt}.

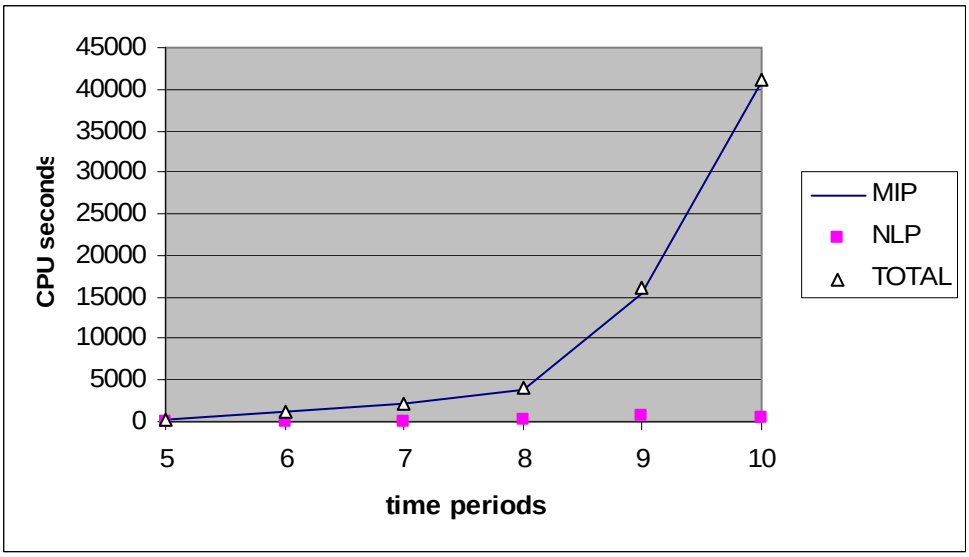

Figure 2: MINLP solution times.

To have an idea on how much time is spent on solving the MIP master problem and NLP subproblem, we solved several instances for different time periods (5 to 10) based on the same data. Figure 2 shows the computational time for the MIP and NLP sections of the instances tested. As we can see, the solution time is practically associated with the time it takes in solving the master problems. This is an important reason that motivates the the use of disjunctive programming. As mentioned before, this proposed model is successful and a very valuable tool for attempting to find decisions to problems with 10 or less time periods; however, when attempting to solve instances with time horizon of 20 time periods, the direct use of the solver proved insufficient, motivating the development of the proposed solution approach.

3. ROLLING HORIZON ALGORITHM

The model previously presented can be solved directly with branch and bound methods. For problems of considerable size and complexity, involving long time horizons, the computational effort can be expensive. In order to obtain solutions for large problem instances, a rolling horizon algorithm (RHA) is considered, to aid the solution time and to be able to include considerable number of time periods.

The RHA is a heuristic framework used to reduce the computational effort of multiperiod problems while finding an approximation of the optimal solution. Instead of solving the complete design horizon, the problem is decomposed into a sequence of sub-problems that are solved recursively (Beraldi et al., 2008).

For a horizon of H time periods and taking r periods at a time, using f as a counter and model DDD as a base model the Rolling Horizon Algorithm is shown in Figure 3.

The time horizon is partitioned differently for each sub problem. In every subproblem the initial part of the partition is modeled with the Detailed Disjunctive Design problem (DDD), the rest of the horizon is modeled with the relaxation of the DDD problem, which we will call Relaxed Disjunctive Design problem (RDD). The relaxation of the discrete variables is used in RDD. The binary variables found in the solution of the detailed subproblem are fixed and the algorithm proceeds to the next partition and subproblem (Eridirik-Dogan and Grossmann, 2007b). This helps the model because it keeps the information of the complete horizon in each iteration of the RHA.

Procedure RHA (P, H, r)

Input: P := An instance of the problem; H := Number of planning horizon time periods;

r := Number of time periods that sets the size of the subproblem

Output: X := A feasible solution for the problem

1. $start = 1$ (initialize period counter)
2. while ($start \leq H$) do
3. $end = \min \{ start + r - 1, H \}$
4. binary variables obtained previously for periods [0, $start - 1$] are fixed
5. binary variables for future periods [$end + 1$, H] are relaxed
6. X = Solution of model with optimal binary variables for periods [$start$, end]
7. $start = start + r$ (update period counter)
8. end-while
9. return X
10. stop

Figure 3. Pseudo-code of Rolling Horizon Algorithm.

In each sub-problem the periods solved by the detailed problem increase, meanwhile the periods solved by the relaxed problem decrease, as seen in Figure 4. The computational complexity of the rolling horizon algorithm is practically the same as the one for DDD since this model is solved in each iteration and the number of iterations is relatively small because most of the binary variables are being fixed to those obtained in previous iterations even though the size of the detailed problem increases with each iteration. This recurrent scheme keeps going until the complete horizon has been solved for the detailed problem.

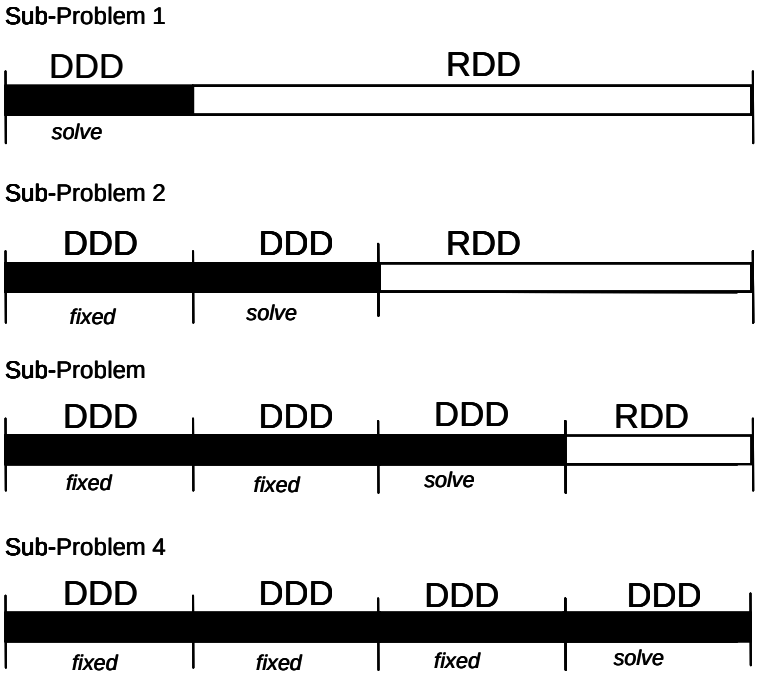

Figure 4. Rolling horizon algorithm scheme (Erdirik-Dogan and Grossmann, 2007b).

Even though it is possible to fix every variable to the value obtained in the sub-problems, only binary variables are fixed in each new subproblem. These represent the plant configuration. The continuous variables (volumes, batch size, etc.) are left free in order to reduce possible infeasibilities as usual.

4. EMPIRICAL WORK

RHA is a succession of MINLP solved to optimality. Global optimality is shown in detail for the uniqueness of the solution of the NLP subproblems for a model of one time period in Vaselenak, Grossmann, and Westerberg (1987). The proof is done by reducing the NLP subproblem to a nonlinear program that involves a linear objective function, linear inequalities and cuasi-convex inequalities. It then follows that if a Kuhn-Tucker point exists,

it will correspond to the global optimum solution. This means that all NLP subproblems that arise from our original model have a unique local optimum provided the productions are all greater than or equal to zero. The equations added to such model are all linear, thus optimality conditions remain.

An example with three products and four production stages is solved for a design horizon of 20 years. Parameters for the example are given in Tables 1 through 6. Table 1 has the number of new units allowed to be added Z_j, the number of existing units in each stage N_j^{old}, and the upper and lower limits for the volume of the new units, as well as the fixed and variable coefficients for the expansion costs. Table 3 contains the demand information for each product in each time period. Table 4 has the volume data of the initial plant configuration. Tables 5 and 6 have the processing times, and the size factor for each product in each stage respectively.

Table 7 contains the results of applying the rolling horizon algorithm to model DDD for a design horizon of 20 time periods. The rolling horizon algorithm pretends to use the information of the relaxed model and its ability to find optimal solution fast, 251 CPU seconds (second line in Table 6), to find the discrete variables in a shorter time horizon, the one of the sub-problem being solved by the detailed problem in that iteration.

Table 2. Parameter values for the example.

Parameters	stage *j* 1	2	3	4
Z_j	10	10	10	10
N_j^{old}	1	1	2	1
V^{lo}_j	1	2.5	2.5	2
V^{up}_j	10	10	10	10
$Cjt \quad \forall t$	13.29	35.21	42.85	7.19
$K_{jt} \quad \forall t$	0.01329	0.03521	0.04285	0.00719

Table 3. Demand Q_{it}

time period	products 1	2	3
1	200.0	600.0	1000.0
2	220.0	720.0	1150.0
3	242.0	864.0	1322.5
4	266.2	1036.8	1520.9
5	292.8	1244.2	1749.0
6	322.1	1493.0	2011.4
7	354.3	1791.6	2313.1
8	389.7	2149.9	2660.0
9	428.7	2579.9	3059.0
10	471.6	3095.9	3517.9
11	518.7	3715.0	4045.6
12	570.6	4458.1	4652.4
13	627.7	5349.7	5350.3
14	690.5	6419.6	6152.8
15	759.5	7703.5	7075.7
16	835.4	9244.2	8137.1
17	919.0	11093.1	9357.6
18	1010.9	13311.7	10761.3
19	1112.0	15974.0	12375.5
20	1223.2	19168.8	14231.8

Table 4. V_{jm}^{old}

Volume of initial existing unit *m* in stage *j* in 1000L

m\j	1	2	3	4
1	1	4	3	3
2	-	-	3	-

Table 5. T_{ij}

The process time of product *i* in stage *j* in h

i\j	1	2	3	4
1	3.73	288	336	2.08
2	3.73	216	216	2.08
3	3.73	168	120	2.08

Table 6. S_{ij}

Size factor for product *i* in stage *j* in l/kg

i\j	1	2	3	4
1	0.3	11	11	5.76
2	0.3	11	11	5.76
3	0.3	11	11	5.76

These results are taking into consideration subproblems of five time periods. Note how the total number of variables and equations does not change. The number of discrete variables being solved is also the same; however, the effect in computational time is quite considerable. The number of binary variables being solved is the same because although the number of time periods being solved by the detailed problem increases, the binary variables for the time periods already covered by DDD are fixed, leaving the number of binary variables being solved by DDD equal to those present in five time periods only.

Table 7. Results and statistics for example

Problem			Binary Variables	Continuous Variables	Equations	CPU sec.	Objective value $
Detailed problem			8,840	20,883	27,798	***	***
Relaxed problem			8840R	20,883	27,798	251.39	524.07
Rolling Horizon Algorithm	subproblem 1	DDD 5 periods RDD 15 periods	2,240 (6600R)	20883	27,798	937.21	1801.74
	subproblem 2	DDD 10 periods RDD 10 periods	2,240 (4,400R)	20883	27,798	1915.78	1801.74
	subproblem 3	DDD 15 periods RDD 5 periods	2,240 (2,200R)	20883	27,798	6928.66	2147.0
	subproblem 4	DDD 20 periods 15 periods fixed	2240	20883	27,798	7463.12	3526.60
					FINAL	**17143.4**	**3526.6**

*** No solution was found.

The third line in Table 6 corresponds to subproblem 1, where the first 5 time periods are solved by DDD and the rest are relaxed. For subproblem 2, the binary variables for the first 5 time periods are fixed, model DDD is run throughout period 10, and periods 11 through 20 are relaxed. For subproblem 3, binary variables from periods 1 to 10 are fixed, the detail model is solved throughout period 15, and those corresponding to time periods 16 to 20 are relaxed. Finally in the last subproblem, subproblem 4, the binary variables from time periods

1 to 15 are fixed to those obtained in previous sub problems, and the detailed model is solved for the complete time horizon.

With model DDD it was not possible to find a solution for a design horizon of 20 periods, with the rolling horizon algorithm a solution was found in less than 5 hours. However, optimality was lost, since this solution is an approximation, i.e., an upper bound to the optimal solution.

Although the optimal solution was not found by the detailed model for the complete planning horizon, it was possible to feed the solution found by the RH algorithm as a starting point for the model in full space, and find the optimal solution value of $3244.78 in a 440,190 CPU seconds. Although the computational times are not comparable, since the model had a good feasible starting point, the value of the solution gives us a measure of the optimality gap of the considered approach. When comparing the optimal solution to that found by the rolling horizon algorithm, we have an optimality gap of 8.68%.

5. CONCLUSIONS

A multiperiod capacity expansion or multiperiod retrofit problem is a better form to approach design and investment decisions compared to the typical retrofit problem where just one period is taken into account. This however, increases the size and complexity of the problem enormously. To try to mitigate this effect, a disjunctive model is proposed, obtained by transforming the original problem using the convex hull relaxation over a disjunctive set. The disjunctive model proved to improve results when compared to the same problem modeled in full space. The disjunctive model, however, could not find solution for a horizon of 20 time periods which was of interest, to address this problem, a rolling horizon algorithm was proposed.

The proposed rolling horizon algorithm has the benefit that it can use the information of the demands for the entire design horizon. This is convenient since it allows the model to anticipate future demands and use that information when deciding both the amount of volume and when to install new equipment. This formulation considers all feasible options for this kind of problem.

The RHA was illustrated in a case study with 20 time periods. It was found the reported solution had an optimality gap of 8.6%, which is more than reasonable for industry standards. Furthermore, it was illustrated how this solution could be used as a starting solution for the direct solution method for the DDD model, finding a true global optimal solution in this particular example. Even though this finding of a global optimum cannot be guaranteed for every possible instance, this solution strategy can certainly be applied to attempt to improve the solution found by the RHA.

As possible areas of opportunity for future work it would be worthy to consider solving to optimality the detailed model through a non commercial solver, this was not attempted due to time limitations.

Acknowledgements: The authors of this work would like to thank the Mexican Council for Science and Technology (CONACYT) and UANL´s Scientific and Technological Research Support Program for their financial supports in this research.

REFERENCES

Barbosa-Póvoa, A.P. (2007). A critical review on the design and retrofit of batch plants. *Computers and Chemical Engineering*, 31(7):833-855.

Beraldi, P., Ghiaji, G., Grieco, A., and Guerriero, E. (2008). Rolling-horizon and fix-and-relax heuristics for the parallel machine lot sizing and scheduling problem with sequence dependent set-up costs. *Computers and Operations Research*, 35(11):3644-3656.

Caballero, J.A., Milán-Yañez, D., and Grossmann, I.E. (2005). Optimal synthesis of distillation columns: Integration of process simulators in a disjunctive programming environment. *Computer Aided Chemical Engineering*, 20:715-720

Dimitriadis, A.V., Shah, N., and Pantelides, C.C. (1997). RTN-based rolling horizon algorithms for medium term scheduling of multipurpose plants. *Computers and Chemical Engineering*, 21(s1):s1061-s1066.

Erdirik-Dogan, M., and Grossmann, I.E. (2007a) Planning models for parallel batch reactors with sequence-dependent changeovers. *AIChE Journa,l* 53(9):2284-2300.

Erdirik-Dogan, M., and Grossmann, I.E. (2007b). A decomposition method for the simultaneous planning and scheduling of single-stage continuous multiproduct plants. *Industrial and Engineering Chemistry Research,* 46(15):5250.

Fletcher, R., Hall, J.A., and Johns, W.R. (1991) Flexible retrofit design of multiproduct batch plants. *Computers and Chemical Engineering*, 15(12):843-852.

Grossmann, I.E. (2004). Advances in logic-based optimization approaches to process integration and supply chain management. In M.A. Galan and E. Del Valle (editors) *Chemical Engineering: Trends and Developments*. Wiley, Weinheim, Germany.

Lee, S., and Grossmann, I.E. (2003). Global optimization of nonlinear generalized disjunctive programming with bilinear equality constraints: Applications to process networks. *Computers and Chemical Engineering*, 27(11):1557-1575.

Macchietto, S. (2005). Integrated Batch Processing: A model for advanced manufacturing. In *Proceedings of the 2005 APACT Conference,* pp. 20-22. Birmingham, United Kingdom.

Montagna, J.M. (2003). Optimal retrofit of multiproduct batch plants. *Computers and Chemical Engineering*, 27(8-9):1277-1290.

Moreno, M.S., Montagna, J.M., and Iribarren, O.A. (2007). Multiperiod optimization for the design and planning of multiproduct batch plants. *Computers and Chemical Engineering*, 31(9):1159-1173.

Mussati, S.F., Barttfeld, M., Aguirre, P.A., and Scenna, N.J. (2008). A disjunctive programming model for superstructure optimization of power desalting plants. *Desalination*, 222(1-3):457-465.

Oldenburg, J., and Marquardt W. (2008). Disjunctive modeling for optimal control of hybrid systems. *Computers and Chemical Engineering*, 32(10):2346-2364.

Pinto, T., Barbosa-Póvoa, F.D., and Novais, A.Q. (2005). Optimal design of batch plants with a periodic mode of operation. *Computers and Chemical Engineering*, 29(6):1293-1303.

Raman, R., and Grossmann, I.E. (1994). Modeling and computational techniques for logic based integer programming. *Computers and Chemical Engineering*, 18(7):563-578.

Van den Heever, S., Grossmann, I.E., and Vasantharanjan, S. (2000). Integrating complex economic objective with the design and planning of offshore oilfield infrastructures. *Computers and Chemical Engineering*, 24(2-7):1049-1055.

Vaselenak, J.A., Grossmann, I.E., and Westerberg, A.W. (1987). Optimal retrofit design in multiproduct batch plants. *Industrial and Engineering Chemistry Research*, 26(4):718-726.