Dinkelbach's Algorithm as an Efficient Method for Solving a Class of MINLP Models for Large-Scale Cyclic Scheduling Problems

Fengqi You,[1] Pedro M. Castro,[2] Ignacio E. Grossmann[1*]

[1] Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213

[2] Departamento de Modelação e Simulação de Processos, INETI, 1649-038 Lisboa, Portugal

January, 2009

Abstract

In this paper we consider the solution methods for mixed-integer linear fractional programming (MILFP) models, which arise in cyclic process scheduling problems. We first discuss convexity properties of MILFP problems, and then investigate the capability of solving MILFP problems with MINLP methods. Dinkelbach's algorithm is introduced as an efficient method for solving large scale MILFP problems for which its optimality and convergence properties are established. Extensive computational examples are presented to compare Dinkelbach's algorithm with various MINLP methods. To illustrate the applications of this algorithm, we consider industrial cyclic scheduling problems for a reaction-separation network and a tissue paper mill with byproduct recycling. These problems are formulated as MILFP models based on a continuous time Resource-Task Network (RTN). The results show that orders of magnitude reduction in CPU times can be achieved when using this algorithm compared to solving the problems with commercial MINLP solvers.

Keywords: Cyclic Scheduling; Mixed-Integer Linear Fractional Programming; Mixed-integer Nonlinear Programming; Dinkelbach's Algorithm; Resource-Task Network;

[*] To whom all correspondence should be addressed. E-mail: grossmann@cmu.edu

1. Introduction

Cyclic scheduling is appropriate wherever product demands are stable over extended periods of time. In such cases, cyclic scheduling is usually implemented to find the best resource assignments as well as tasks sequencing, and then repeat such production pattern over an extended period of time (Sahinidis & Grossmann, 1991; Pinto & Grossmann, 1994; Castro and co-workers, 2003, 2005, 2009; Pochet & Warichet, 2008). Cyclic scheduling problems can be encountered in refinery operation (Pinto et al., 2000), chemical manufacturing (Sahinidis & Grossmann, 1991), pulp mills (Castro et al., 2003) and paper production (Castro et al., 2009). If continuous-time formulations are used, the objective function typically involves the maximization of a certain economic indicator (e.g. revenue, profit) divided by the cycle time, which is a model variable. If there are linear constraints and the objective function is given by the ratio of linear terms divided by the cycle time, the resulting mathematical problem is a special type of mixed-integer nonlinear program (MINLP) that is known as a mixed-integer linear fractional program (MILFP).

An MILFP includes both continuous and discrete variables, has an objective function in general form as the ratio of two linear functions and all the constraints are linear. Mathematically, an MILFP can be formulated as following problem (P):

$$\text{(P)} \quad \max \quad \frac{a_0 + \sum_{i \in I} a_i x_i}{b_0 + \sum_{i \in I} b_i x_i}$$

$$\text{s.t.} \quad c_{0j} + \sum_{i \in I} c_{ij} x_i = 0 , \quad \forall j$$

$$x_i \geq 0 , \ \forall i \in I_1 \ \text{ and } \ x_i \in \{0,1\} , \ \forall i \in I_2 \ \text{ and } \ I_1 \cup I_2 = I$$

where it is assumed that $b_0 + \sum_{i \in I} b_i x_i > 0$ for all feasible solutions and where all inequalities are converted into equalities through the use of slack variables. Due to the nonlinearities in the objective function and the combinatorial nature because of the presence of 0-1 variables, the

MILFP problems can become computationally intractable, especially for large scale instances with more than hundreds of binary variables.

Since (P) has both continuous and discrete variables, and it includes a nonlinear objective function with nonconvexities, it is also a mixed-integer nonlinear programming (MINLP) problem. Thus, optimization methods for MINLP problems can also be utilized for MILFP problems. Global optimization methods for MINLP problems include the branch and reduce method (Ryoo & Sahinidis, 1996; Tawarmalani & Sahinidis, 2004), the α-BB method (Adjiman, et al., 2000), the spatial branch and bound search method for bilinear and linear fractional terms (Quesada & Grossmann, 1995; Smith & Pantelides, 1999) and the outer-approximation method by Kesavan et al. (2000). All these methods rely on a branch and bound solution procedure. The difference among these methods lies on the definition of the convex envelopes for computing the lower bound, and on how to perform the branching on the discrete and continuous variables.

In addition to global optimization methods, MINLP methods that rely on convexity assumptions can also be utilized. Although global optimality cannot be guaranteed, these methods can usually obtain optimal or near optimal solutions for MILFP problems much faster than global optimizers. These MINLP methods include the branch & bound method (Leyffer, 2001), generalized Benders decomposition (Geoffrion, 1972), outer-approximation (Duran & Grossmann, 1986; Viswanathan & Grossmann, 1990), LP/NLP based branch and bound (Quesada & Grossmann, 1992), and extended cutting plane Method (Westerlund and Pettersson, 1995). A number of computer codes are available that implement these methods. The program DICOPT is an MINLP solver that is based on the Outer-Approximation algorithm, and is available in the modeling system GAMS (Brooke, 1998). The code α-ECP implements the extended cutting plane method by Westerlund and Pettersson (1995, 2002). Codes that implement the branch-and-bound method include the code MINLP_BB (Leyffer, 2001)

available in AMPL, and the program SBB which is also available in GAMS. Recently, the open source MINLP solver Bonmin (Bonami, et al. 2008), which is part of the COIN-OR project, implements an extension of the branch-and-cut outer-approximation algorithm that was proposed by Quesada and Grossmann (1992), as well as the branch and bound and outer-approximation method.

The rest part of this paper is organized as follows. In Section 2, we discuss the special properties of solving MILFP problems with convex MINLP methods. The Dinkelbach's algorithm is introduced and discussed in Section 3. In Section 4 and Section 5, we present the numerical examples and cases studies. Section 6 concludes this paper.

2. MINLP Methods for MILFP

Due to the special structure of MILFP problems, we can obtain global optimal solutions by finding the local optima with certain MINLP methods. Before addressing the solution methods, let us first consider problem (RP), which is the nonlinear relaxation of (P) with all the binary variables $x_i \in \{0,1\}$, $\forall i \in I_2$ relaxed as continuous variables $0 \le x_i \le 1$, $\forall i \in I_2$.

$$\text{(RP)} \quad \max \ \frac{a_0 + \sum_{i \in I} a_i x_i}{b_0 + \sum_{i \in I} b_i x_i}$$

$$\text{s.t.} \quad c_{0j} + \sum_{i \in I} c_{ij} x_i = 0, \quad \forall j$$

$$b_0 + \sum_{i \in I} b_i x_i > 0$$

$$x_i \ge 0, \ \forall i \in I_1 \ \text{and} \ 0 \le x_i \le 1, \ \forall i \in I_2 \ \text{and} \ I_1 \cup I_2 = I$$

It is easy to see that (RP) is a linear fractional program (LFP), in which the objective function is the ratio of two linear functions, all the constraints are linear and all the variables are continuous. The LFP problems are well studied and some of its important properties will assist the solution of MILFP with MINLP methods.

Lemma 1: Let $Q(x) = \left(a_0 + \sum_{i \in I} a_i x_i\right) \Big/ \left(b_0 + \sum_{i \in I} b_i x_i\right)$, and let F be a convex set such that $b_0 + \sum_{i \in I} b_i x_i \neq 0$ over F. Then, $Q(x)$ is both pseudoconvex and pseudoconcave over F.

Proof: See Chapter 11.4, "Linear Fractional Programming" in Bazaraa et al. (2004).

Lemma 2: $Q(x)$ is strictly quasiconvex and strictly quasiconcave over F.

Proof: Based on Lemma 1 that $Q(x)$ is pseudoconvex and pseudoconcave, we can conclude that $Q(x)$ is also strictly quasiconvex and strictly quasiconcave over F (Bazaraa et al., 2004).

Lemma 3: Every local optimum of $Q(x)$ over F is also its global solution.

Proof: Lemma 2 and the properties of strictly quasiconvex and strictly quasiconcave functions (Bazaraa et al., 2004) lead to Lemma 3.

Based on Lemma 3, we can conclude that the local maximum obtained by solving the LFP problem (RP) with any nonlinear programming (NLP) solver is also the global maximum of (RP). Furthermore, we can obtain the global optimal solution of the MILFP problem (P) with an MINLP method that can handle pseudoconvex/pseudoconcave objective function (eg. branch-and-bound, α-ECP methods).

Lemma 4: A global optimal solution of the MILFP problem (P) can be obtained by solving it with MINLP methods that can handle pseudoconvex/pseudoconcave objective function (eg. branch-and-bound, α-ECP).

Proof: Based on Lemma 3, the NLP relaxation of the MILFP problem (P) or an NLP subproblem with a subset of fixed 0-1 variables can be solved to *global* optimality by using a local NLP solver. Since the branch-and-bound method relies on solving a sequence of relaxed

NLP subproblems, global optimality can be guaranteed with that method. The α-ECP method by definition can handle pseudoconvex/pseudoconcave functions, and therefore it can also guarantee the global optimality.

Lemma 4 allows us to obtain global optimal solutions of MILFP problems with convex MINLP solvers, such as SBB and α-ECP, which are potentially faster than using a global optimizer. Note that the Generalized Benders Decomposition (Geoffrion, 1972) algorithm and Outer-Approximation (Duran & Grossmann, 1986; Viswanathan & Grossmann, 1990) method do not guarantee global optimality, since in both cases the lower/upper bounds predicted by the master problem rely on the assumption that the functions are convex.

Besides the MINLP methods, an alternative to solve the MILFP problems is to use Dinkelbach's algorithm by successively solving a number of mixed-integer linear programming (MILP) problems.

3. Dinkelbach's Algorithm for MILFP

By exploiting the relationship between nonlinear fractional programming and nonlinear parametric programming, Dinkelbach (1967) developed an algorithm to solve convex nonlinear fractional programming problems (without discrete variables) by successive solving a sequence of simplified NLP problems. As an extension, Dinkelbach's algorithm has recently been implemented to solve the MILFP problems (Bradley and Arntzen, 1999; Pochet & Warichet, 2008), but the optimality and convergence properties of using Dinkelbach's algorithm to solve MILFP problems have not been fully addressed (Warichet, 2007). In this section, we first introduce the algorithmic scheme for solving MILFP problems with Dinkelbach's algorithm, and then show that similar optimality and convergence properties (such as superlinear convergence rate) of using Dinkelbach's algorithm for convex nonlinear fractional programming problems (Dinkelbach, 1967; Schaible, 1976) also exist in using this

algorithm for MILFP problems.

Consider $N(x) = a_0 + \sum_{i \in I} a_i x_i$, $D(x) = b_0 + \sum_{i \in I} b_i x_i$, $Q(x) = N(x) / D(x)$ and the feasible region of (P) as $S : \{D(x) > 0$ and $c_{0j} + \sum_{i \in I} c_{ij} x_i = 0, \ \forall j$ and $x_i \geq 0$, $\forall i \in I_1$ and $x_i = 0, \ 1$, $\forall i \in I_2$ and $I_1 \cup I_2 = I\}$, the Dinkelbach's algorithm for MILFP problem (P): $\max\{Q(x) \mid x \in S\}$ is as follows:

Step 1: choose arbitrary $x_1 \in S$ and set $q_2 = \dfrac{N(x_1)}{D(x_1)}$ (or set $q_2 = 0$), let $k = 2$.

Step 2: solve the MILP problem $F(q_k) = \max\{N(x) - q_k D(x) \mid x \in S\}$, and denote the optimal solution as x_k.

Step 3: If $F(q_k) < \delta$ (optimality tolerance), stop and output x_k as the optimal solution; If $F(q_k) \geq \delta$, let $q_{k+1} = \dfrac{N(x_k)}{D(x_k)}$ and go to *Step 2* to replace k with $k+1$ and q_k with q_{k+1}.

***Proposition 1* (Optimality):** $F(q^*) = \max\{N(x) - q^* D(x) \mid x \in S\} = 0$ if and only if $q^* = \dfrac{N(x^*)}{D(x^*)} = \max\{N(x) / D(x) \mid x \in S\}$.

Proof: See appendix.

***Proposition 2* (Convergence):** Dinkelbach's algorithm converges superlinearly for MILFP problem (P).

Proof: See appendix.

Based on Proposition 1 and Proposition 2, we can solve a sequence of MILP subproblems to obtain the global optimal solution of an MILFP problem. A major advantage of using this algorithm is that no NLP solver is required, and the memory usage during the computational process tends to be rather small compared with using MINLP methods, especially for large-scale MILFP problems. In the next two sections, we present computational results for solving MILFP problems with Dinkelbach's algorithm and various MINLP methods.

4. Computational Results

In order to illustrate the application of the proposed solution strategies, we present computational experiments for 20 large scale instances. The computational experiments are carried out on an Intel 3.2 GHz machine with 512 MB memory. The proposed solution procedure is coded in GAMS 22.8.1 (Brooke, 1998). The MILP problems in Dinkelbach's solution procedure are solved using CPLEX 10, the MINLP solvers include SBB (special branch-and-bound algorithm), DICOPT (outer-approximation algorithm), and α-ECP (α - extended cutting-plane method), and the global optimizer used in the computational experiments is BARON 8.1.4. The relative optimality tolerance is set to be 10^{-6}.

We solve the MILFP problems (CE) as follows:

$$\text{(CE)} \quad \max \quad \frac{a_0 + \sum_{i \in I} a1_i x_i + \sum_{j \in J} a2_j y_j}{b_0 + \sum_{i \in I} b1_i x_i + \sum_{j \in J} b2_j y_j}$$

$$\text{s.t.} \quad c_{0k} + \sum_{i \in I} c1_{ik} x_i + \sum_{j \in J} c2_{jk} y_j = 0, \qquad \forall k \in K$$

$$b_0 + \sum_{i \in I} b1_i x_i + \sum_{j \in J} b2_j y_j \geq 0.001$$

$$x_i \geq 0, \ \forall i \in I \ \text{ and } \ y_j \in \{0,1\}, \ \forall j \in J$$

which includes $|I|$ continuous variables, $|J|$ binary variables, and $|K|+1$ constraints/equations. The values of $|I|$, $|J|$ and $|K|$ range from 100 to 2,000. Given the large size of problems, the input data are generated randomly. a_0 is generated uniformly on U[-10, 10], b_0 is generated uniformly on U[-10, 10], $a1_i$ is generated uniformly on U[0, 10], $a2_j$ is generated uniformly on U[0, 10], $b1_i$ is generated uniformly on U[0, 10], $b2_j$ is

generated uniformly on $U[0, 10]$, c_{0k} is generated uniformly on $U[-15, 20]$, $c1_{ik}$ is generated uniformly on $U[-10, 10]$, $c2_{jk}$ is generated uniformly on $U[-20, 15]$.

For each instance, we solve Problem (CE) with Dinkelbach's algorithm using CPLEX (the initial value of q is set to 0), and MINLP solvers SBB, DICOPT, α-ECP, as well as the global optimizer BARON. The computational results are given in Table 1. The optimal solutions obtained with all the methods are the same and equal to the global maximum, although DICOPT does not guarantee global optimality. The CPU times vary from one approach to the other. Specifically, solving MILFP problems with Dinkelbach's algorithm requires up to 7 iterations and it is usually faster than using the global optimizer BARON and the MINLP solver α-ECP. Dinkelbach's algorithm usually requires slightly more computational times than SBB and DICOPT for medium instances, but usually less CPU time than these MINLP solvers for large scale instances. More importantly, for very large scale instances with up to thousands of constraints and variables (instances 18-20), DICOPT and SBB both exceed the memory limit while Dinkelbach's algorithm can still return global optimal solutions. This is due to the smaller memory requirement when solving MILP rather than MINLP problems. As for the comparison between the MINLP solvers and the global optimizer, SBB and DICOPT require similar computational resources and are both much faster than α-ECP and BARON, especially for large scale instances (instances 11-17). The computational studies suggest that SBB and DICOPT might be the most efficient solvers for medium size MILFP problems, and Dinkelbach's algorithm may have higher computational efficiency for very large scale instances. In all runs, DICOPT was able to find the global optimal solutions, despite the fact that the outer-approximation method cannot guarantee global optimality for pseudoconvex/pseudoconcave functions. The results also show that Dinkelbach's algorithm with CPLEX and the convex MINLP solvers (SBB, DICOPT and α-ECP) are far more efficient than global optimizer BARON for solving MILFP problems.

Table 1 Computational results for MILFP Problems with Dinkelbach's algorithm and MINLP methods

No.	Cont. Var. $\|I\|$	Disc. Var. $\|J\|$	Eqs. $\|K\|+1$	Dinkelbach's Algorithm (CPLEX)			SBB		DICOPT		α-ECP		BARON	
				Iter.	Obj.	CPU (s)	Obj.	CPU (s)	Obj.	CPU (s)	Obj.	CPU (s)	Obj.	CPU (s)
1	100	100	101	6	3.033	0.59	3.033	0.11	3.033	0.06	3.033	5.00	3.033	0.69
8	200	100	101	5	2.213	0.65	2.213	0.08	2.213	0.08	2.213	7.00	2.213	10.97
9	300	100	101	6	2.267	1.45	2.267	0.49	2.267	0.87	2.267	161.0	2.267	34.48
10	500	100	101	5	2.085	2.16	2.085	0.53	2.085	1.48	2.085	18.98	2.085	71.64
2	100	200	101	6	3.256	0.97	3.256	0.06	3.256	0.08	3.256	6.00	3.256	11.68
3	100	300	101	7	3.556	1.44	3.556	0.47	3.556	0.87	3.556	65.40	3.556	31.05
4	100	500	101	6	3.691	1.62	3.691	0.39	3.691	0.11	3.691	18.72	3.691	36.40
5	100	100	201	6	3.033	0.84	3.033	0.19	3.033	0.06	3.033	8.00	3.033	12.69
6	100	100	301	6	3.033	1.34	3.033	0.11	3.033	0.08	3.033	12.00	3.033	28.03
7	100	100	501	6	3.033	2.58	3.033	0.14	3.033	0.16	3.033	26.88	3.033	25.72
11	500	500	501	6	2.634	15.79	2.634	5.59	2.634	12.19	2.634	852.70	2.593~3.225*	3,600*
12	1,000	500	501	6	2.311	68.11	2.311	50.31	2.311	45.11	1.786*	3,600*	2.310~14.811*	3,600*
13	500	1,000	501	6	2.887	24.29	2.887	0.92	2.887	0.90	2.887	1,460	2.887~5.872*	3,600*
14	500	500	1,001	6	2.633	34.11	2.633	1.77	2.633	1.77	2.633	1,647	2.633~19.205*	3,600*
15	1,000	1,000	1,001	7	2.608	222.22	2.608	146.57	2.608	164.09	2.430*	3,600*	---	3,600**
16	2,000	1,000	1,001	6	2.310	63.58	2.310	839.28	2.310	468.67	---	3,600**	---	3,600**
17	1,000	2,000	1,001	6	3.085	69.92	3.085	963.50	3.085	888.97	---	3,600**	---	3,600**
18	1,000	1,000	2,001	7	2.603	1466.59	---**	>memo	---**	> memo	---	3,600**	---	3,600**
19	2,000	2,000	1,001	6	2.593	517.95	---**	>memo	---**	> memo	---	3,600**	---	3,600**
20	2,000	2,000	2,001	6	2.705	242.23	---**	>memo	---**	> memo	---	3,600**	---	3,600**

***: Suboptimal solutions (or lower and upper bounds for BARON) obtained after 1 hour (3,600 seconds)**

****: No solution was returned after 1 hour (3,600 seconds)**

>memo: terminates because memory upper limit exceed

5. Cyclic Scheduling Problems

Cyclic scheduling problems can be formulated as mixed-integer linear fractional programs (Bradley and Arntzen, 1999; Pochet & Warichet, 2008). In this section, we employ the Dinkelbach's algorithm to solve large scale MILFP models for cyclic production scheduling from two case studies when compared to the MINLP commercial solvers.

5.1. Case study 1: Cyclic Scheduling for a Reaction-Separation Network

The first case study deals with a variant of the most studied scheduling problem in the process systems engineering literature (Kondili et al., 1993). It involves a multipurpose pharmaceutical batch plant producing two products from three different raw-materials through various reaction and separation operations. Figure 1 shows the associated RTN (Pantelides, 1994), annotated with information on task duration and minimum and maximum task size. Notice that the duration of the reaction tasks is dependent on the type of equipment being used. The product values are equal to \$12,000/kg and \$10,000/kg, respectively, as in Schilling and Pantelides (1999).

The problem was first investigated by Schilling & Pantelides (1999) and more recently addressed by Castro et al. (2003) and Wu & Ierapetritou (2004). Different continuous-time multipurpose formulations can be used to solve the problem. The first two contributions relied on a single time grid to keep track of events taking place, while the latter is a unit-specific formulation. In this paper, we will be using the one proposed by Castro and co-workers (2003, 2005), which has been shown to be more efficient than the one by Schilling & Pantelides (1999), primarily due to the need of fewer event points to find the optimal solution. Bear in mind, however, that the formulation of Wu & Ierapetritou (2004) shows better performance. The small differences observed in the values of the reported solutions between the papers result from slightly different data.

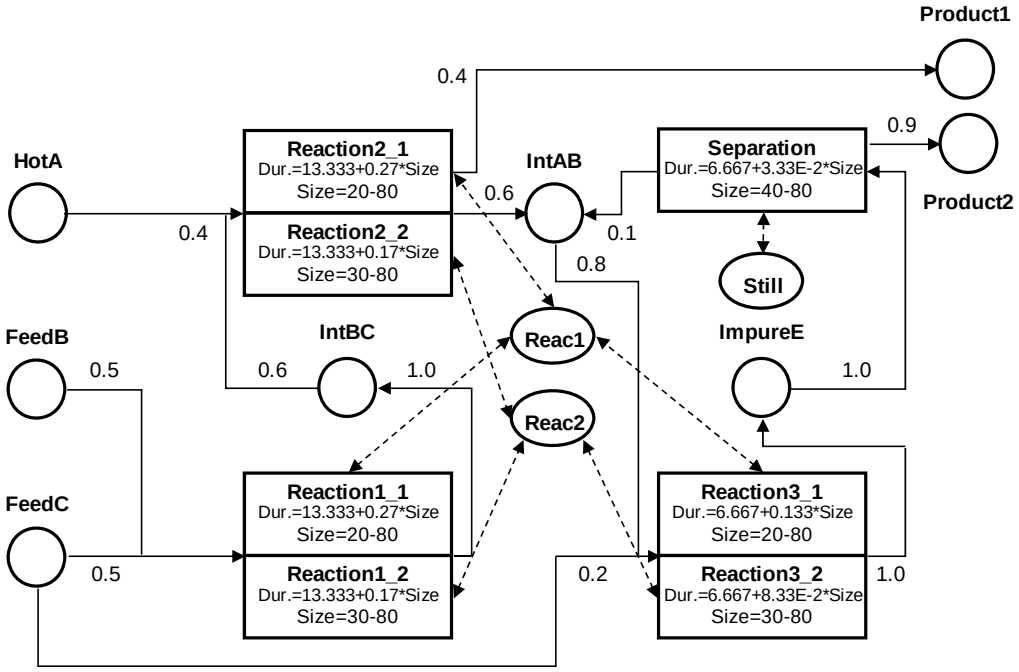

Figure 1 Resource-Task Network representation of case study 1. Duration in h, size in kg.

The scheduling formulation uses the RTN process representation, which relies on generic entities like resources (set R) and tasks (set I) to describe the process/product recipe. Through the use of the wrap-around operator Ω(·) (Shah et al., 1993),

$$\Omega(t) = \begin{cases} t, 1 \le t \le |T| \\ t + |T|, t < 1 \end{cases}$$

the cyclic scheduling formulation can be written in a very compact form, eqs 1-7. In addition, an anchoring constraint should be used for reducing solution degeneracy arising from cyclic scheduling permutations.

$$\max \quad \left(\sum_{r \in R^{FP}} v_r \cdot \Delta_r \right) \Big/ H \tag{1}$$

$$R_{r,t} = R_{r,\Omega(t-1)} + \sum_{i \in I} [\sum_{\substack{t' \in T \\ t < t' \le \Delta t + t \vee t' \le \Delta t + t - |T|}} (\mu_{r,i} \overline{N}_{i,t,t'} + \nu_{r,i} \overline{\xi}_{i,t,t'}) + \sum_{\substack{t' \in T \\ t - \Delta t \le t' < t \vee t' \ge t - \Delta t + |T|}} (\overline{\mu}_{r,i} \overline{N}_{i,t',t} + \overline{\nu}_{r,i} \overline{\xi}_{i,t',t})] + (\Delta_r \big|_{r \in R^{RM}} - \Delta_r \big|_{r \in R^{FP}}) \Big|_{t=1} \quad \forall r \in R, t \in T \tag{2}$$

$$R_{r,|T|} = 1 + \sum_{i \in I} \sum_{t \in T} (\sum_{\substack{t' \in T \\ t < t' \le \Delta t + t \vee t' \le \Delta t + t - |T|}} \mu_{r,i} \overline{N}_{i,t,t'} + \sum_{\substack{t' \in T \\ t < t' \le \Delta t + t}} \overline{\mu}_{r,i} \overline{N}_{i,t,t'}) \quad \forall r \in R^{EQ} \tag{3}$$

$$\overline{N}_{i,t,t'} \sum_{r \in R^{EQ}} \overline{\mu}_{r,i} V_r^{\min} \le \overline{\xi}_{i,t,t'} \le \overline{N}_{i,t,t'} \sum_{r \in R^{EQ}} \overline{\mu}_{r,i} V_r^{\max} \ \forall i \in I, t \in T, t' \in T, t < t' \le \Delta t + t - |T| \quad (4)$$

$$T_{t'} - T_t \ge \sum_{i \in I} (\alpha_i \overline{N}_{i,t,t'} + \beta_i \overline{\xi}_{i,t,t'}) \ \forall i \in I, t \in T, t' \in T, t < t' \le \Delta t + t \quad (5)$$

$$T_t - T_{t'} \le H - \sum_{i \in I} (\alpha_i \overline{N}_{i,t,t'} + \beta_i \overline{\xi}_{i,t,t'}) \ \forall i \in I, t \in T, t' \in T, t' \le \Delta t + t - |T| \quad (6)$$

$$H^{\min} \le H \le H^{\max}; T_1 = 0; 0 \le T_t \le H^{\max}; \overline{N}_{i,t,t'} \in \{0,1\}; \overline{\xi}_{i,t,t'}, R_{r,t}, \Delta_r \ge 0 \quad (7)$$

The binary variables $\overline{N}_{i,t,t'}$ identify the start of task *i* at event point *t* and ending at event point *t*', while the continuous variables $\overline{\xi}_{i,t,t'}$ give the amount processed, which must lie within the equipment unit's ($r \in \mathrm{R}^{\mathrm{EQ}}$) minimum ($V_r^{\min}$) and maximum capacities ($V_r^{\max}$), see eq 4. In this problem, the amount handled by the task affects its duration, with parameters α_i (h) and β_i (h/kg), giving the fixed and batch size dependent processing times, respectively. The excess amount of resource *r* at event point *t* is given by $R_{r,t}$. Raw-materials ($r \in \mathrm{R}^{\mathrm{RM}}$) and final products ($r \in \mathrm{R}^{\mathrm{FP}}$) have net consumption/production over the cycle, with variables Δ_r holding the exact amounts. The latter have a certain market value v_r ($/kg), and the objective is to maximize the revenue while minimizing the cycle time, *H* (h), i.e. maximizing the hourly revenue ($/h), see eq 1. Variable T_t holds the time of event point *t* relative to the start of the cycle. Finally, structural parameters $\mu_{r,i}, \overline{\mu}_{r,i}$ contain the consumption/production of resource *r* at the start/end (over bar) of the task that is independent on the amount processed. These are linked to the dashed arrows in Figure 1. Parameters $v_{r,i}$ and $\overline{v}_{r,i}$ are used for the batch size dependent values and are linked to the solid arrows.

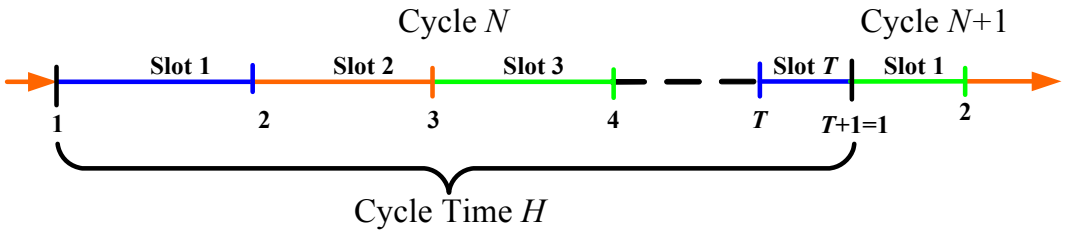

Figure 2 Event points and time grids in continuous-time formulation

Table 2 Results for Case Study 1 with Dinkelbach's algorithm and MINLP methods

$\|T\|$	Δt	H^{min}	H^{max}	Disc. Var.	Cont. Var.	Eqs.	Obj. (k$/h)	H(h)	CPU (s)	Solver
							28.768	36.866	11.97 (3 iter.)	Dinkelbach
							28.768	36.866	1448.04	SBB
4	2	20	40	56	110	216	28.768	36.866	11.79	DICOPT
							28.768	36.866	87.80	α-ECP
							28.768	36.866	455.54	BARON
							32.209	63.708	24.29 (4 iter.)	Dinkelbach
							32.209	63.708	1516.24	SBB
4	2	40	70	56	110	216	26.514	40	10.34	DICOPT
							32.209	63.708	37.18	α-ECP
							32.209	63.708	2986.54	BARON
							32.345	63.601	30.20 (3 iter.)	Dinkelbach
							32.345	63.601	6962.72	SBB
4	3	40	70	84	138	300	26.514	40	36.09	DICOPT
							32.345	63.601	504.06	α-ECP
							32.345	63.601	3059.87	BARON
							32.757	62.801	287.50 (3 iter.)	Dinkelbach
							32.209~60.617*	63.708	7,200*	SBB
5	3	40	70	105	172	374	26.514	40	94.25	DICOPT
							32.757	62.801	469.75	α-ECP
							32.757~64.806*	62.801*	7,200*	BARON
							32.605	97.585	288.67 (3 iter.)	Dinkelbach
							32.604~41.384*	97.585*	7,200*	SBB
5	3	70	100	105	172	374	30.302	70	284.70	DICOPT
							32.605	97.585	492.34	α-ECP
							32.577~46.211*	97.812*	7,200*	BARON
							33.396	105.677	857.84 (3 iter.)	Dinkelbach
							31.871~41.929*	118.192*	7,200*	SBB
6	3	100	140	126	204	448	33.396	105.677	5,554.94	DICOPT
							33.396	105.677	1654.66	α-ECP
							33.396~41.644*	105.677*	7,200*	BARON
							33.396	105.677	995.20 (3 iter.)	Dinkelbach
							31.254~44.470*	120.00*	7,200*	SBB
6	4	100	140	168	246	574	---	---	7,200*	DICOPT
							28.893*	126.140*	7,200*	α-ECP
							33.396~41.578*	105.677*	7,200*	BARON
							34.108	103.47	2,298.72 (3 iter.)	Dinkelbach
							26.885~46.743*	120.00*	7,200*	SBB
7	4	100	140	196	286	670	---	---	7,200*	DICOPT
							24.295*	136.43*	7,200*	α-ECP
							34.108~45.537*	103.47*	7,200*	BARON

***: Suboptimal solutions (or lower and upper bounds for BARON and SBB) obtained after 2 hours (7,200 seconds)**

The drawback of a time grid formulation is that the solution returned depends on the number of event points, $|T|$ (as shown in Figure 2), and also on the number of intervals that a task can span, which is specified through parameter Δt. Only for a sufficiently large number of both parameters can we find the global optimal optimum. The solution also depends on the cycle time range $H \in [H^{min}; H^{max}]$ as illustrated by Wu & Ierapetritou (2004).

In this paper, however, the main goal is to compare the computational performance of various solvers/algorithms for MILFP problems, so we simply tested them for different values of these four parameters. We solve this problem using the same machine and software environment as introduced in Section 3 with Dinkelbach's algorithm (solver CPLEX), MINLP solvers SBB, DICOPT and α-ECP, as well as global optimizer BARON. Table 2 lists the results obtained.

As we can see, the problem sizes increase as the number of event points $|T|$ and the number of intervals that a task can span Δt increase. For the first three instances with $|T|=4$, the optimal solutions obtained by using Dinkelbach's algorithm are the same as the global optimal solutions obtained by using BARON, and Dinkelbach's algorithm requires much less computational time than BARON. Although the optimal solutions from SBB and α-ECP are also the same, these two solvers require longer CPU times than Dinkelbach's algorithm. DICOPT can obtain solutions faster than Dinkelbach's algorithm, but for the second and the third instances in Table 2, the solutions are not the global optimum. For those two instances with $|T|=5$, the computational results in Table 2 show that both SBB and BARON failed to converge in 2 hours, although suboptimal solutions closed to global maximum are returned. We can also see that α-ECP returned the same global optimal solutions as those obtained by Dinkelbach's algorithm, although α-ECP requires more CPU times. DICOPT is the fastest solver, but again returned suboptimal solutions for these two instances. It is interesting to note that for the second, third and fourth instances, if the lower bound of the cycle time is reduced

from 40 to 20, then DICOPT does converge to the global optimum, because the lower bound predicted by the MILP master problem is a valid bound for these new instances. For large scale instances with $|T|=6$ and $|T|=7$, most of the MINLP solvers failed to converge in 2 hours (DICOPT and α-ECP can both solve the sixth instance), but Dinkelbach's algorithm can solve all the three instances within 1 hour. Overall, Dinkelbach's algorithm is clearly the best approach for solving large scale MILFP problems for this cyclic process scheduling problem. It requires much less computational times than most of the MINLP solvers and guarantees the global optimality of the solutions. Although DICOPT is a fast solver for this type of problems, it may return suboptimal solutions for some instances.

5.2. Case study 2: Cyclic Scheduling for a Tissue Paper Mill with Byproduct Recycling

The second case study is taken from Castro et al. (2009) and involves a continuous multiproduct tissue-paper plant. The cyclic scheduling problem, occurring at a Scandinavian tissue paper mill consists of maximizing the profit of the plant and finding the optimal cycle time while meeting minimum demands for all products. There are basically two production stages, each featuring two continuous lines in parallel, the stock preparation lines in the first stage, and the tissue-paper machines in the second stage, see Figure 3. In the first stage, the different raw-materials (ONP=old newspaper, MOW=mixed office waste, VF=virgin fiber) together with some recycled fiber (broke) are prepared for the tissue machines. Five intermediate products result from stock preparation, with two of them having dedicated stockyard piles of limited capacity. These are then processed in the tissue machines to get the five qualities, from least to most valuable product: P50, P60, P75, P80 and P85, where the number represents the brightness in %ISO.

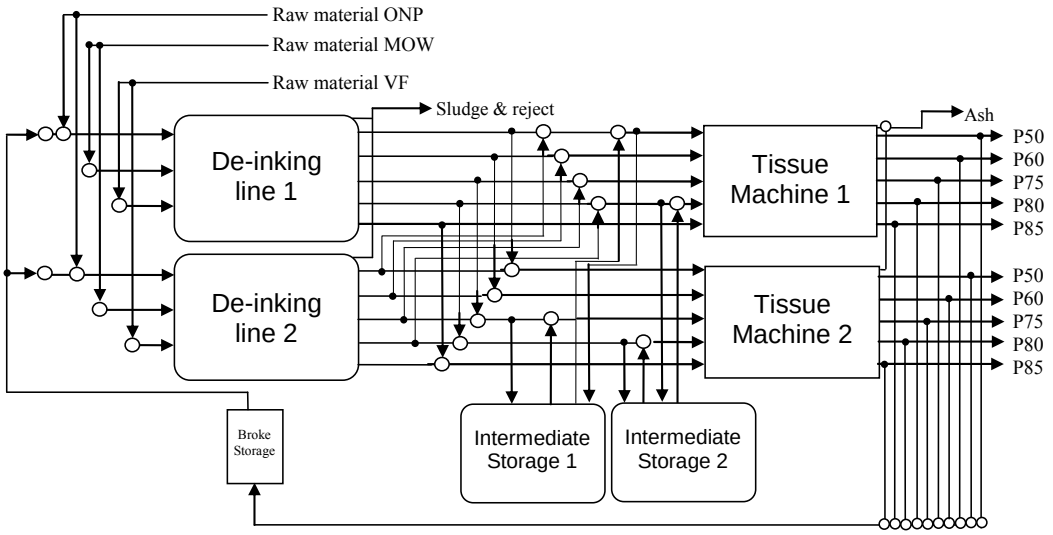

Figure 3. Flowsheet of the tissue mill.

In order to be more environmentally friendly and lower the operating costs, a proper broke recycling strategy should be implemented. Broke is mainly generated when processing fibers in the tissue machines and in the downstream converting lines. Despite the fact that the latter are not shown in the flowsheet, their fiber losses are accounted for in the tissue machines. Three scenarios were analyzed in Castro et al. (2009): (i) no recycling; (ii) incorporate all broke in the lowest quality products (P50 and P60); (iii) rather than mixing all broke, recycle it according to quality. More specifically, broke from P85 will partly replace VF, broke from P80, MOW, while broke from the remaining products will be mixed with ONP. The RTN corresponding to this option, the most cost efficient, is shown in Figure 4.

Similar to case study 1, we will be differentiating the resources. Equipment units (R^{EQ}) include the stock preparation lines (SP1-2), tissue machines (TM1-2), dewatering line that prepares the intermediates for storage (DW), and the two dedicated storage lines (L67, L80). The subset of raw-materials (R^{RM}) contains ONP, MOW and VF; the intermediates comprise ONP50, ONP60, ONP67, MOW80, VF85, S67 and S80; and the final products (R^{FP}) take in P50-P85. The elements of the subset of broke resources (R^{BK}) vary according to the recycling policy. As for the tasks, all are of the continuous type and are characterized by maximum

processing rate $\rho_i^{\max}$. A peculiar problem feature is that we cannot set beforehand the proportion of broke to be incorporated in a certain intermediate, since the amount produced will be determined as part of the optimization procedure and we want to use as much as possible. In other words, there are flexible proportions of input materials for the tasks executed in the stock preparation lines. Such variable recipe tasks (I^{VR}) will be subject to additional sets of constraints.

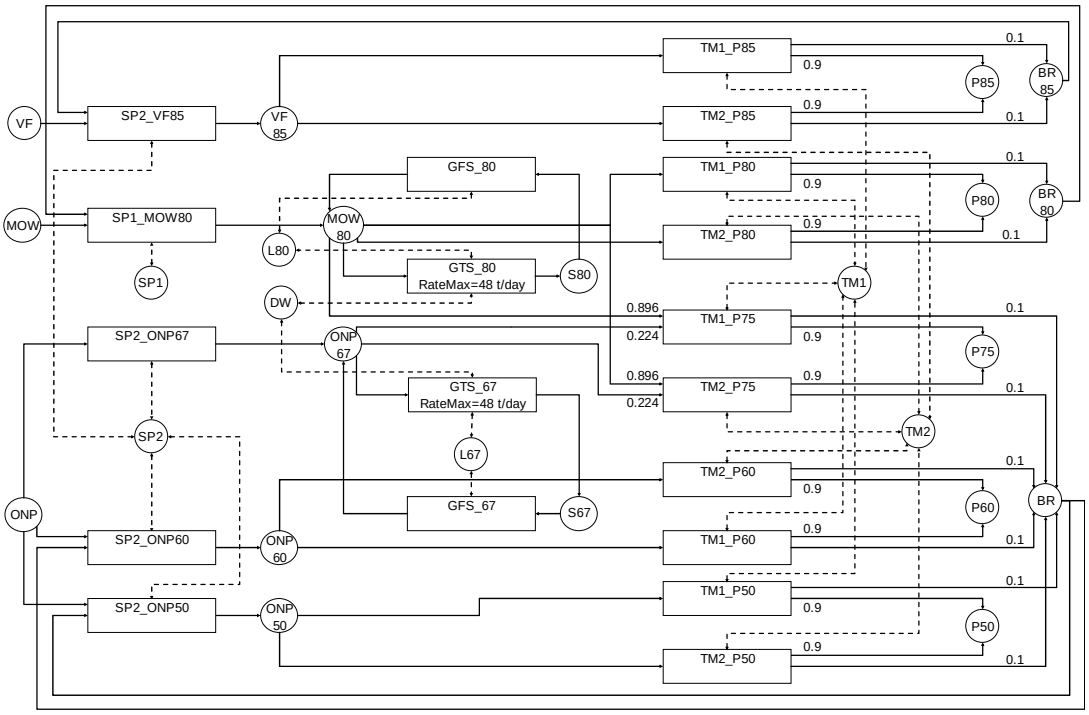

Figure 4. Resource-Task Network representation of best broke recycling policy.

In contrast to their batch counterparts, it can be assumed without loss of generality that continuous tasks last a single time interval. As a consequence, only one time index is needed in the task-related model variables. Binary variables $N_{i,t}$ assign the start of task i to event point t, while the total amount produced is given by $\xi_{i,t}$. For variable recipe tasks, the part due to resource r is accounted for by $\xi^*_{r,i,t}$. In order to identify changeovers in the tissue

machines, variables $X_{r,i,i',t}$ are employed. The remaining variables were already presented in case 1.

The model constraints are given next. The objective function is the maximization of the annual profit, given in relative monetary units (r.m.u./year), eq 8. The first term includes the materials value, which is negative for raw-materials and broke resources and positive for final products. The second term includes the total operating cost and the third, the total changeover cost. Parameter *wd* gives the number of working days per year. Notice that the numerator is linear while the denominator, the cycle time *H*, is a variable, making this model an MILFP.

The other constraints are similar to case 1, with differences arising from handling continuous rather than batch tasks. In particular, note that structural parameters $\lambda_{r,i}$ and $\overline{\lambda}_{r,i}$ have replaced $\nu_{r,i}$ and $\overline{\nu}_{r,i}$, in the excess resource balances (compare eq 9 to eq 2). Furthermore, the upper bound on the amount processed is now determined by the product of the maximum processing rate and the upper bound on the cycle time (eq 12). In addition, the timing constraints, eq 17, are written for every equipment resource rather than every task, where the duration of the task is determined by dividing the amount by the maximum processing rate. Specific constraints for this example are given in eqs 13-16. The extent of a certain variable recipe task must be equal to the sum of the partial extents, where parameter ff_i (fiber factor) can be viewed as a yield. Eq 14 limits broke incorporation to 20% and eq 15 activates the changeover variables. Finally, eq 16 ensures that the product demands are met.

$$\max \quad \left(\sum_{r\in R^{RM}\cup R^{BK}\cup R^{FP}} v_r \Delta_r - \sum_{i\in I}\sum_{t\in T} co_i \xi_{i,t} - \sum_{r\in R^{TM}}\sum_{i\in I}\sum_{i'\in I}\sum_{t\in T} ccg_{i,i'} X_{r,i,i',t} \right) \cdot wd \,/\, H \tag{8}$$

$$R_{r,t} = R_{r,\Omega(t-1)} + \sum_{i\in I}(\mu_{r,i} N_{i,t} + \overline{\mu}_{r,i} N_{i,\Omega(t-1)} + \lambda_{r,i}\xi_{i,\Omega(t-1)} + \overline{\lambda}_{r,i}\xi^*_{r,i,\Omega(t-1)}) + (\Delta_r\big|_{r\in R^{RM}} - \Delta_r\big|_{r\in R^{FP}\cup R^{BK}})\Big|_{t=1} \quad \forall r\in R, t\in T \tag{9}$$

$$R_{r,|T|} = 1 + \sum_{i\in I}\mu_{r,i} N_{i,|T|} \quad \forall r \in R^{EQ} \tag{10}$$

$$R_{r,t} \le R_r^{\max} \ \forall r \in R^{IT}, t \in T \tag{11}$$

$$\xi_{i,t} \le \rho_i^{\max} \cdot H^{\max} \cdot N_{i,t} \ \forall i \in I, t \in T \tag{12}$$

$$ff_i \xi_{i,t} = -\sum_{r \in R^{RM} \cup R^{BK}} \overline{\lambda}_{i,r} \xi^*_{r,i,t} \ \forall i \in I^{VR}, t \in T \tag{13}$$

$$0.2\, ff_i \xi_{i,t} \ge -\sum_{r \in R^{BK}} \overline{\lambda}_{i,r} \xi^*_{r,i,t} \ \forall i \in I^{VR}, t \in T \tag{14}$$

$$X_{r,i,i',t} \ge N_{i',t} + N_{i,\Omega(t-1)} - 1 \ \forall r \in R^{TM}, i, i' \in I, t \in T \tag{15}$$

$$d_r^{\min} \cdot H \le wd \cdot \Delta_r \le d_r^{\max} \cdot H \ \forall r \in R^{FP} \tag{16}$$

$$T_{t+1}\big|_{t \ne |T|} + H\big|_{t=|T|} - T_t \ge \sum_{i \in I} \overline{\mu}_{r,i} \xi_{i,t} / \rho_i^{\max} \ \forall r \in R^{EQ}, t \in T \tag{17}$$

$$H^{\min} \le H \le H^{\max}; T_1 = 0;\ 0 \le T_t \le H^{\max}; N_{i,t} \in \{0,1\}; 0 \le X_{r,i,i',t} \le 1; \xi_{i,t}, \xi^*_{r,i,t}, R_{r,t}, \Delta_r \ge 0 \tag{18}$$

We consider four instances, total number of event points |T|=4, |T|=5, |T|=6 and |T|=7 (Δt=1 for all the instances). We solve this problem using the same machine and software environment as introduced in Section 3 with Dinkelbach's algorithm (solver CPLEX). Given the problem sizes of these instances and the fact from the previous case study that global optimizer BARON usually requires long CPU times for large scale instances, we only compare the performance of Dinkelbach's algorithm with those of the MINLP solvers SBB, DICOPT and α-ECP. The results are listed in Table 3.

From Table 3, we can see that the same optimal solutions are obtained by using Dinkelbach's algorithm and MINLP solvers for the instances that MINLP solvers can return optimal solution within 20 hours. Similar to what we observed in the previous case study, Dinkelbach's algorithm requires much less computational times than the MINLP solvers, especially SBB. For the large scale instances with |T|=6 and |T|=7, only Dinkelbach's algorithm and DICOPT can return optimal solutions, while SBB and α-ECP failed to converge within 20 hours. For these two instances, Dinkelbach's algorithm still requires significantly less CPU time than DICOPT, although DICOPT does return global optimal solutions for these instances.

Notice that the cycle time is converging to its upper bound, indicating that the objective function would continue to increase. However, longer cycle times would also mean longer storage periods for the intermediates and higher degradation resulting from instance from direct sunlight, which is not desirable. Since the obtained optimal schedules are the same as in Castro et al. (2009), the readers can refer to this paper for the detailed solutions. The computational results of this case study suggest the advantage of using Dinkelbach's algorithm to solve MILFP models of cyclic scheduling problems.

Table 3 Results for Case Study 2 with Dinkelbach's algorithm and MINLP methods

$\|T\|$	H^{min}	H^{max}	Disc. Var.	Cont. Var.	Eqs.	Obj. (r.m.u./yr)	H(h)	CPU (s)	Solver
4	3	13.8	76	384	416	94.722	7.600	34.33 (4 iter.)	Dinkelbach
						94.722	7.600	1,238.84	SBB
						94.722	7.600	42.23	DICOPT
						94.722	7.600	42.97	α-ECP
5	3	13.8	95	478	516	94.868	8.371	578.65 (3 iter.)	Dinkelbach
						92.814*	5.318*	72,000*	SBB
						94.868	8.371	1,007.63	DICOPT
						94.868	8.371	1,405.31	α-ECP
6	3	13.8	570	616	216	95.248	13.80	1,967.92 (3 iter.)	Dinkelbach
						89.914*	3.226*	72,000*	SBB
						95.248	13.80	10,349.52	DICOPT
						91.280*	4.672*	72,000*	α-ECP
7	3	13.8	664	716	670	95.248	13.80	7,720.95 (3 iter.)	Dinkelbach
						69.758*	3.00*	72,000*	SBB
						95.248	13.80	41,660.20	DICOPT
						93.468*	5.804*	72,000*	α-ECP

***: Suboptimal solutions obtained after 20 hours (72,000 seconds)**

6. Conclusions

In this paper, we have discussed the convexity properties of MILFP problems and their NLP relaxation. The capability and optimality of solving MILFP problems with various MINLP methods has been addressed. We also show that Dinkelbach's algorithm, which is used to solve continuous fractional programming problems, can be an efficient method for

solving large scale MILFP problems. We have addressed the optimality and convergence issues of this algorithm, and proved that it also has superlinear convergence rate. Extensive computational studies were performed to demonstrate the efficiency of this algorithm. Case studies of cyclic scheduling problems for a reaction-separation network and for a tissue paper mill with byproduct recycling were also considered to illustrate the practical capability of this algorithm. The results show that Dinkelbach's algorithm required significantly less computational effort to solve MILFP test problems compared to commercial MINLP solvers, such as DICOPT (Outer-Approximation), SBB (special branch-and-bound), α-ECP (extended cutting plane method), and the global optimizer BARON (branch-and-reduce).

Acknowledgment

The authors acknowledge the financial support from the National Science Foundation under Grant No. OCI-0750826 and from the Luso-American Foundation.

Appendix

Let $N(x) = a_0 + \sum_{i \in I} a_i x_i$, $D(x) = b_0 + \sum_{i \in I} b_i x_i$, $F(q) = \max\{N(x) - qD(x) \mid x \in S\}$, $Q(x) = N(x) / D(x)$, and the feasible region be $S : \{D(x) > 0$ and $c_{0j} + \sum_{i \in I} c_{ij} x_i = 0, \ \forall j$ and $x_i \geq 0$, $\forall i \in I_1$ and $0 \leq x_i \leq 1$, $\forall i \in I_2$ and $I_1 \cup I_2 = I\}$

Before proving Proposition 1, first consider the following lemmas.

Lemma 4: The function $F(q) = \max\{N(x) - qD(x) \mid x \in S\}$ is convex.

Proof: Let x_t be the optimal solution that maximize $F(tq' + (1-t)q'')$ with $q' \neq q''$ and $0 \leq t \leq 1$. Thus, we have:

$$
\begin{aligned}
&F\left(tq'+(1-t)q''\right)\\
&=N(x_t)-\left(tq'+(1-t)q''\right)D(x_t)=t\left[N(x_t)-q'D(x_t)\right]+(1-t)\left[N(x_t)-q''D(x_t)\right]\\
&\le t\cdot\max\left\{N(x_t)-q'D(x_t)\mid x\in S\right\}+(1-t)\cdot\max\left\{N(x_t)-q''D(x_t)\mid x\in S\right\}\\
&=tF(q')+(1-t)F(q'')
\end{aligned}
$$

Lemma 5: For $q'<q''$, we have $F(q')>F(q'')$, i.e. $F(q)=\max\left\{N(x)-qD(x)\mid x\in S\right\}$ is strictly monotonic decreasing.

Proof: Let $x^*\in S$ be the optimal solution that maximize $F(q'')$. Assuming $D(x)>0$, we have:

$$
\begin{aligned}
&F(q'')\\
&=\max\left\{N(x)-q''D(x)\mid x\in S\right\}=N(x^*)-q''D(x^*)\\
&\le N(x^*)-q'D(x^*)=\max\left\{N(x)-q'D(x)\mid x\in S\right\}\\
&=F(q')
\end{aligned}
$$

Lemma 6: $F(q)=0$ has a unique solution.

Proof: It is easy to see that for $D(x)>0$, $\lim_{q\to-\infty}F(q)=+\infty$ and $\lim_{q\to+\infty}F(q)=-\infty$. Furthermore, based on Lemma 5, we can conclude that $F(q)=0$ has a unique solution.

Lemma 7: For any $x'\in S$, let $q'=\dfrac{N(x')}{D(x')}$. Then, we have $F(q')\ge 0$.

Proof: It is easy to see that $F(q')=\max\left\{N(x)-q'D(x)\mid x\in S\right\}\ge N(x')-q'D(x')=0$.

Proof of Proposition 1:

a) Let $x^*\in S$ be an optimal solution of $\max\left\{N(x)-q^*D(x)\mid x\in S\right\}$, so we have

$N(x^*) - q^* D(x^*) = 0$ and $N(x) - q^* D(x) \le N(x^*) - q^* D(x^*) = 0$, $\forall x \in S$.

Since $D(x) > 0$, we have $q^* = N(x^*)/D(x^*) \ge N(x)/D(x)$, $\forall x \in S$.

Thus, q^* is the maximum of $\max\{N(x) - qD(x) \mid x \in S\}$ and x^* is the optimal solution of $\max\{N(x)/D(x) \mid x \in S\}$.

b) Let x^* be an optimal solution of $\max\{N(x)/D(x) \mid x \in S\}$ and q^* be the optimal objective function value, so we have $q^* = N(x^*)/D(x^*) \ge N(x)/D(x)$, $\forall x \in S$.

Since $D(x) > 0$, we have $N(x) - q^* D(x) \le N(x^*) - q^* D(x^*) = 0$, $\forall x \in S$.

This implies that x^* is the optimal solution of $\max\{N(x) - q^* D(x) \mid x \in S\}$.

Before proving Proposition 2, consider the following lemmas.

Lemma 8: Let x' and x'' be the optimal solution of $F(q')$ and $F(q'')$. If $q' < q''$, $D(x') \ge D(x'')$.

Proof: Since x' and x'' be the optimal solution of $F(q')$ and $F(q'')$, we have:

$$N(x') - q' D(x') \ge N(x'') - q' D(x'')$$

$$N(x'') - q'' D(x'') \ge N(x') - q'' D(x')$$

Add the above two inequalities and rearrange both sides of the new one, we have:

$$(q'' - q') D(x') \ge (q'' - q') D(x'')$$

As $q'' > q'$, therefore we have $D(x') \ge D(x'')$.

Lemma 9: Let x' and x'' be the optimal solution of $F(q')$ and $F(q'')$. Then,

$$Q(x'') - Q(x') \geq \frac{F(q'')}{D(x'')} - \frac{F(q'')}{D(x')}.$$

Proof: As x'' is the optimal solution of $F(q'')$, we have:

$$N(x'') - q''D(x'') \geq N(x') - q''D(x')$$

Dividing both sides by $D(x') > 0$, we have $\frac{N(x'')}{D(x')} - \frac{q''D(x'')}{D(x')} \geq \frac{N(x')}{D(x')} - q''$.

Thus, we have

$$\begin{aligned} & Q(x'') - Q(x') \\ &= \frac{N(x'')}{D(x'')} - \frac{N(x')}{D(x')} \geq \frac{N(x'')}{D(x'')} - \frac{N(x'')}{D(x')} + q''\left[\frac{D(x'')}{D(x')} - \frac{D(x'')}{D(x'')}\right] = \left(-F(q'')\right)\left(\frac{1}{D(x')} - \frac{1}{D(x'')}\right) \\ &= \frac{F(q'')}{D(x'')} - \frac{F(q'')}{D(x')} \end{aligned}$$

Lemma 10: Let x' and x'' be the optimal solution of $F(q')$ and $F(q'')$. If $q' \leq q'' \leq q^*$,

Then, $Q(x') \leq Q(x'')$, where q^* satisfies $F(q^*) = 0$.

Proof: The proof readily follows from Lemmas 7, 8 and 9.

Lemma 11: Let x' and x'' be the optimal solution of $F(q')$ and $F(q'')$. Then,

$$Q(x'') - Q(x') \leq \left[-F(q'') + (q' - q'')D(x'')\right]\left(\frac{1}{D(x')} - \frac{1}{D(x'')}\right).$$

Proof: Similar to Lemma 9, considering the optimality condition for x',

We have $N(x') - q'D(x') \geq N(x'') - q'D(x'')$.

Dividing both sides by $D(x')$, we have $\frac{N(x')}{D(x')} - q' \geq \frac{N(x'')}{D(x')} - q'\frac{D(x'')}{D(x')}$.

Thus we have,

$$
\begin{aligned}
&Q(x'')-Q(x') \\
&\le \frac{N(x'')}{D(x'')}-\frac{N(x'')}{D(x')}-q'\left[\frac{D(x'')}{D(x')}-\frac{D(x'')}{D(x'')}\right]=\left(-N(x'')+q'D(x'')\right)\left(\frac{1}{D(x')}-\frac{1}{D(x'')}\right) \\
&=\left[-F(q'')+(q'-q'')D(x'')\right]\left(\frac{1}{D(x')}-\frac{1}{D(x'')}\right)
\end{aligned}
$$

Lemma 12: Let x' and x^* be the optimal solution of $F(q')$ and $F(q^*)$, where q^* satisfies $F(q^*)=0$. Then we have: $q^*-Q(x')\le(q^*-q')\left(1-\frac{D(x^*)}{D(x')}\right)$

Proof: Since q^* satisfies $F(q^*)=0$, we have $Q(x^*)=q^*$. Based on Lemma 11, it is easy to prove Lemma 12.

Proof of Proposition 2:

Dinkelbach's algorithm updates q_i with the previous value of $Q(x)$, i.e. $q_{i+1}=Q(x_i)$, where x_i is the optimal solution of $F(q_i)$. From Lemma 12, the algorithm converges with a rate of $\left(1-\frac{D(x^*)}{D(x_i)}\right)$. From Lemma 5, we have $\left(1-\frac{D(x^*)}{D(x_i)}\right)$ is nonincreasing. Therefore, the sequence $\{q_i\}$ obtained by Dinkelbach's algorithm converges superlinearly to q^* for each $q_i<q^*$.

References

1) Adjiman, C. S.; Androulakis, I. P.; Floudas, C. A. (2000). Global optimization of mixed-integer nonlinear problems. *AIChE Journal*, 16(9), 1769-1797.

2) Bazaraa, M. S.; Sherali, H. D.; Shetty, C. M. (2004). Nonlinear Programming: Theory and

Algorithms. Wiley, New York.

3) Bonami, P.; Biegler, L. T.; Conn, A. R.; Cornuéjols, G.; Grossmann, I. E.; Laird, C. D.; Lee, J.; Lodi, A.; Margot, F.; Sawaya, N; Waechter, A. (2008). An Algorithmic Framework for Convex Mixed Integer Nonlinear Programs. *Discrete Optimization*, 5(2), 186-204

4) Bradley, J. R.; Arntzen, B. C. (1999). The Simultaneous Planning of Production, Capacity, and Inventory in Seasonal Demand Environments. *Operations Research*, 47, (6), 795-806.

5) Brooke, A.; Kendrick, D.; Meeraus, A.; Raman, R. (1998). GAMS- A User's Manual. In GAMS Development Corp.

6) Castro, P.M., Barbosa-Póvoa, A.P., & Matos, H.A. (2003). Optimal Periodic Scheduling of Batch Plants Using RTN-Based Discrete and Continuous-Time Formulations: A Case Study Approach. *Industrial & Engineering Chemistry Research*, 42, 3346.

7) Castro, P.M., Barbosa-Póvoa, A.P., & Novais, A.Q. (2005). Simultaneous Design and Scheduling of Multipurpose Plants Using Resource Task Network Based Continuous-Time Formulations. *Industrial & Engineering Chemistry Research*, 44, 343.

8) Castro, P.M.; Westerland, J.; Forssel, S. (2009). Scheduling of a continuous plant with recycling of by products: A case study from a tissue paper mill. *Computers & Chemical Engineering,* 33, 347-358.

9) Dinkelbach W. (1967). On Nonlinear Fractional Programming. *Management Science*, 13, (7), 492-498.

10) Duran, M. A.; Grossmann, I. E. (1986). An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*, 36, (3), 307.

11) Geoffrion, A. M. (1972). Generalized Benders Decomposition. *Journal of Optimization Theory and Applications*, 10, (4), 237-260.

12) Kesavan, P.; Allgor, R. J.; Gatzke, E. P.; Barton, P. I. (2004). Outer Approximation Algorithms for Separable Nonconvex Mixed-Integer Nonlinear Programs. *Mathematical*

Programming, 100, (3), 517-535.

13) Kondili, E., Pantelides, C.C., & Sargent, R. (1993). A General Algortihm for Short-term Scheduling of Batch Operations. I, MILP Formulation. *Computers & Chemical Engineering*, 17, 211.

14) Leyffer, S. (2001). Integrating SQP and branch and bound for mixed integer nonlinear programming. *Computational Optimization and Applications*, 18, 295-309.

15) Pantelides, C.C. (1994). Unified Frameworks for the optimal process planning and scheduling. In Proceedings of the second conference on foundations of computer aided operations (p.253). New York: Cache Publications.

16) Pinto, J. M.; Grossmann, I. E. (1994). Optimal Cyclic Scheduling of Multistage Continuous Multiproduct Plants. *Computers & Chemical Engineering,* 18(9), 797-816

17) Pinto, J. M.; Joly, M.; Moro, L. F. L. (2000). Planning and scheduling models for refinery operations. *Computers & Chemical Engineering,* 24(9-10), 2259-2276

18) Pochet, Y.; Warichet, F. (2008). A tighter continuous time formulation for the cyclic scheduling of a mixed plant. *Computers & Chemical Engineering,* 32(11), 2723-2744

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

20) Quesada, I.; Grossmann, I. E. (1995). A Global Optimization Algorithm for Linear Fractional and Bilinear Programs. *Journal of Global Optimization*, 6, 39-76.

21) Ryoo, H. S.; Sahinidis, N. V., (1996). A branch-and-reduce approach to global optimization. *Journal of Global Optimization*, 8, (2), 107-138.

22) Sahinidis, N. V. (1996). BARON: A general purpose global optimization software package. *Journal of Global Optimization*, 8, (2), 201-205.

23) Sahinidis, N. V.; Grossmann, I. E. (1991). MINLP model for cyclic multiproduct scheduling on continuous parallel lines. *Computers & Chemical Engineering*, 15, (2),

85-103

24) Schaible, S. (1976). Fractional Programming II: On Dinkelbach's Algorithm. *Management Science*. 22, (7), 868-873.

25) Schilling, G., & Pantelides, C.C. (1999). Optimal Periodic Scheduling of Multipurpose Plants. *Computers & Chemical Engineering*, *23*, 635.

26) Shah, N., Pantelides, C.C., & Sargent, R. (1993). Optimal Periodic Scheduling of Multipurpose Batch Plants. *Annals of Operations Research*, *42*, 193.

27) Smith, E. M. B.; Pantelides, C. C. (1999). A symbolic reformulation/spatial branch and bound algorithm for the global optimization of nonconvex MINLPs. *Computers & Chemical Engineering*, 23, 457-478.

28) Tawarmalani, M.; Sahinidis, N. V. (2004). Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. *Mathematical Programming*, 99, 563-591.

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

30) Warichet, F. (2007). *Scheduling of mixed batch-continuous production lines*. PhD thesis, Université Catholique de Louvain, Belgium.

31) Westerlund, T.; Pettersson, F. (1995). A cutting plane method for solving convex MINLP problems. *Computers & Chemical Engineering*, 19, S131-S136.

32) Westerlund, T.; Porn, R. (2002). Solving Pseudo-Convex Mixed Integer Optimization Problems by Cutting Plane Techniques. *Optimization & Engineering*, 3, (3), 253-280.

33) Wu, D., & Ierapetritou, M. (2004). Cyclic short-term scheduling of multiproduct batch plants using continuous-time representation. *Computers & Chemical Engineering*, *28*, 2271–2286