Robust and Efficient Algorithm for Optimizing Crude Oil Operations

*J. Li, W. K. Li, I. A. Karimi[1] (**Speaker**), R. Srinivasan*
Department of Chemical and Biomolecular Engineering
National University of Singapore
4 Engineering Drive 4, Singapore 117576

1. Introduction

Scheduling of crude oil is an important and complex routine task in a refinery. It involves crude oil unloading, tank allocation, storage and blending of different crudes, and CDU charging. Optimal crude oil scheduling can increase profits by using cheaper crudes, minimizing crude changeovers, avoiding ship demurrage, and managing crude inventory optimally. However, mathematical modeling of the blending of different crudes in storage tanks results in bilinear terms, which turns the whole problem into a difficult, nonconvex, mixed integer nonlinear program (MINLP).

So far, several efforts (Lee et al., 1996; Li et al., 2002; Moro and Pinto, 2004; and Reddy et al., 2004a,b) in the literature have attempted to solve this MINLP problem. Lee et al. (1996) used reformulation linearization technology (RLT) to turn bilinear equations into linear forms. However, this linearization approximation leads to composition discrepancy (the amounts of individual crudes delivered from a tank to CDU are not proportional to the crude composition in the tank) as shown by Li et al. (2002) and Reddy et al. (2004b). Discretization procedure of Moro and Pinto (2004) leads to discrete values for flow rates and increases problem size to an extent that makes it almost impossible to solve reasonably sized problems. General purpose solver such as DICOPT and the method of Li et al. (2002) require solving one MILP and one NLP iteratively. Reddy et al. (2004a,b) solve a series of MILPs to avoid solving NLP. However, we find that DICOPT as well as the algorithms of Li et al. (2002) and Reddy et al. (2004a, b) fail to get feasible schedules in several examples, although feasible solutions do exist. Moreover, these algorithms still need large solution times for solving large, practical problems.

Therefore, no reliable, robust, and efficient algorithm exists in the literature for this real, practical, and very useful problem. Moreover, only one crude property is considered in the literature. The flow rate change to any CDU is not limited either. In this paper, we will first identify fifteen crude properties that are critical to crude distillation and downstream processing. We enhance the practical utility of Reddy et al. (2004b)'s MINLP formulation by adding appropriate linear blending correlations for these properties. Next, we will propose a robust algorithm to solve this MINLP problem. To further increase solution speed and improve optimality, we will develop a partial relaxation strategy in which we relax the integrality restrictions on the binary variables of limited use. In addition, we will revise Reddy et al. (2004b)'s formulation to ensure practically realistic schedules with limited flow rate changes to the CDUs.

2. Problem Definition

The problem we studied is taken from Reddy et al. (2004b) with some modifications. Figure 1 shows the schematic configuration of crude oil scheduling in a typical marine access refinery. The detailed

[1] Corresponding author
Email: cheiak@nus.edu.sg

description can be referred to Reddy et al. (2004b).

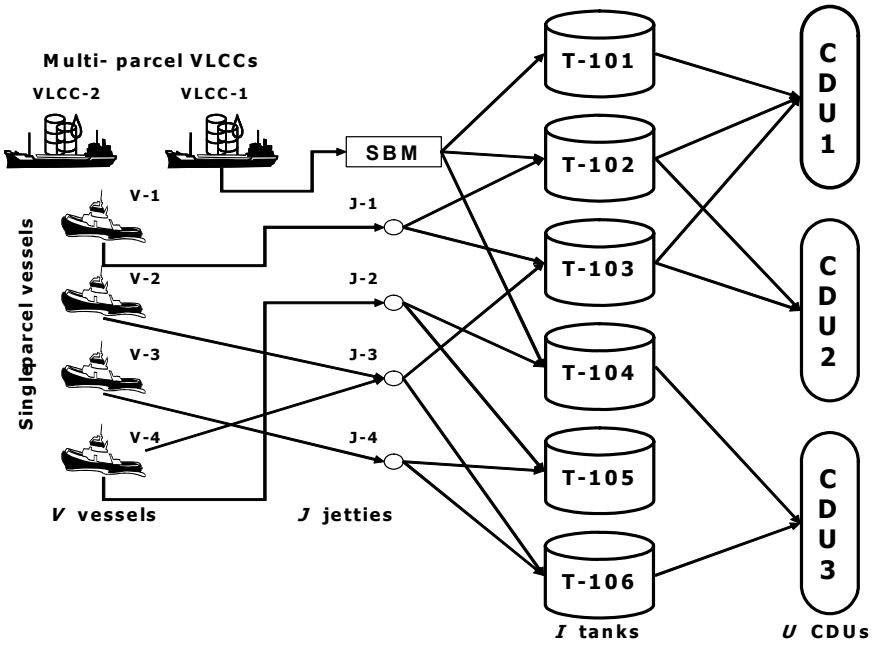

Figure 1 Schematic of oil unloading and processing

3. Mathematical Formulation

In this paper we use the formulation developed by Reddy et al. (2004b). The details about their formulation including the definition of variables and parameters can be found in their paper (Reddy et al., 2004b).

4. Flow rate change limitation to any CDU

The flow rate of any CDU cannot change too much between two adjacent periods in order to stabilize the CDU operation. We regulate that the flow rate change of any CDU in the next period should be within some acceptable fraction of that in the previous period as follows,

$$FU_{u(t+1)} \geq fkk1 \cdot FU_{ut} \tag{1a}$$

$$FU_{u(t+1)} \leq fkk2 \cdot FU_{ut} \tag{1b}$$

$fkk1$ and $fkk2$ are parameters. $fkk1$ can be the value within 75%~85% while $fkk2$ can be the value within 1.15%~1.25%.

5. Infeasibility of Reddy et al. (2004b)'s algorithm

As mentioned before, Reddy et al. (2004b)'s algorithm may lead to infeasibility in some examples. Now we analyze their algorithm in detail. In their algorithm, they use the following two constraints (2a) and (2b) to correct composition discrepancy. If f_{ict} is known, then the whole problem turns to MILP problem not MINLP problem. They also observe that during some blocks of contiguous

$$FCTU_{iuct} = f_{ict} \cdot FTU_{iut} \tag{2a}$$

$$VCT_{ict} = f_{ict} \cdot V_{it} \tag{2b}$$

periods, the composition in each tank does not change. For such a block, if f_{ict} is known, then the problem is also a MILP problem. Thus, the whole periods are divided into two blocks for each tank. One for which the tank composition is known, and the other for which the tank composition is not. They also notice that the composition in each tank changes only when a tank receives crudes. Otherwise, it does not change. Moreover, the status of all configurations in a refinery is also known in the beginning. Therefore, they first identify the first possible period for each tank to receive crudes. From the first period to this first possible period of receiving crudes, the composition in each tank is constant and known, denoted as the initial block of periods. They use constraints (2a) and (2b) in this initial block and drop (2a) and (2b) in the remaining periods. The whole problem is a MILP problem. A solution can be obtained with no composition discrepancy in this initial block. After solving, they fix all variables in this initial block. Then they try to identify the second possible period for each tank to receive crudes and the second block for each tank. They continue with this strategy until a near optimal solution is obtained. Therefore the reason for infeasibility of their algorithm is the progressive fixing of binary variables.

6. Robust Algorithm

As discussed above, once the infeasibility occurs in the nth period, it means that the combination of binary variables before the nth period is probably infeasible. However, their algorithm lacks a mechanism to retract from these infeasible combinations. If we could eliminate this combination somehow, we could obtain a feasible solution. Based on this idea, we propose an integer cut strategy to remove this infeasible combination of binary variables. This allows us to return back to solve it again until the solver can give a feasible solution.

6.1 Integer Cut Constraint

Based on the above discussion, we should remove the infeasible combination of binary variables before period n once Reddy et al. (2004b)'s algorithm cannot find a feasible solution in period n. The following lemma gives the integer cut constraint.

Lemma 1 Consider a binary variable solution y_j =1(j∈**NZ**) and y_j=0 (j∈**Z**) of n binary variables, y_j (j=1, 2, …, n), the following constraint

$$\sum_{j\in NZ} y_j - \sum_{j\in Z} y_j \le |NZ| - 1$$

eliminates this binary solution.

Where:

|NZ| is the cardinality of set **NZ**, while |Z| is the cardinality of set **Z**.

Based on Lemma 1, we can remove the infeasible combination of binary variables.

6.2 Time Representation

When infeasibility of Reddy et al. (2004b)'s algorithm occurs in period n, the combination of binary variables before period n is infeasible. There are many cases for this infeasible combination of binary variables. For example, the combination of binary variables in period (n-1) may be infeasible. Or the combination of binary variables in period 2 may be infeasible. Or the combination of binary variables in several periods before period n is infeasible. In other words, we do not know which combination of binary variables is infeasible. One way is to do integer cut reversely based on Reddy et al. (2004b)'s algorithm. This method is effective when infeasible combination of integers is near to period n. However this method will do many integer cuts when infeasible combination of binary

variables happens in earlier periods. Another way is to do integer cut from the first period to period (n-1). When n is big, |NZ| or |Z| or both are big. It means the feasible region for y_j is large. We should find a feasible combination of integers in a large integer region. This will also take much time even if the infeasible combination of integers is near to period n. Therefore, we divide the scheduling horizon into several blocks according to the scheduled arrival times of vessels, denoted as Vessel-Arrival-Time based blocks. The first block begins at time zero and ends at the scheduled arrival time of the first vessel. The second block follows immediately after the first, and ends at the scheduled arrival time of the second vessel, and the remaining periods follow likewise. If the arrival time of a vessel is earlier than the latest expected departure time of its previous vessel, then the two blocks are formed to one block. The block division is shown in Figure 2. Each block spans several periods. Our block can overcome the disadvantages of the above two methods.

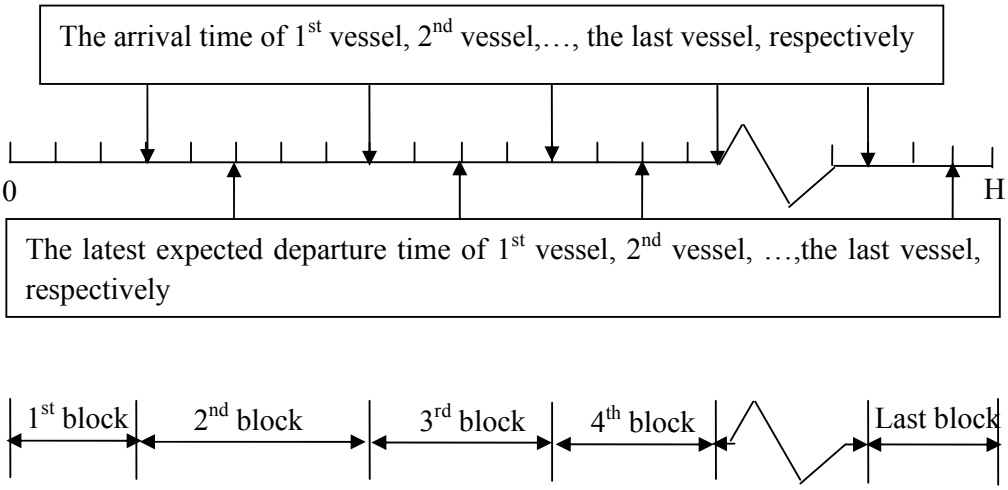

Figure 2 Vessel-Arrival-Time based block division

6.3 Integer Cut Variables

Note that if |NZ| or |Z| is a big number, then we may need many integer cuts. Thus we should reduce |NZ| and |Z| as much as possible to improve the efficiency of integer cut strategy. Based on this discussion, we choose integer cut variable.

In this problem, only XP_{pt}, XI_{it} and Y_{iut} are binary variables. In any period at least one Y_{iut} is equal to 1 and any Y_{iut} cannot be fixed because which tank feeds which CDU at any period is not known a prior. Therefore |NZ| or |Z| or both of Y_{iut} are large for each integer cut although crude oil segregation exists. So Y_{iut} should not be chosen for integer cut variables. Parcels may be connected to SBM or jetties although they may not begin to unload. In the same way, parcels may be connected to SBM or jetties although they are unloaded completely. Although we can fix some XP_{pt} in terms of the arrival time of each parcel, |NZ| or |Z| for XP_{pt} is still bigger compared to XI_{it} which will be explained as follows. Moreover, changing parcel connection to SBM or jetties seems to be nonsense. XI_{it} denotes tank connection to SBM / jetty discharge line. If the tank receives a parcel, then XI_{it} is equal to one. Otherwise it is equal to zero. Reddy et al. (2004b) defined possible unloading periods for parcels, in other periods XI_{it} was fixed to zero, but XP_{pt} cannot be fixed to zero because parcel demurrage is allowed. So both |NZ| and |Z| for XI_{it} are smaller than XP_{pt} and Y_{iut}.

Crudes are segregated into several classes in practical operations of a refinery. Different classes of crudes are stored in different tanks and processed in different CDUs. We identify which class or classes of crudes fail and do integer cut respectively. This can reduce both |NZ| and |Z| and improve the efficiency of integer cut strategy. In addition, the change of XI_{it} will cause the change of Y_{iut} and may cause the change of XP_{pt} based on the formulation, but the change of XP_{pt} may not lead to the change of XI_{it} and Y_{iut}. Therefore integer cut for XP_{pt} at each time may not lead to any changes of other variables.

Therefore, we choose XI_{it} as variables in our proposed integer cut. According to Lemma 1, the integer cut constraint for our problem can be described as follows.

$$\sum_{i\in NZi}\sum_{t\in B} XI_{it} - \sum_{i\in Zi}\sum_{t\in B} XI_{it} \le \ |NZi| - 1 \tag{3}$$

Where

|NZi| is the cardinality of set **NZi**.

6.4 Another MILP Formulation

In Reddy et al. (2004b)'s algorithm, they use constraints (2a) and (2b) to correct the composition discrepancy in period t when solving in period t. If composition discrepancy exists in period t, then a solution can be obtained although the solution is infeasible. In other words, their algorithm cannot lead to infeasibility. Based on this idea, we introduce two slack positive variables $u1_{iuct}$ and $u2_{iuct}$ and add them to the constraints (2a) as follows,

$$FCTU_{iuct} = f_{ict} * FTU_{iut} - u1_{iuct} + u2_{iuct} \tag{4}$$

If slack variables $u1_{iuct}$ and $u2_{iuct}$ are both zero, then constraint (4) is reduced to (2a). There is no composition discrepancy. If $u1_{iuct}$ or $u2_{iuct}$, or both are nonzero, then composition discrepancy exists. Based on $u1_{iuct}$ or $u2_{iuct}$, we can know which class or classes of crudes cannot meet constraint (2a). To get the value of $u1_{iuct}$ and $u2_{iuc,}$ we construct another optimization problem.

To be convenient, the original MILP optimization problem denoted as **(F)** can be presented as follows,

$$\text{Max} \quad \text{Profit} = \sum_i\sum_u\sum_c\sum_t FCTU_{iuct}CP_c - \sum_v DC_v - COC\sum_u\sum_t CO_{ut} - \sum_t SC_t$$

Subject to: original constraints in Reddy et al. (2004b) and constraints (1a) and (1b)

(F)

Another MILP optimization problem using Equation (4) instead of Equation (2a) and denoting as **(F1)** is constructed as:

$$\text{Min} \quad PK = \sum_i\sum_u\sum_c\sum_t (u1_{iuct} + u2_{iuct}) \qquad (i, u) \in \boldsymbol{IU}, (i, c) \in \boldsymbol{IC}$$

Subject to: original constraints in Reddy et al. (2004b) except (2a) and constraints (1a), (1b) and (5)

(F1)

Because **(F1)** is another MILP problem, solving this MILP problem may need much time. Note that **(F1)** is just to get values of $u1_{iuct}$ and $u2_{iuct}$, we do not need to get an optimal solution for all cases. Therefore we prescribe computation time of solving **(F1)**. By doing this

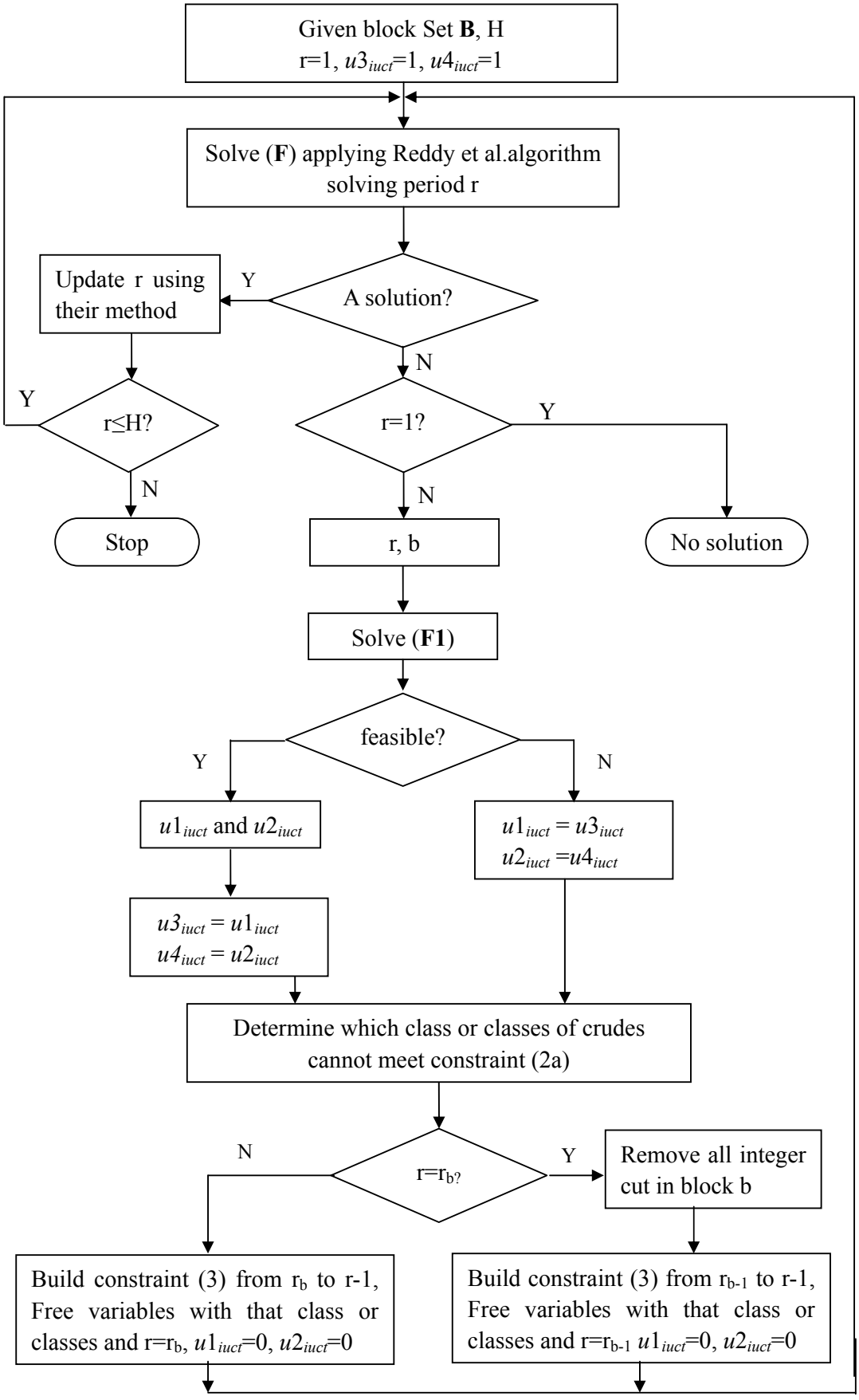

Figure 3 The flow chart of our improved algorithm

, we may get an optimal solution, feasible solution or no solution for **(F1)**.

6.5 Integer Cut Strategy

Now we present an iteratively algorithm based on integer cut to solve this MINLP problem. We first solve **(F)** by using Reddy et al. (2004b)'s algorithm and let r denote the period. When their algorithm fails, then we know in which period it fails according to r and know which block period r belongs to. Then we solve **(F1)** to obtain $u1_{iuct}$ and $u2_{iuct}$. $u1_{iuct}$ and $u2_{iuct}$ tell us which or which classes of crudes cannot satisfy constraint (2a). If only one class of crudes cannot meet constraints (2a), then we set up integer cut constraint for XI_{it} of that class of crudes from the beginning of that block to period (r-1) by using constraints (3). If two or more classes of crudes cannot satisfy constraints (2a) simultaneously, we set up integer cut constraints for different classes of crudes respectively. Then we add it or them to both **(F)** and **(F1)**, return back to the beginning of that block and solve **(F)** again until the solver can find a feasible solution for **(F)**. If the solver cannot give us a feasible solution in that block, then we will return back to the previous block and do integer cut again. The detailed flow chart for the improved algorithm is shown in Figure 3.

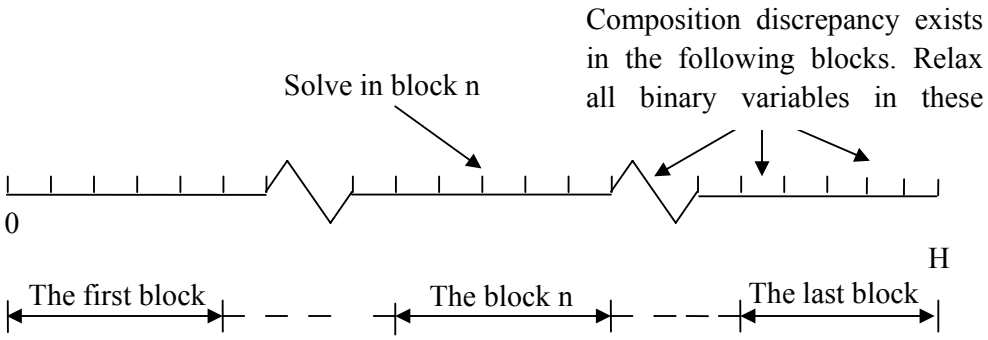

Figure 4 The schematic of partial relaxation idea

7. Partial Relaxation strategy

Another challenge of crude oil scheduling is the speed and optimality. Our algorithm shows that composition discrepancy exists in the following blocks when we solve previous blocks. It means all variables in the following blocks are not feasible and useless for us. Based on this idea, we relax all binary variables in the following blocks when solving previous blocks. Figure 4 shows the idea of this partial relaxation method. By using this partial relaxation method, we can get a feasible solution. Starting from this initial feasible solution, we have two choices. One is to solve the whole problem as a NLP by fixing all binary variables and a MIP by fixing compositions of all tanks iteratively. The other is to solve a MIP and a NLP iteratively. The termination criterion for both choices is that a better objective cannot find between two successive NLP or the absolute relative difference between two successive NLP is smaller than an appropriate tolerance. We choose the better solution as our final solution. The tolerance used in this paper is 1.0×10^{-5}. The whole process is called partial relaxation strategy. The procedure of partial relaxation strategy is shown in Figure 5.

8. Results and Discussions

Twenty examples are studied whose data mainly obtained from Reddy et al. (2004b) and Li et al. (2002). These examples involve different sizes and different real life operation features. Fifteen crude properties that are critical to crude distillation and downstream processing are incorporated in some

examples. These fifteen crude properties are: specific gravity, sulfur content, nitrogen content, oxygen content, carbon residue content, pour point, flash point, Ni content, Reid vapor pressure, asphaltene content, aromatics content, paraffins content, naphthene content, viscosity content and wax content. All examples are computed on a Dell workstation PWS650 (Inter® Xeron™ CPU 3.06GHZ, 3.5 GB memory) running Windows NT using solver CPLEX 9.0.

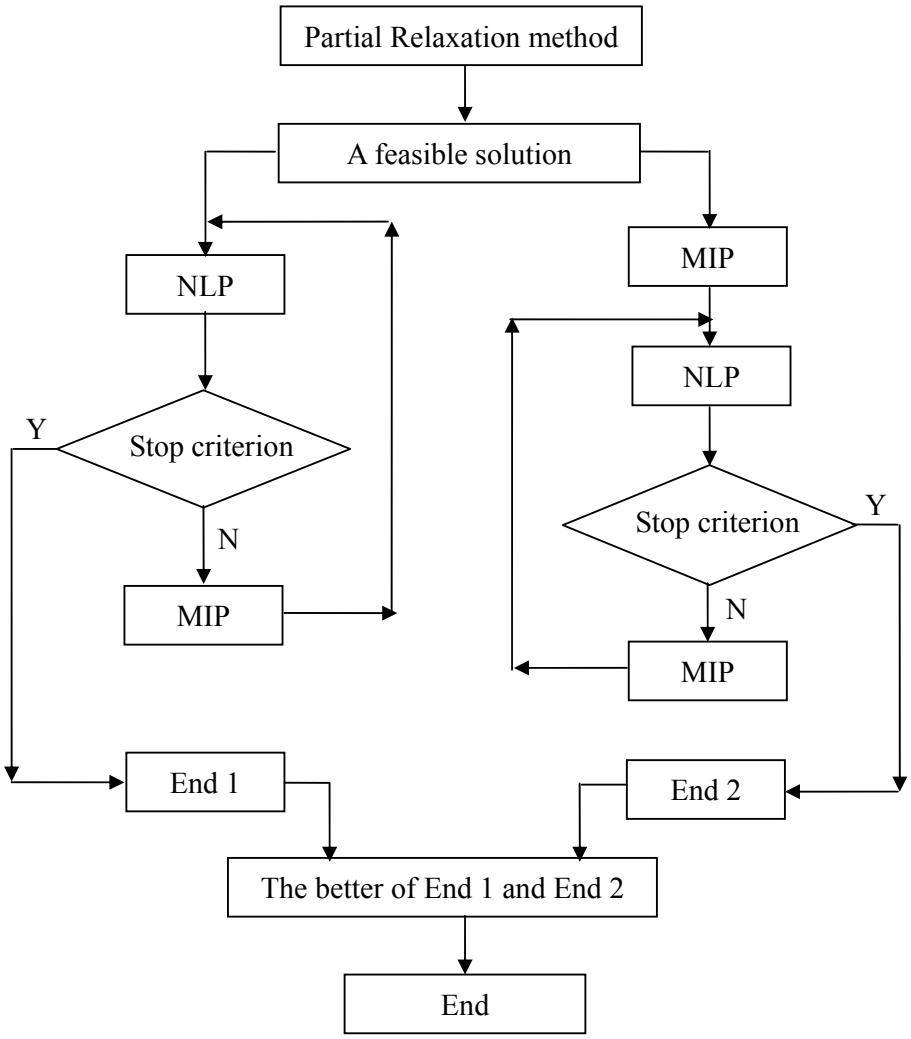

Figure 5 The procedure of partial relaxation strategy

All twenty examples are solved using the algorithms such as DICOPT, Li et al. 2002, Reddy et al., 2004b and our robust algorithm. The results are shown in Tables 1a to 1c. From Tables 1a and 1c, it can be concluded that DICOPT fails to solve most problems and is horribly slow in solving the rest. The algorithm of Li et al. (2002) also fails in most problems. Even the best algorithm of Reddy et al. (2004b) fails to solve several problems. In contract, our improved algorithm works on all problems and is much more efficient than the other three algorithms (DICOPT, Li et al. 2002 and Reddy et al. 2004b).

We also solve these twenty examples using partial relaxation strategy. The result is shown in Table 2a and 2b. Table 2a and 2b show that the partial relaxation strategy greatly reduces the

computation time and at the same time improves the solution quality for most examples. To further illustrate the capability of our partial relaxation strategy, three bigger examples whose horizons are about 60 periods (i.e., 20 days) are also solved, which is shown in Table 3. From Table 3, we can see that we can get feasible solutions for these three bigger examples within less computation time.

9. Conclusion

In this paper, we proposed a new robust algorithm based on a backtracking strategy using an intelligent integer cut to solve this complex MINLP problem. We evaluated the robustness of our improved algorithm using twenty examples of different sizes and with different real life operation features and compared its performance with other three algorithms. The results show that our improved algorithm is much more efficient than the other three algorithms. We also developed a partial relaxation strategy to further increase solution speed and improve solution quality. Our tests show that the partial relaxation strategy greatly reduced the computation time and improved the solution quality for most examples simultaneously, especially for scheduling problems with horizons as long as 20 days. In addition, two constraints were imposed to make sure practically realistic schedules with limited flow rate changes to the CDUs.

Reference

Lee, H., Pinto, J. M., Grossmann, I. E., Park, S., Mixed-integer Linear Programming Model for Refinery Short-Term Scheduling of Crude Oil Unloading with Inventory Management, *Industrial and Engineering Chemistry Research*, 1996, 35(5), 1630-1641.

Li W. K., Hui, C. W., Hua, B., Zhong, X. T., Scheduling Crude Oil Unloading, Storage and Processing, *Industrial and Engineering Chemistry Research*, 2002, 41, 6723-6734.

Moro, L. F. L., Pinto, J. M., Mixed-Integer Programming Approach for Short-Term Crude Oil Scheduling, *Industrial and Engineering Chemistry Research*, 2004, 43, 85-94.

Reddy, P. C. P., Karimi, I. A., Srinivasan, R., A Novel Continuous-time formulation for Scheduling of Crude Oil Operations, *Chemical Engineering Science*, 2004a, 59, 1325-1341.

Reddy, P. C. P., Karimi, I. A., Srinivasan, R., A Novel Solution Approach for Optimizing Scheduling of Crude Oil Operations, *AIChE Journal,* 2004b, 50(6), 1177-1197.

Nomenclature

Sets

B Set of blocks

Subscripts

b Blocks

Parameters

*fkk*1 Flow rate fraction between two adjacent period

*fkk*2 Flow rate fraction between two adjacent period

r A parameter denotes period

mn_b A parameter denotes block b is being solved

rb_b The first period of block b

Continuous Variables

$u1_{iuct}$ Slack variable

$u2_{iuct}$ Slack variable

Table 1a The comparison of different algorithms

Examples	Algorithms	Discrete Variables	Computation Time (CPU Seconds)	Objective (K$)	Relative MILP Gaps % (periods)
1	DICOPT	53	N/A(I)	N/A	0% (1-9)
	Li et al.(2002)	53	N/A	N/A	0% (1-9)
	Reddy et al.(2004b)	53	N/A	N/A	0% (1-9)
	Improved Algorithm	53	2.5	5069.74	0% (1-9)
2	DICOPT	151	N/A(II)	N/A	0.01% (1-7)
	Li et al.(2002)	151	N/A	N/A	0.01% (1-7)
	Reddy et al.(2004b)	151	N/A	N/A	0.01% (1-7)
	Improved Algorithm	151	4.3	101190750.00	0.01% (1-7)
3	DICOPT	304	42688.86(III)	N/A	5%(1-20)
	Li et al.(2002)	304	N/A	N/A	5%(1-20)
	Reddy et al.(2004b)	304	N/A	N/A	5%(1-5), 3.5%(6-10), 2%(11-15), 0%(16-20)
	Improved Algorithm	304	263.0	3417.29	5%(1-5), 3.5%(6-10), 2%(11-15), 0%(16-20)
4	DICOPT	304	30668.8(III)	N/A	5%(1-20)
	Li et al.(2002)	304	N/A	N/A	5%(1-20)
	Reddy et al.(2004b)	304	N/A	N/A	5%(1-5), 3.5%(6-10), 2%(11-15), 0%(16-20)
	Improved Algorithm	304	180.3	3422.12	5%(1-5), 3.5%(6-10), 2%(11-15), 0%(16-20)
5	DICOPT	304	22383.62	3000.13	5% (1-20)
	Li et al.(2002)	304	N/A	N/A	5%(1-20)
	Reddy et al.(2004b)	304	N/A	N/A	5%(1-5), 3.5%(6-10), 2%(11-15), 0%(16-20)
	Improved Algorithm	304	158.1	3065.86	5%(1-5), 3.5%(6-10), 2%(11-15), 0%(16-20)
6	DICOPT	589	N/A(II)	N/A	7%(1-42)
	Li et al.(2002)	589	N/A	N/A	7%(1-42)
	Reddy et al.(2004b)	589	N/A	N/A	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
	Improved Algorithm	589	1830.2	4514.6	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
7	DICOPT	589	38033.36(III)	N/A	7%(1-42)
	Li et al.(2002)	589	N/A	N/A	7%(1-42)
	Reddy et al.(2004b)	589	N/A	N/A	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
	Improved Algorithm	589	6887.7	4533.35	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)

Table 1b The comparison of different algorithms (Continue)

Examples	Algorithms	Discrete Variables	Computation Time (CPU Seconds)	Objective (K$)	Relative MILP Gaps % (periods)
8	DICOPT	589	55226.3(III)	N/A	7%(1-42)
	Li et al.(2002)	589	N/A	N/A	7%(1-42)
	Reddy et al.(2004b)	589	N/A	N/A	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
	Improved Algorithm	589	1921.1	4567.58	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
9	DICOPT	589	33403.8(III)	N/A	7%(1-42)
	Li et al.(2002)	589	N/A	N/A	7%(1-42)
	Reddy et al.(2004b)	589	N/A	N/A	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
	Improved Algorithm	589	5726.8	4578.57	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
10	DICOPT	589	40155.1(III)	N/A	7%(1-42)
	Li et al.(2002)	589	N/A	N/A	7%(1-42)
	Reddy et al.(2004b)	589	N/A	N/A	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
	Improved Algorithm	589	5562.4	4542.78	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
11	DICOPT	589	49143.1(III)	N/A	7%(1-42)
	Li et al.(2002)	589	N/A	N/A	7%(1-42)
	Reddy et al.(2004b)	589	N/A	N/A	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
	Improved Algorithm	589	8832.5	4150.61	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
12	DICOPT	589	53132.61(III)	N/A	7%(1-42)
	Li et al.(2002)	589	N/A	N/A	7%(1-42)
	Reddy et al.(2004b)	589	10913.4	4059.39	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
	Improved Algorithm	589	10913.4	4059.39	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
13	DICOPT	589	71427.05(III)	N/A	7%(1-42)
	Li et al.(2002)	589	N/A	N/A	7%(1-42)
	Reddy et al.(2004b)	589	N/A	N/A	7%(1-21), 4%(22-33), 2%(34-36), 0%(37-42)
	Improved Algorithm	589	7272.6	4029.4	7%(1-21), 4%(22-33), 2%(34-36), 0%(37-42)
14	DICOPT		61123.45(III)	N/A	7% (1-42)
	Li et al.(2002)		N/A	N/A	7% (1-42)
	Reddy et al.(2004b)		N/A	N/A	7% (1-19), 4% (20-31), 2% (32-33), 0% (34-42)
	Improved Algorithm		15097.9	4424.47	7% (1-19), 4% (20-31), 2% (32-33), 0% (34-42)

Table 1c The comparison of different algorithms (End)

Examples	Algorithms	Discrete Variables	Computation Time (CPU Seconds)	Objective (K$)	Relative MILP Gaps % (periods)
15	DICOPT	624	68156.73(III)	N/A	7%(1-42)
	Li et al.(2002)	624	N/A	N/A	7%(1-42)
	Reddy et al.(2004b)	624	N/A	N/A	4%(1-21), 3%(22-25), 0.1%(26-42)
	Improved Algorithm	624	14892	4718.75	4%(1-21), 3%(22-25), 0.1%(26-42)
16	DICOPT	589	14829.1(III)	4400.68	7%(1-42)
	Li et al.(2002)	589	N/A	N/A	7%(1-42)
	Reddy et al.(2004b)	589	N/A	N/A	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
	Improved Algorithm	589	3067.9	4456.93	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
17	DICOPT	589	56122.6(III)	N/A	7%(1-42)
	Li et al.(2002)	589	N/A	N/A	7%(1-42)
	Reddy et al.(2004b)	589	N/A	N/A	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
	Improved Algorithm	589	10596.8	4437.41	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
18	DICOPT	589	7027.73	4572.15	7%(1-42)
	Li et al.(2002)	589	679.1	4640.6	7%(1-42)
	Reddy et al.(2004b)	589	1532.9	4573.74	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
	Improved Algorithm	589	1532.9	4573.74	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
19	DICOPT	589	27178.95(III)	N/A	4%(1-42)
	Li et al.(2002)	589	N/A	N/A	4%(1-42)
	Reddy et al.(2004b)	589	2736.2	4727.9	4%(1-21), 3%(22-25), 0.1%(26-42)
	Improved Algorithm	589	2736.2	4727.9	4%(1-21), 3%(22-25), 0.1%(26-42)
20	DICOPT	624	7329.19	4743.36	4%(1-42)
	Li et al.(2002)	624	N/A	N/A	4%(1-42)
	Reddy et al.(2004b)	624	2919.4	4719.79	4%(1-21), 3%(22-25), 0.1%(26-42)
	Improved Algorithm	624	2919.4	4719.79	4%(1-21), 3%(22-25), 0.1%(26-42)

N/A: Nonsense

(I): DICOPT will stop when 200 iterations have reached.

(II): The first relaxed NLP of DICOPT is infeasible

(III): DICOPT stop when the computation time have reached.

Table 2a The comparison result of partial relaxation method and non-relaxation method

Examples	Methods	Single Variables	Discrete Variables	Solution Time (CPU Seconds)	The Number of Integer cut	Final Objective	Relative MILP Gap % (Periods)
1	Non-relaxed	426	53	2.50	1	5069.94	0% (1-9)
	Partial-Relaxation	426	53	3.40	1	5069.94	0% (1-9)
2	Non-Relaxation	811	128	4.30	1	101190750.00	0.01% (1-7)
	Partial-Relaxation	811	128	5.00	1	101190800.00	0.01% (1-7)
3	Non-Relaxation	2972	304	263.00	3	3417.29	5%(1-5), 3.5%(6-10), 2%(11-15), 0%(16-20)
	Partial-Relaxation	2972	246	270.80	61	3456.67	5%(1-5), 3.5%(6-10), 2%(11-15), 0%(16-20)
4	Non-Relaxation	2972	304	180.30	2	3422.12	5%(1-5), 3.5%(6-10), 2%(11-15), 0%(16-20)
	Partial-Relaxation	2809	195	64.70	7	3427.90	5%(1-5), 3.5%(6-10), 2%(11-15), 0%(16-20)
5	Non-Relaxation	2972	304	158.10	1	3065.86	5%(1-5), 3.5%(6-10), 2%(11-15), 0%(16-20)
	Partial-Relaxation	2809	195	156.10	9	3086.59	5%(1-5), 3.5%(6-10), 2%(11-15), 0%(16-20)
6	Non-Relaxation	6102	589	1830.20	1	4514.60	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
	Partial-Relaxation	5785	327	7681.90	241	4622.76	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
7	Non-Relaxation	6102	589	6887.70	2	4533.35	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
	Partial-Relaxation	5785	327	335.40	2	4605.16	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
8	Non-Relaxation	6102	589	1921.10	5	4567.58	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
	Partial-Relaxation	5785	327	111.10	0	4644.75	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
9	Non-Relaxation	6102	589	5726.80	7	4578.57	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
	Partial-Relaxation	5785	327	1660.90	18	4611.40	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
10	Non-Relaxation	6102	589	5562.40	11	4542.78	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
	Partial-Relaxation	5785	327	1465.30	5	4611.25	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)

Table 2b The comparison result of paritial relaxation method and non-relaxation method (Continued)

Examples	Methods	Single Variables	Discrete Variables	Solution Time (CPU Seconds)	The Number of Integer cut	Final Objective	Relative MILP Gap % (Periods)
11	Non-Relaxation	6102	589	8832.50	4	4150.61	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
	Partial-Relaxation	5785	327	226.60	0	4165.28	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
12	Non-Relaxation	6106	589	10913.40	10	4059.39	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
	Partial-Relaxation	5789	303	4366.30	5	4124.70	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
13	Non-Relaxation	6106	589	7272.60	11	4029.40	7%(1-21), 4%(22-33), 2%(34-36), 0%(37-42)
	Partial-Relaxation	5889	325	4482.40	21	4076.18	7%(1-21), 4%(22-33), 2%(34-36), 0%(37-42)
14	Non-Relaxation	6106	585	15097.90	7	4424.47	7% (1-19), 4% (20-31), 2% (32-33), 0% (34-42)
	Partial-Relaxation	5849	323	4568.00	2	4526.42	7% (1-19), 4% (20-31), 2% (32-33), 0% (34-42)
15	Non-Relaxation	6219	624	14892.00	23	4718.75	4%(1-21), 2%(22-25), 0.1%(26-42)
	Partial-Relaxation	5742	323	4968.70	23	4745.84	4%(1-21), 2%(22-25), 0.1%(26-42)
16	Non-Relaxation	6102	589	3067.90	1	4456.93	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
	Partial-Relaxation	5785	327	1688.60	1	4458.58	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
17	Non-Relaxation	6102	589	10596.80	9	4437.41	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
	Partial-Relaxation	5785	327	4430.50	33	4473.29	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
18	Non-Relaxation	6102	589	1532.90	0	4573.74	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
	Partial-Relaxation	5785	327	254.70	0	4611.02	7%(1-18), 4%(19-28), 2%(29-30), 0%(31-42)
19	Non-Relaxation	6219	624	2736.20	0	4727.90	4%(1-21), 2%(22-25), 0.1%(26-42)
	Partial-Relaxation	5742	323	600.10	0	4745.40	4%(1-21), 2%(22-25), 0.1%(26-42)
20	Non-Relaxation	6219	624	2919.40	0	4719.79	4%(1-21), 2%(22-25), 0.1%(26-42)
	Partial-Relaxation	5742	323	445.61	0	4748.58	4%(1-21), 2%(22-25), 0.1%(26-42)

Table 3 The performance of partial relaxation strategy for bigger problems

Examples	Single Variables	Discrete Variables	Non Zero Elements	Computation Time (CPU Seconds)	Periods	Objective	Relative Gap (periods)
1	8491	364	52049	2278.97	60	7485.50	10% (1-20), 6% (21-32), 4% (33-46) 2% (47-49), 0% (50-60)
2	8491	364	152849	2506.16	60	7412.81	10% (1-20), 6% (21-32), 4% (33-46) 2% (47-49), 0% (50-60)
3	9140	382	81585	1067.99	60	7492.97	10% (1-20), 6% (21-32), 4% (33-46) 2% (47-49), 0% (50-60)