Improving the Robustness and Efficiency of Crude Scheduling Algorithms

Jie Li, Wenkai Li, and I. A. Karimi
Dept. of Chemical and Biomolecular Engineering, National University of Singapore,
4 Engineering Drive 4, Singapore 117576

Rajagopalan Srinivasan
Dept. of Chemical and Biomolecular Engineering, National University of Singapore, 4 Engineering Drive 4,
Singapore 117576
Process Sciences and Modeling, Institute of Chemical and Engineering Sciences, 1 Pesek Road,
Jurong Island, Singapore 627833

DOI 10.1002/aic.11280
Published online August 31, 2007 in Wiley InterScience (www.interscience.wiley.com).

Higher crude prices have made it even more imperative that refiners blend low-quality and high-quality crudes optimally to maximize their margins. Mathematical modeling of crude blending results in bilinear terms, which combined with commonly used feed quality specifications, make crude scheduling a large, nonconvex, mixed-integer nonlinear optimization problem. The existing literature algorithms and software (DICOPT/GAMS and BARON/GAMS; Brooke et al., GAMS: a user's guide, GAMS, 1998) fail to solve practical instances of this difficult and useful problem. In this paper, we first enhance the practical utility of our previous crude scheduling algorithm by adding 15 properties and corresponding linearly additive indices, which are used in the refinery industry to ensure feed quality. Then, we propose some new iterative strategies to improve the robustness and solution quality of this algorithm. We also propose a partial relaxation strategy to increase its solution speed. We prove its enhanced performance using 24 industry-scale examples, and estimate bounds on solution quality. In contrast to existing algorithms or software that fail to solve most of these problems, our revised algorithm solves all problems successfully and gives profits within 6% of our computed upper bounds. © 2007 American Institute of Chemical Engineers *AIChE J,* 53: 2659–2680, 2007

Keywords: crude oil, scheduling, refinery, short-term scheduling, blending operations, mixed-integer linear program, crude quality

Introduction

Crude oil costs account for nearly 80% of a refinery's turnover.[1] Crude oils vary significantly in compositions, product yields, properties, and prices. Premium crudes such as West Texas Intermediate (WTI), Brent blend, etc. sell roughly $15 per barrel higher than the low-quality crudes such as Arabia heavy, Soudieh, etc. Most refineries use varying blends of several crude oils over time to exploit the higher margins of low-cost crude oils. With declining supplies and increasing prices of premium crude oils, the challenge facing the refiner is how to best exploit the greater margins of the low-cost crudes to increase profits. However, the low-cost crudes are almost always high in less-than-desirable components or traits such as sulfur, aromatics, high residue,

Correspondence concerning this article should be addressed to I. A. Karimi at cheiak@nus.edu.sg etc., and cause processing and/or product quality problems in crude distillation units (CDUs) and downstream units. Therefore, a key issue in the refinery business is to identify and process optimal blends of low-cost and premium crudes to minimize the operational problems yet maximize profit margins. As noted by Kelly and Mann,[1,2] scheduling of crude oil operations in a refinery is an critical task that can save millions of dollars per year, if done in an optimal manner. However, that is easier said than done.

This paper addresses the crude scheduling problem described by Reddy et al.[3] for a typical marine-access refinery. As mentioned by them, the task of crude oil scheduling in today's refinery is becoming increasingly complex and involves integrated management of activities such as crude arrivals, unloading, storage, blending, and charging or processing over days to weeks. Crude schedulers react to crude arrivals, schedule crude unloading, assign destination tanks for crude parcels to get clever crude blends, then mix these blends again to charge CDUs at appropriate feed rates to meet product demand and quality targets with minimum giveaways, minimize operational problems, and maximize profits. Clearly, the crude scheduler plays a critical role in determining the bottomline of a refinery. However, his/her task is by no means simple, as evident from the several attempts in the open literature at solving this problem at industrial scale optimally and with reasonable computational effort.

As noted by Reddy et al.,[3] the schedulers have an enviably tough job in a refinery, which has become even more difficult in recent years. They must continuously watch both crude oil movements and plant operation status, and match them to fluctuating demands. In most cases, under intense time-pressure and low inventory flexibility, the schedulers rely largely on experience and select the first feasible solution found by a spreadsheet model or some other method. Clearly, that leaves tremendous room for economic and operability improvement. Quantifiable economic benefits from advanced scheduling are improved options, increased utilization and throughput, intelligent use of low-cost crude stocks, capture of quality barrels, reduction of yield and quality giveaways, improved control and predictability of downstream production, reduced demurrage costs, reduced slop generation from changeovers, improved inventory and safety stock control, etc.

The presence of blending in crude oil operations that do not use charge tanks of specified compositions gives rise to bilinear terms in a mathematical formulation for scheduling. Additionally, discrete scheduling decisions such as selecting a tank to unload and the often complex nonlinear nature of crude properties and qualities make such a model difficult, nonlinear, nonconvex mathematical program or a nonconvex, mixed integer nonlinear program (MINLP). As we show later, even the best existing commercial solvers (e.g. DICOPT/GAMS[4] and BARON/GAMS[4]) are unable to solve these scheduling problems of practical, industrial size in reasonable time. Hence, in recent years, several researchers have addressed this problem in various forms and reported a variety of continuous-time and discrete-time models, often along with specialized algorithms to solve them with reasonable computational effort. However, the problem remains far from being solved satisfactorily. While attaining a guaranteed globally optimal solution to this nonlinear nonconvex problem is a challenging and important issue, getting good solutions for practical-size problems in reasonable times, even without the guarantee of global optimality, is an even more pressing issue at this time in our opinion. Thus, we make it clear that this paper focuses on the latter and not the former. While we do our best to get the best solutions, we are not aiming to get the guaranteed globally optimal solutions in this work.

To begin with, no existing work to our knowledge on crude oil scheduling has so far considered nonlinear crude/product properties. Thus, most existing models without charge tanks of known compositions have primarily been mixed integer bilinear programs (MIBLPs). With this in mind, the existing work involves four broad approaches, notwithstanding their nature of time representation (continuous-time vs. discrete-time). The first approach[5,6] comprises approximating the original MIBLP by a pure MILP. Apart from several deficiencies in the existing work, described in detail by Reddy et al.,[3] the main issue with this approach has been the composition discrepancy pointed out by Li et al.[7] and Reddy et al.[3] For instance, Lee et al.[5] used reformulation linearization technology (RLT) for bilinear terms, but this linearization approximation leads to a mismatch between the composition of crude delivered by a set of storage tanks to a CDU and that actually received by the CDU. To avoid this composition discrepancy, Li et al.[7] proposed an iterative decomposition algorithm that solves an alternating series of MILP approximations and NLPs. However, as commented by Kelly and Mann[1,2] and shown by Reddy et al.,[3] this decomposition approach may fail to obtain a feasible solution even when one exists. Moro and Pinto[8] presented an alternate approach for dealing with bilinear terms by using discrete values for continuous variables such as CDU feed rates. However, in addition to getting approximate optimal solutions, this discretization procedure increases problem size to an extent that makes it almost impossible to solve reasonably sized problems. Almost concurrently, Reddy et al.[3,9] proposed a novel rolling-horizon solution algorithm, which not only avoids composition discrepancy without increasing the problem size, but also avoids solving any NLP or MINLP. They used their algorithm to solve several moderate-size and one large practical-size problems. In spite of their relative success in solving this difficult problem, we show later that their algorithm can also fail to obtain feasible schedules and needs long solution times for solving large, practical-size problems. Therefore, it is fair to say that the existing open literature still lacks a reliable, robust, and efficient algorithm for this practical and important problem. For a detailed review of work on this problem, please refer Reddy et al.[3]

In this paper, we build on the model and algorithm of Reddy et al.[3] for scheduling crude oil operations to overcome some deficiencies of the existing work. As the first step, we identify and model the most important nonlinear crude properties that are crucial to crude distillation and downstream processing. Then, we develop a robust, improved, and more efficient version of their algorithm. Finally, we develop a procedure to estimate the quality of schedules by developing a rigorous upper bound for this nonconvex problem. We begin with an example to motivate our work, present a brief description of the problem, and then review some salient features of the formulation proposed by Reddy et al.,[3] which provides the basis for this work. We then extend that formulation to accommodate nonlinear crude properties. In subsequent sections, we develop strategies to improve robustness, quality, and solution speed of Reddy et al.'s algorithm,[3] and present ways to estimate solution quality by means of a tight upper bounding strategy. Finally, we use 24 large simulated examples to demonstrate numerically the robustness and effectiveness of our improved algorithm.

Problem Statement

Figure 1 shows a schematic of crude oil operations in a typical marine-access refinery. It comprises offshore facilities for crude unloading such as a single buoy mooring (SBM) or single point mooring (SPM) station, onshore facilities for crude unloading such as one or more jetties, tank farm consisting of crude storage and/or charging tanks, and processing units such as CDUs. The crude storage tanks hold crude blends rather than pure crudes. Their compositions vary with time. In this work, we assume that the refinery has no separate charging tanks; crude storage tanks also act as charging tanks. The unloading facilities supply crude to storage tanks via pipelines. The pipeline connecting the SBM/SPM station with crude tanks is called the SBM/SPM line, and it normally has a substantial holdup.

Two types of ships supply crudes to the refinery. Very large crude carriers (VLCCs) or ultra large crude carriers (ULCCs) carry multiple parcels of several crudes and dock at the SBM/SPM station offshore. Small vessels carry single crudes and berth at the jetties. The entire crude oil operation involves unloading and blending crudes from ships into various storage tanks at various times, and charging CDUs from one or more storage tanks at various rates over time. Thus, crude oil operations in a typical refinery involve both scheduling and allocation decisions. The problem can be stated as follows:

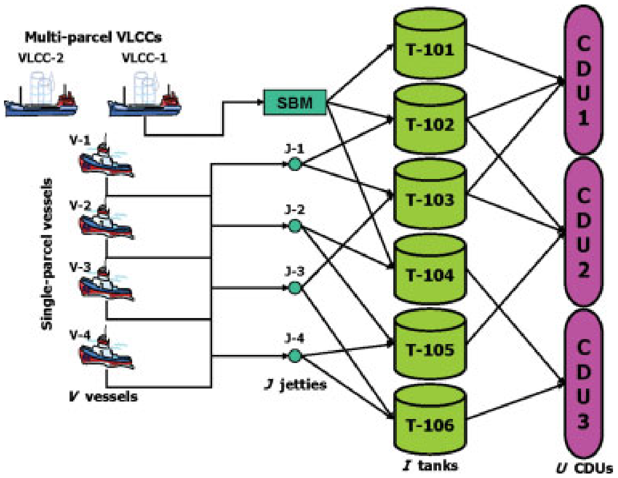

Figure 1. Schematic of crude oil unloading, blending, and processing.

[Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

Given:

1. *Crude delivery data:* Estimated arrival times of ships, their crude parcels, and parcel sizes.
2. *Maritime infrastructure:* Jetties, jetty-tank and SBM-tank connections, crude unloading transfer rates, and SBM pipeline holdup volume and its resident crude.
3. *Tank farm data:* Storage tanks, their capacities, their initial crude stocks and compositions, and crude quality specifications or limits.
4. *Crude processing data:* CDUs, processing rates, and crude quality specifications.
5. *Economic data:* Demurrage, crude changeover costs, safety stock penalties, crude margins, and product demands.

Determine:

1. Unloading schedule for each ship including the timings, rates, and tanks for all parcel transfers.
2. Inventory and crude concentration profiles of all storage tanks.
3. Charging schedule for each CDU including the feed tanks, feed rates, and timings.

Subject to the operating practices:

1. A storage tank cannot receive and feed crude at the same time.
2. Each tank needs 8 hours to settle and remove brine after each crude receipt.
3. Multiple tanks can feed a CDU simultaneously and vice versa.
4. Only one VLCC can dock at the SBM station at any time.
5. A parcel can unload to only one storage tank at any moment, but may unload to multiple tanks over time.
6. Sequence in which a VLCC unloads its parcels is known a priori. This is normally fixed when the VLCC loads its parcels and the refinery needs to specify that at the time of shipping.

Assuming:

1. Holdup of the SBM line is far smaller than a typical parcel size. Thus, only one crude resides in the SBM line at the end of each parcel transfer. Crude flow is plug flow in the SBM.
2. Holdup of the jetty pipeline is negligible.
3. Crude mixing is perfect in each storage tank.
4. Crude changeover times are negligible.
5. During operation, CDUs never shut down.

The objective of the scheduling problem is to maximize the gross profit, which is the revenue computed in terms of crude margins minus the operating costs such as demurrage, safety stock penalties, etc.

Base Formulation

While several discrete-time[3,5,7,10] and continuous-time[6,9] models exist in the literature for solving varying forms of the crude scheduling problem, we need one that will form the basis for our algorithmic improvements. While continuous-time formulations have advantages, Reddy et al.[3] showed that it is not fully certain, if they are clearly the best for this specific problem. We prefer to use the discrete-time model of Reddy et al.[3] because of some advantages. First, it embodies some features of a continuous-time formulation by allowing two parcels to transfer during any period. This partially obviates the need for a continuous-time model. Second, it accommodates some important structural and operational features of a marine-access refinery such as SBM/SPM, jetties, multiparcel VLCCs, multiple tanks feeding a CDU simultaneously and vice versa, brine settling, crude segregation, accurate demurrage accounting, crude changeovers, etc. Third, its use of 8-h time periods is quite practical and suitable for refinery operations, as it matches with the common brine settling time and shift-based operation of refineries. A shorter time-slot would make the discrete-time models difficult to solve, as the binary variables will increase considerably. On the other hand, a longer time-slot will reduce precision. However, note that Reddy's algorithm[3,9] is not constrained by the assumption of 8-h slots. Fourth, it accounts accurately for the significant holdup of the SBM line as done by treating the SBM holdup as a distinct single-crude parcel. When the first VLCC parcel is unloaded, this SBM parcel is ejected first. Similarly, when the last VLCC parcel is unloaded, a portion remains in the SBM line and becomes the next SBM parcel.

For the sake of completeness and easier comprehension of this work, we now present the key aspects of the MINLP model of Reddy et al.[3] Full details are available in their paper. The model represents time in terms of consecutive 8-h periods (t) and converts all ships and the initial crude holdup in the SBM line into a series of single-crude parcels (p) with appropriate arrival times. It uses three primary binary variables to model parcel-to-SBM/jetty, tank-to-SBM/jetty, and tank-to-CDU connections.

$$XP_{pt} = \begin{cases} 1 & \text{if parcel } p \text{ is connected for transfer during period } t \\ 0 & \text{otherwise} \end{cases}$$

$$XT_{it} = \begin{cases} 1 & \text{if tank } i \text{ is connected to receive crude during period } t \\ 0 & \text{otherwise} \end{cases}$$

$$Y_{iut} = \begin{cases} 1 & \text{if tank } i \text{ feeds CDU } u \text{ during period } t \\ 0 & \text{otherwise} \end{cases}$$

A major problem with some existing models is the composition discrepancy arising from the linearization of the following bilinear constraints related to crude blending in storage tanks:

$$FCTU_{iuct} = f_{ict} \cdot FTU_{iut} \quad (i,u) \in \boldsymbol{IU}, (i,c) \in \boldsymbol{IC} \tag{1}$$

$$VCT_{ict} = f_{ict} \cdot V_{it} \quad (i,c) \in \boldsymbol{IC} \tag{2}$$

where f_{ict} denotes the fraction of crude c in tank i at the end of period t, $FCTU_{iuct}$ is the amount of crude c from tank i to CDU u during period t, FTU_{iut} is the total amount of crude from tank i to CDU u during t, VCT_{ict} is the amount of crude c in tank i at the end of period t, and V_{it} is the total amount of crude in tank i at the end of period t; $\boldsymbol{IC} = \{(i,c) \mid \text{tank } i \text{ can hold crude } c\}$; $\boldsymbol{IU} = \{(i,u) \mid \text{tank } i \text{ can feed CDU } u\}$. Equations 1 and 2 are bilinear, f_{ict} is unknown, except at the start of the scheduling horizon (or equivalently the first period), when the initial crude compositions of tanks were known. When these are approximated by linear constraints, the composition of crude received by a CDU may not match with that supplied by the tanks, which has been termed as composition discrepancy by Li et al.[7] and Reddy et al.[3] To avoid both MINLP solution and composition discrepancy, Reddy et al.[3] developed an effective heuristic iterative strategy to obtain good schedules, which we discuss later.

Motivation

Although the model and algorithm of Reddy et al.[3,9] are the best in the literature so far for this MINLP problem, they still have some shortcomings. In addition to linear crude properties, long solution times, and lack of quality estimates, a major issue is robustness or the ability to give a feasible solution in large and difficult problems. Consider the following example to illustrate that their algorithm may fail to obtain a feasible solution, even when one exists.

A refinery has one SBM pipeline, four storage tanks (T1–T4), two CDUs (CDU1 and CDU2), and processes four crudes (C1–C4) that are segregated into two classes (CL1 and CL2). C1 and C2 belong to CL1, can be stored in T1 and T4, and can be processed in CDU1. Similarly, C3 and C4 belong to CL2, can be stored in T2 and T3, and can be processed in CDU2. The scheduling horizon is 3 days, in which one VLCC carrying three crude parcels (300 kbbl C1, 300 kbbl C4, 350 kbbl C3, unloaded in that sequence) arrives at time zero. At time zero, the SBM pipeline is holding 10 kbbl C2 from the last parcel. For simplicity, we consider only one key component, whose allowable ranges are [0.0045, 0.006] and [0.014, 0.0153] vol % for CDU1 and CDU2 respectively. Other data are shown in Table 1.

Reddy's algorithm[3,9] fails to find a feasible solution after four iterations. In the four iterations, the algorithm fixes the schedule up to period 4, as shown in Table 2. All parcels have been unloaded in this partial schedule, which is now frozen for the fifth iteration. At the end of period 4, T2 has 200 kbbl of crude with 0.015166 vol % of key component and T3 has 790 kbbl with 0.015437 vol % of key component. T2 satisfies the key component specification of [0.014, 0.0153] vol % for CDU2, but T3 does not. Since no crudes will arrive, CDU2 must use crudes in T2 and T3 for periods 5–9. Furthermore, T3 just finished receiving a parcel in period 4, hence it cannot supply crude in period 5. Now, the total demand of CDU2 is 550 kbbl, of which 200 kbbl has been met until period 4 and 350 kbbl remain to be processed during periods 5–9. Since we have a low-quality crude in T3, the best scenario is to feed 50 kbbl to CDU2 from T2 in period 5 and then use a mixture from T2 and T3 for the remaining periods. However, the lower limit on crude holdup in T2 is 50 kbbl, hence 100 kbbl of crude is available from T2 for the last four periods. Thus, our best chance for meeting the demand and quality constraints in periods 6–9 is to use 100 kbbl from T2 and 200 kbbl from T3. Unfortunately, this crude mixture has 0.015347 vol % of key component, which is unacceptable crude quality for CDU2, and the algorithm fails in iteration 5. However, when we solve this example manually, we do obtain a feasible solution shown in Table 2.

Interestingly, both DICOPT/GAMS[4] and the iterative method of Li et al.[7] fail to solve this example. Thus, there is a need to develop a tailor-made, better, robust, and more efficient algorithm for this difficult crude oil scheduling problem.

Table 1. Data for Example 1

Tank & CDU	Capacity (kbbl)	Heel (kbbl)	Initial Inventory (kbbl)	Allowable Crude (Class)	Initial Crude Amount (kbbl)		Crude Concentration Range (Min–Max)		Total Demand (kbbl)	Processing Limits (kbbl/period) Min−Max	Property Specification Range (Min–Max)
					C1 or C3	C2 or C4	C1 or C3	C2 or C4			
T1	700	50	300	C1,C2 (1)	200	100	0.2–0.8	0.2–0.8	–	–	–
T2	700	50	300	C3,C4 (2)	100	200	0.2–0.8	0.2–0.8	–	–	–
T3	900	50	250	C3,C4 (2)	50	200	0.0–1.0	0.0–1.0	–	–	–
T4	700	50	300	C1,C2 (1)	130	170	0.2–0.8	0.2–0.8	–	–	–
CDU1	–	–	–	C1,C2 (1)	–	–	0.3–0.7	0.3–0.7	550	50–100	0.0045–0.0060
CDU2	–	–	–	C3,C4 (2)	–	–	0.0–1.0	0.0–1.0	550	50–100	0.0140–0.0153

Crude	Property Specification	Margin ($/bbl)
C1	0.0050	3.0
C2	0.0060	4.5
C3	0.0165	5.0
C4	0.0145	6.0

Parce–Tank flow rate:10–400 kbbl/period

Tank–CDU flow rate: 0–100 kbbl/period

Damurrage: 100k$/period

Changeover loss: $5000/instance

Safety stock penalty: 0.2 $/bbl/period

Desired safety stock: 1200 kbbl

One VLCC (300 kbbl C1,300 kbbl C4,340 kbbl C3) arrives at time zero

Extensions of Reddy's Model

The model of Reddy et al.[3] needs two refinements to extend its practical utility for scheduling crude oil operations. The first relates to the uncontrolled changes in CDU feed rates. Reddy et al.[3] allowed the CDU feed rates to fluctuate uncontrolled between successive periods. For instance, in the schedule for their motivating example, the feed to CDU2 is 50 kbbl in period 7, while it is 100 kbbl in period 8; which is a 100% change in 8 h. Clearly, such drastic changes in feed rates may disrupt CDU operation, may generate off-spec distillation cuts, and may even be impossible to achieve without destabilizing the column. They can be disallowed by simply adding the following two constraints:

$$\gamma_u^L FU_{ut} \leq FU_{u(t+1)} \leq \gamma_u^U FU_{ut} \qquad \text{(3a,b)}$$

where FU_{ut} denotes total amount of crude fed to CDU u during period t. Parameters γ_u^L and γ_u^U can be suitably set to control period-to-period changes in crude feed flows.

The second model refinement is about ensuring the quality of feeds to CDUs. This is a critical operating requirement in practice, as a feed with poor quality can seriously disrupt the operation of a CDU and even downstream units. Ensuring acceptable feed qualities to CDUs is especially critical in this problem, where the goal is to exploit cheap, poor-quality crudes to enhance profitability. A variety of crude properties are used in practice such as specific gravity, sulfur, nitrogen, oxygen, carbon residue, pour point, flash point, nickel, Reid vapor pressure, asphaltene, aromatics, paraffins, naphthene, wax, and viscosity. Reddy et al.[3] used the idea of key component concentrations to model these crude quality specifications. They imposed the following constraints to specify acceptable lower and upper limits on the concentration of a key component.

$$\theta_{ku}^L FU_{ut} \leq \sum_i \sum_c FCTU_{iuct}\theta_{kc} \leq \theta_{ku}^U FU_{ut}$$

$$(i,u) \in \boldsymbol{IU}, (i,c) \in \boldsymbol{IC} \qquad \text{(4a,b)}$$

where θ_{kc} is the concentration of a key component k in crude c, and θ_{ku}^L and θ_{ku}^U respectively are the lower and upper acceptable limits on that in the feed to CDU u. While Reddy et al.[3] did mention that the above constraints can be applied to most common crude quality specifications; their approach has some limitations. First, it assumes (Eq. 4) the key component concentration to be linearly additive. However, many crude properties (e.g. viscosity) involve highly nonlinear mixing rules, where Eq. 4 cannot work. Second, it also assumes that the key component concentrations are additive on a volume basis. Crude properties such as density and sulfur are additive on a weight basis. Therefore, we need a better approach to address crude quality specifications.

To this end, we note that a linear blending index exists for almost every crude property involving nonlinear mixing correlations. Furthermore, these blending indices are either volume-based or weight-based. Table 3 provides the blending indices and their additive bases for the 15 most commonly used crude properties. Thus, instead of key component concentrations, we use an appropriate blending index for each

Table 2. Schedules for Example 1

		Crude Amount [to CDU No.] (from Vessel No.) in kbbl for Period								
Algorithm	Tank	1	2	3	4	5	6	7	8	9
RA	1	−50.0[1]	−50.0[1]	−8.3[1]	−8.3[1]	−8.3[1]	−8.3[1]	−8.3[1]	−8.3[1]	−100.0[1]
	2			−50.0[2]	−50.0[2]	−50.0[2]				

	3	−50.0[2]	−50.0[2]	+300.0(3)
				+10.0(3)
					+330.0(4)		−100.0[2]	−50.0[2]	−50.0[2]	−100.0[2]

	4	+10.0(1)
		+300.0(2)
				−50.0[1]	−50.0[1]	−50.0[1]	−50.0[1]	−50.0[1]	−50.0[1]	

Manual	1	−50.0[1]	−50.0[1]	−50.0[1]	−50.0[1]	−10.0[1]	−10.0[1]	−10.0[1]	−10.0[1]	−10.0[1]
	2		−50.0[2]	−50.0[2]	+340.0(4)					
	3	−50.0[2]	+300.0(3)		−50.0[2]	−50.0[2]	−50.0[2]	−50.0[2]	−100.0[2]	−100.0[2]

	4	+10.0(1)
		+290.0(2)
			+10.0(2)			−60.0[1]	−60.0[1]	−60.0[1]	−60.0[1]	−60.0[1]

RRA	1	−50.0[1]	−50.0[1]	−54.9[1]	−15.8[1]	−15.8[1]	−15.8[1]	−15.8[1]	−15.8[1]	−15.8[1]
	2		−50.0[2]	−50.0[2]	+340.0(4)					
	3	−50.0[2]	+300.0(3)		−50.0[2]	−50.0[2]	−55.9[2]	−67.1[2]	−80.5[2]	−96.6[2]

	4	+10.0(1)
		+300.0(2)
					−50.0[1]	−50.0[1]	−50.0[1]	−50.0[1]	−50.0[1]	−50.0[1]

"−" sign represents delivery to [CDU], "+" sign represents receipt from (Parcels).

crude property and define the following constraint to accommodate weight-based blending indices as follows:

$$\theta_{ku}^{L}\left(\sum_{i}\sum_{c} FCTU_{iuct}\rho_c\right) \le \sum_{i}\sum_{c} FCTU_{iuct}\rho_c\theta_{kc}$$
$$\le \theta_{ku}^{U}\left(\sum_{i}\sum_{c} FCTU_{iuct}\rho_c\right) \quad (i,u)\in \boldsymbol{IU}, (i,c)\in \boldsymbol{IC} \qquad \text{(5a,b)}$$

where ρ_c as the density of crude c, and θ_{kc} now refers to the blending index for property k of crude c in Eqs. 4 and 5.

With these extensions, we obtain a revised MINLP model of Reddy et al.[3] with the same scheduling objective as in Reddy et al.[3] We call it **F**. The remainder of this paper aims to solve **F** by proposing improved algorithms.

Improving Robustness and Efficiency

We first review the key steps of the algorithm of Reddy et al.,[3] which we call RA (Reddy's algorithm) for the sake of brevity. Recall that Reddy et al.[3] used 8-h periods in their discrete-time model. The success of their algorithm in eliminating composition discrepancy without solving a nonlinear problem hinges on recognizing that Eqs. 1 and 2 would be linear, if we knew f_{ict} or tank composition. If a tank receives no crude during several periods, then its composition cannot change. We call the sets of such periods as zones of constant

Table 3. Crude Properties, Their Relevance, and Corresponding Indexes and Correlations

Crude Property	Blending Index	Addition Base	Relevance to (Important for)	Index Correlation
Specific Gravity (SG)	DNI	Volume	Crudes, all products	1/SG
Sulfur	SULI	Weight	Crudes, all products	Weighted average
Nitrogen	NITI	Weight	Crudes, residue streams (550+°C), vacuum gas oil (370–550°C)	Weighted average
Carbon Residue	CRI	Weight	Crudes, residue streams (550+°C), vacuum gas oil (370–550°C)	Weighted average
Pour Point (PP °C)	PIndex	Volume	Crudes, all products	316200 × exp(12.5 × Log(0.001(1.8 × PP + 491.67))
Freeze Point (°C)	FreezeIndex	Volume	Kerosene(150–280°C)	3162000 × exp(12.5 × Log(0.001(1.8 × Freeze Point + 491.67))
Flash Point (FLP °C)	FP Index	Volume	All products	exp((−6.1184 + (2414/(FLP + 230.56))) × log(10))
Smoke Point (SMP mm)	SMI	Volume	Kerosene (150–280°C)	−362 + 3200/log(SMP)
Nil	NiIndex	Weight	Crudes, residue streams (550+°C), Vacuum gas oil (370–550 °C)	Weighted average
Reid Vapor Pressure (RVP Bar)	RVI	Volume	Curdes, products up to naphtha range boiling below 200°C	exp(1.14 × log(100 × RVP)
Asphaltenes	ASPI	Weight	Curdes, residue streams (550+°C), vacuum gas oil (370–550°C)	Weighted average
Aromatics	AROI	Volume	Naphtha range boiling below 200°C	Volumetric average
paraffins	PARI	Volume	Naphtha range boiling below 200°C	Volumetric average
Naphthenes	NAPHI	Volume	Naphtha range boiling below 200°C	Volumetric average
Viscosity @ 50°C (Visc_cst)	ViscIndex	Weight	Crudes, residue streams (550+°C), Vacuum gas oil (370–550°C)	79.1 + 33.47 × (log(log(Visc_cst + 0.8)/log(10))/log(10))

Index correlations from Reddy (Singapore Petroleum Company Pte Ltd.)

composition, which can be easily identified from the arrival times of ships as explained by Reddy et al.[3] Reddy et al.[3] combine this observation with the fact that tank compositions are known at time zero to divide the scheduling horizon into two distinct blocks for each tank. The front block (in time) in which the tank composition is known and constant and the rear one in which it is unknown. For all periods in the front block and the first period in the second block, they use Eqs. 1 and 2 with known f_{ict}, and for the remaining periods, they ignore Eqs. 1 and 2. Now, as the first step of their algorithm, they identify the initial zone of periods for each tank for which the tank composition is constant and known. The length of this block may vary from tank to tank. For the remaining periods, the composition in each tank is unknown. They use Eqs. 1 and 2 for this first zone of periods, drop Eqs. 1 and 2 for the remaining periods, and solve the MILP. Because no linearization was used in the first zone for each tank, the corresponding MILP solution is free of composition discrepancy. Then, they identify the longest zone of periods for which compositions are known in *all* tanks. They freeze the schedule for that zone and repeat the procedure again. This rolling-horizon type of strategy ensures that the final schedule has no composition discrepancy.

In the Motivation section, we showed that RA may fail to get a feasible schedule. When RA fails to find a feasible schedule at an iteration, one possible culprit is the early (frozen) part of the schedule. This frozen schedule involved fixing of binary variables, and their fixed values may mean an infeasible combination. RA lacks a mechanism to retract from these infeasible combinations. Thus, a simple way to resurrect RA would be to eliminate some infeasible combinations of binary variables by using the well-known integer cut from Balas and Jeroslow.[11] Using this basic idea, we now develop a backtracking strategy that allows us to identify and remove infeasible combinations of frozen binary variables, which enables us to proceed forward to obtain a feasible solution.

Backtracking Strategy

RA moves forward by exactly one block of periods at each iteration, where that block represents the first one or more contiguous periods after the frozen schedule, in which tank compositions are constant and known. Let us call these blocks as composition-based blocks. Thus, if RA fails to get a feasible solution at iteration *n*, then the fixed values of binary variables in one or more of previous blocks ($n - 1$ to 1) or some combinations thereof may cause infeasibility. Since we do not know exactly which combination is infeasible, one option is to include in the integer cut all the binary variables fixed so far from blocks 1 to ($n - 1$). Clearly, this may work for small *n*, because the cut would involve only a few binary variables. However, this would be inefficient for large *n*. An alternate option is to consider previously frozen blocks one at a time backward. In other words, if the algorithm fails to give a feasible solution in block *n*, then we retract to block ($n - 1$), impose an integer cut for the previous binary combination in block $n - 1$, and then resume the algorithm from block ($n - 1$). If solving block ($n - 1$) does not yield a feasible solution, then we retract to block $n - 2$, and repeat the same procedure. We continue to retract, until we get a feasible solution, and then proceed forward as in RA. This method would be effective, when the block that caused infeasibility is not too far away before block *n*.

Often in RA, a block comprises one period only, so the aforementioned strategy will require backtracking by single periods and that would be slow. Therefore, we define new blocks that are different from those used by RA. To this end, we define blocks based on the scheduled arrival times of vessels, but ensure that each vessel unloads entirely in exactly one block. We call these as vessel-based blocks to differentiate them from the composition-based blocks used by RA. 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, if we are sure that the first vessel will complete unloading before the second arrives. If this cannot be guaranteed, then the first block will extend to the arrival of the third vessel. We continue likewise to define all blocks such that no vessel will unload in more than one block. For example, if four vessels arrive at the starts of periods 4, 5, 13, and 17, and each vessel needs at least four periods to unload, then we get four blocks. Block 1 spans periods 1–3, block 2 spans 4–12, block 3 spans 13–16, and block 5 spans 17–last. Note that the model of Reddy et al.[3] assumes a priori the periods in which a vessel could possibly unload, which enables us to define the blocks in the aforementioned manner. Since it estimates these periods in a conservative manner, such blocks can always be defined.

Variables for Integer Cuts

Even if we implement cuts in one block at a time using the blocks defined above, the cuts can still involve large numbers of variables. To strengthen them, we need an effective strategy for selecting the best set of variables. The first key consideration in this selection is the number of variables. The fewer the variables in the cuts, the tighter the cuts; therefore we would like to pick the fewest variables. Second, the variables must have a direct impact on the feasibility of a schedule later. If a decision has no direct impact on the future feasibility of a schedule, then it is not critical, and its corresponding variables serve no useful purpose in the cut. This is because we are trying to correct the frozen decisions that led to infeasibility later in time in the schedule.

With the above two criteria in mind, let us examine the binary variables in the model of Reddy et al.,[3] namely XP_{pt}, XT_{it}, and Y_{iut}. XP_{pt} denotes parcel-to-SBM/jetty connections. These connections must occur for every parcel to be unloaded. These connections have limited direct impact on the compositions of crude stocks in tanks, because the tank that the parcel is unloaded into also depends on XT_{it}. Thus, XP_{pt} has little direct effect on the feasibility of subsequent scheduling decisions, and it should not be in the cuts. A similar argument can be made for Y_{iut}. The tanks that are used to feed CDUs at a given time have limited direct effect on the future compositions of tanks and feasibility of later schedule. Moreover, the number of Y_{iut} is always more than that of XT_{it} in a vessel-based block, because Y_{iut} may be nonzero even in the constant-composition zones, while XT_{it} must be zero in those zones. Thus, we need not include them in cuts. In contrast to these two binary variables, XT_{it} denotes tank-to-SBM/jetty connections. Effectively, these decisions control which tanks receive crudes and largely determine the qualities of crude stocks in the tanks, while XP_{pt} and Y_{iut} do not at least directly. This further illustrates that XT_{it} has a more direct effect on the future composition of tanks and the feasibility of subsequent schedule compared to XP_{pt} and Y_{iut}. A poor blending decision can prove costly later, as it may become impossible to blend the stocks at hand to satisfy the feed quality requirements of CDUs. Thus, it is clear that some combinations of XT_{it} could lead to infeasibility in a problem and we must eliminate these using integer cuts to achieve feasible schedules.

If we include all XT_{it} variables in the integer cuts, then we get,

$$\sum_{(i,t)\ni XT_{it}=1} XT_{it} - \sum_{(i,t)\ni XT_{it}=0} XT_{it} \leq NZ - 1 \qquad (6)$$

where NZ is the number of terms in the first summation. As stated earlier, Eq. 6 is obtained directly from Balas and Jeroslow.[11] It simply eliminates the particular combination of XT_{it} values from being considered again in the MILP. It is possible to reduce the number of binary variables in Eq. 6 in some cases. Often, refineries segregate crudes into various classes, store them in different tanks, and process them in different CDUs. If we can somehow identify the class/es of crudes or tanks that caused infeasibility, then we can include in Eq. 6 only the variables associated with those crudes or tanks. We now derive a procedure for doing this for a refinery practicing such a segregation of crudes and tanks.

Recall that RA uses Eqs. 1 and 2 with known f_{ict} in the current constant-composition block to avoid composition discrepancy. When it fails to find a feasible solution at an iteration, a solution to these equations cannot exist without composition discrepancy. If we can identify the equations that cannot be solved, then we would know the tanks that are causing the infeasibility. To this end, we relax Eq. 1 as follows by using two nonnegative slack variables:

$$FCTU_{iuct} = f_{ict}FTU_{iut} - u^{-}_{iuct} + u^{+}_{iuct} \quad (i,u) \in \boldsymbol{IU}, (i,c) \in \boldsymbol{IC} \quad (7)$$

If any of u^{-}_{iuct} or u^{+}_{iuct} is nonzero, then it is clear that composition discrepancy exists. Thus, by adding them to Eq. 1, we can now essentially ask RA to get a solution with composition discrepancy. Note that introducing slack variables in Eq. 2 is not necessary for such a solution. If we can get such a solution, then the nonzero slack variables in that solution will tell us the class or classes of crudes that are violating Eq. 1.

There are two ways to achieve the above solution with nonzero slack variables. One is to add a penalty for u^{-}_{iuct} and u^{+}_{iuct} in the original scheduling objective. However, choosing a suitable penalty with proper weights that do not bias the solutions is a nontrivial problem. We found it hard to select a penalty that is suitable for different cases. The second method is to solve the same model but with a different objective that tries to force the slack variables to zero. The simplest such objective is to minimize the sum of slack variables.

$$\min P = \sum_{i}\sum_{u}\sum_{c}\sum_{t}(u^{+}_{iuct} + u^{-}_{iuct}) \qquad (i,u) \in \boldsymbol{IU}, (i,c) \in \boldsymbol{IC} \quad (8)$$

Recall that we defined **F** as the revised MINLP model of Reddy et al.[3] with the original scheduling objective. Now we define **FL** as **F** augmented with Eqs. 1 and 2 for the periods of known crude compositions in tanks and no Eqs. 1 and 2 for remaining periods. Note that **FL** is MILP model of RA used to solve **F**. Then, we define **FP** as **FL** but with Eq. 8 as the objective in place of the original scheduling objective. Note that **FP** is also an MILP problem and its solution may be computationally expensive. Since the goal of **FP** is just to identify a set of nonzero slack variables, obtaining an exact optimal solution is not necessary in all cases. In order to control solution time, we specify an upper limit on time for solving **FP**. By doing this, we may obtain an optimal solution, feasible solution, or no solution at all for **FP**. With this, we are now in a position to fully describe a revised version of RA, which is designed for obtaining feasible schedules that RA fails to get.

Revised Reddy's Algorithm (RRA)

Let α denote a composition-based block in RA and β denote a vessel-based block defined in the section on backtracking strategy. The revised algorithm (RRA) follows all the steps in RA, except when it fails to get a feasible solution at some iteration. Let α be the current composition-based block that causes a solution failure for RA and β be the vessel-based block to which α belongs. Then, we add the slack variables for block α in **FL** to get **FP** and solve it to identify the nonzero slack variables and the corresponding class/es of crudes that fail to satisfy Eq. 1. For each such crude class, we write one separate integer cut (that includes all crudes in that class) for all periods in block β, but before block α. Then, we update **FL** by adding these cuts permanently. Now, we free all variables before block α in block β and return to the beginning of block β. With the schedule before block β frozen, we restart RA as per normal. Note that when RA fails at the first composition-based block α of a block β, then we have examined all combinations of integer variables in block β and we should backtrack to block $\beta - 1$. We repeat this procedure, until we get the entire schedule. Figure 2 shows the detailed algorithm.

Example 1

Let us illustrate RRA using the motivating example. In that example, one VLCC carrying three parcels arrives at the beginning of the scheduling horizon, thus we have only one vessel-based block (β = 1), which includes periods 1–9. The first four iterations of RRA proceed smoothly, i.e. feasible solutions are readily obtained: α = 1 includes period 1 only, α = 2 includes period 2, α = 3 includes period 3, α = 4 includes period 4, and β = 1 for all iterations. At iteration 5, α = 5 includes periods 5–9. With the schedule frozen for periods 1–4, **FL** fails to give a solution for α = 5. Then, we replace Eq. 1 by the following constraints and solve **FP**:

$$FCTU_{iuct} = f_{ict}FTU_{iut} - u^{-}_{iuct} + u^{+}_{iuct} \quad (i,u) \in \boldsymbol{IU}, (i,c) \in \boldsymbol{IC} \quad (9)$$

Solving **FP** gives us $u^{+}_{3236} = 2.16$, $u^{+}_{3237} = 4.844$, $u^{-}_{3246} = 2.16$, and $u^{-}_{3247} = 4.844$. Since these involve crudes 3 and 4, which belong to Class 2, we can conclude that there is a problem with the way that Class 2 crudes were blended during periods 1–4. To remove this infeasible blending combination, we then use the following integer cut for Class 2.

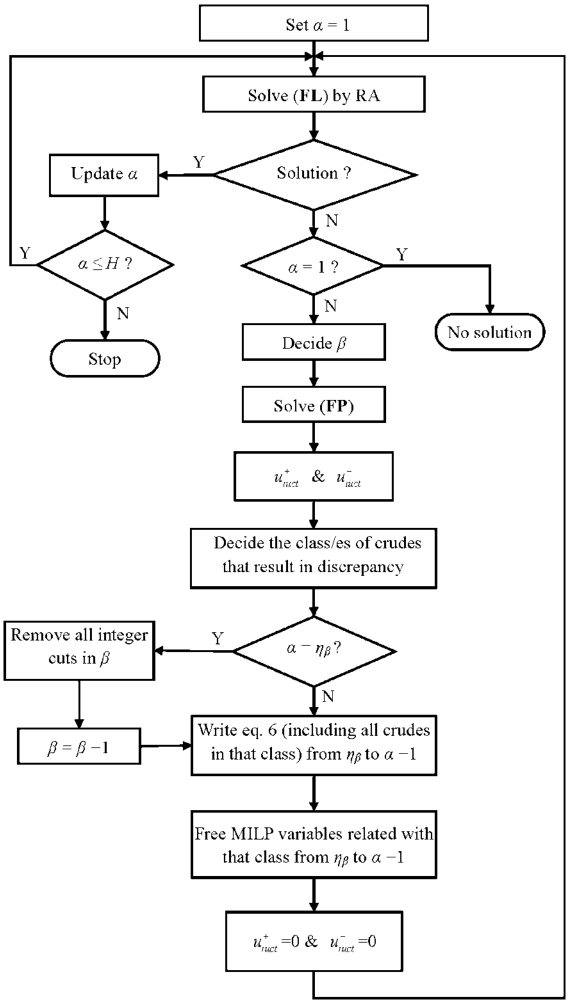
Set $\alpha = 1$
Solve (**FL**) by RA
Solution ?
Y
N
Update α
$\alpha \leq H$?
Y
N
Stop
$\alpha = 1$?
Y
N
No solution
Decide β
Solve (**FP**)
u^{+}_{iuct} & u^{-}_{iuct}
Decide the class/es of crudes that result in discrepancy
$\alpha - \eta_\beta$?
Y
N
Remove all integer cuts in β
$\beta = \beta - 1$
Write eq. 6 (including all crudes in that class) from η_β to $\alpha - 1$
Free MILP variables related with that class from η_β to $\alpha - 1$
$u^{+}_{iuct} = 0$ & $u^{-}_{iuct} = 0$

Figure 2. Flow chart for RRA [Revised Algorithm of Reddy et al.[3,9]].

α denotes a composition-based block, β denotes a vessel-based block, η_β denotes the first composition-based block in block β. H denotes the scheduling horizon.

$$XT_{33} + XT_{34} - \sum_{t=1}^{4} XT_{2t} - XT_{31} - XT_{32} \leq 1 \qquad (10)$$

Then, we fix all XT_{it} (t = 1–4) for Class 1 crudes in **FL**, add Eq. 10 to **FL**, and begin RRA from period 1 again. RRA proceeds to completion without any failure. Table 2 also shows the final schedule without any composition discrepancy.

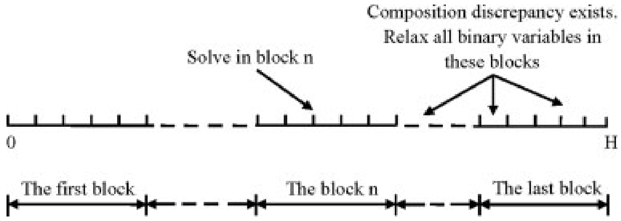
Composition discrepancy exists. Relax all binary variables in these blocks
Solve in block n
0
H
The first block
The block n
The last block

Figure 3. Schematic of RRA-P (Partial Relaxation Strategy).

Partial Relaxation Strategy

Consider RRA at an arbitrary iteration such that the schedule before composition-based block α is fixed. While the MILP solved at this iteration by RRA will treat all binary variables within α and beyond as binary, it will guarantee composition consistency (i.e. no discrepancy) in block α only. In other words, the portion of this MILP solution for blocks beyond α may have composition discrepancy and is of no use at this iteration. In other words, computational effort to get an integer-feasible solution of the MILP for blocks beyond α is not warranted. In fact, we can relax the binary variables for these blocks without affecting the solution for block α. In principle, we can do this for all blocks beyond α, but Eq. 6 involves binary variables from vessel-based blocks and may get partially relaxed, losing its cutting

Table 4. Vessel Arrival Data for Examples 2–24

Example	Arrival Period	Vessel (Crude-Parcel Size kbbl or kton*)
2	1	V1 (C1-3), V2 (C1-3)
	2	V3 (C1-3), V4 (C1-3)
	3	V5 (C1-5), V6 (C1-5),V7 (C1-3), V8 (C1-3)
	4	V9 (C1-3)
	5	V10 (C2-5), V11 (C6-5), V12 (C2,3.5), V13 (C4-3.5)
	6	V14 (C1-3), V15 (C1-3)
	7	V16 (C4-3), V17 (C2-1.5, C6-1.5)
3	1	VLCC–1 (C2-10, C3-250, C4-300, C5-190)
	14	VLCC–2 (C5-10, C6-250, C3-250, C8-240)
4–6	1	VLCC–1 (C2-10, C3-250, C4-300, C5-190)
	14	VLCC–2 (C5-10, C6-250, C3-250, C8-240)
7–11, 17–19	1	VLCC–1 (C2-10, C3-350, C4-200, C5-300)
	16	VLCC–2 (C5-10, C6-200, C8-250, C3-240)
	28	VLCC–3 (C3-10, C6-250, C2-250, C7-190)
12	1	VLCC–1 (C2-10, C3-350, C4-200, C5-300)
	16	VLCC–2 (C5-10, C6-200, C8-250, C3-240)
	28	VLCC–3 (C3-10, C6-250, C2-250, C7-190)
13	1	VLCC–1 (C2-10, C3-350, C4-200, C5-300)
	16	VLCC–2 (C5-10, C6-200, C8-250, C3-240)
	28	VLCC–3 (C3-10, C6-250, C2-250, C7-190)
14	6	VLCC–1 (C2-10, C3-250, C4-200, C5-350)
	21	VLCC–1 (C5-10, C6-250, C8-200, C3-240)
	33	VLCC–3 (C3-10, C6-250, C2-300, C7-190)
15	4	VLCC–1 (C7-10, C1-250, C6-200, C5-240)
	19	VLCC–2 (C5-10, C7-250, C4-300, C2-190)
	31	VLCC–3 (C2-10, C8-300, C3-200, C6-250)
16 & 20–21	1	VLCC–1 (C2-10, C6-100, C8-100, C4-90)
	3	V1 (C2-125)
	4	V2 (C5-125), V3 (C3-100)
	5	V4 (C7-120)
	21	VLCC–2 (C4-10, C8-130, C3-120, C2-100)
	23	V5 (C6-100), V6 (C1-90)
	24	V7 (C7-125)
22–23 & [24]	1[4]	VLCC–1 (C3-10, C5-200, C7-250, C2-190)
	4[7]	V1 (C1-150), V2 (C6-220)
	5[8]	V3 (C2-180), V4 (C4-150)
	6[9]	V5 (C8-230)
	20	VLCC–2 (C2-10, C1-300, C7-240, C5-190)
	20	V6 (C4-160), V7 (C6-210)
	21	V8 (C7-270)
	32	V9 (C8-200), V10 (C4-250), V11 (C2-180), V12 (C3-150)
	46	VLCC–3 (C5-10, C2-200, C8-170, C3-180)
	47	V13 (C1-300), V14 (C7-250)

Arrival periods in brackets are for Example 24, while the alternatives are for Examples 22 and 23
*kton for Example 2.

Table 5. Tank Capacities, Heels, and Initial Inventories for Examples 2–24

	Capacity (kbbl or kton*)									Heel (kbbl or kton*)			
Tank	Ex 2	Ex 3–4	Ex 5–9 & 13	Ex 10–11 & 17–18	Ex 12	Ex 14–15	Ex 16	Ex 19–21	Ex 22–24	Ex 2	Ex 3–11,13 & 16–19	Ex 15 & 20–21	Ex 12, 14 & 22–24
T1	25	570	570	570	980	600	570	570	700	0	60	50	60
T2	25	570	570	570	980	600	570	570	700	0	60	50	60
T3	25	570	570	570	980	600	570	570	700	0	60	50	60
T4	25	980	980	980	980	600	570	980	700	0	110	50	60
T5	40	980	980	980	980	600	570	570	700	0	110	50	60
T6	40	980	570	570	980	600	570	570	700	0	60	50	60
T7	40	570	570	570	980	600	570	570	700	0	60	50	60
T8	0	570	570	980	980	600	570	980	700	0	60	50	60

	Allowable Crude (Class)		Initial Inventory (kbbl or kton*)									
Tank	Ex 2	Ex 3–24	Ex 2	Ex 3–9	Ex 10–11	Ex 12	Ex 13 & 16	Ex 14	Ex 15	Ex 17–18	Ex 19–21	Ex 22–24
T1	C1 (1)	C1-C4 (1)	5	350	400	320	350	420	300	450	350	350
T2	C2 (1)	C5-C8 (2)	6	400	400	400	400	320	350	400	400	300
T3	C3 (1)	C5-C8 (2)	7	350	350	400	350	400	250	350	350	350
T4	C1 (1)	C5-C8 (2)	8	950	950	900	950	280	400	950	950	250
T5	C1 (1)	C5-C8 (2)	8	300	300	280	300	300	300	300	300	210
T6	C2, C4, C6 (1)	C1-C4 (1)	20	80	80	80	80	100	160	80	80	80
T7	C3 (1)	C1-C4 (1)	10	80	80	80	80	80	80	80	80	80
T8	–	C1-C4 (1)	0	450	450	450	450	250	350	450	450	450

*kton for Example 2.

power completely. Therefore, we can relax only the binary variables beyond the vessel-based block β to which α belongs. Figure 3 shows this partial relaxation strategy and we call this algorithm as RRA-P.

Algorithm Evaluation

We use 24 examples (including Example 1 or the motivating example) of varying sizes and features to evaluate the performance of RRA and RRA-P against those of RA (Reddy et al.[3]), LA (Li et al.[7]) and commercial MINLP codes such as DICOPT/GAMS[4] and BARON/GAMS[4] (in solving **F**). Tables 4–9 show the data for these examples. While Examples 1–16 and 22–24 use only one, Examples 17–21 use fifteen specifications on crude feed quality. These properties and their corresponding linearly-additive indexes are listed in Table 3. While Examples 1 and 2 are relatively small-size, 3–6 are medium-size, and 7–24 are large-size. Examples 7–24 are

Table 6. Initial Crude Amounts (kbbl or kton*) for Examples 2–24

	Ex 2				Ex 3–9,13, 16 & 19–21				Ex 10–11				Ex 17–18			
Tank	C1 or C5	C2	C3 or C6	C4	C1 or C5	C2 or C6	C3 or C7	C4 or C8	C1 or C5	C2 or C6	C3 or C7	C4 or C8	C1 or C5	C2 or C6	C3 or C7	C4 or C8
T1	5	–	–	–	50	100	100	100	100	100	100	100	90	120	120	120
T2	–	6	–	–	100	100	100	100	100	100	100	100	100	100	100	100
T3	–	–	7	–	100	100	50	100	100	100	50	100	100	100	50	100
T4	8	–	–	–	200	250	200	300	200	250	200	300	200	250	200	300
T5	8	–	–	–	100	100	50	50	100	100	50	50	100	100	50	50
T6	–	10	5	5	20	20	20	20	20	20	20	20	20	20	20	20
T7	–	–	10	–	20	20	20	20	20	20	20	20	20	20	20	20
T8	–	–	–	–	100	100	100	150	100	100	100	150	100	100	100	150

	Ex 12				Ex 14				Ex 15				Ex 22–24			
Tank	C1 or C5	C2 or C6	C3 or C7	C4 or C8	C1 or C5	C2 or C6	C3 or C7	C4 or C8	C1 or C5	C2 or C6	C3 or C7	C4 or C8	C1 or C5	C2 or C6	C3 or C7	C4 or C8
T1	80	80	80	80	100	120	100	100	100	60	90	50	50	100	100	100
T2	100	100	100	100	80	80	80	80	100	50	100	100	50	100	50	100
T3	100	100	100	100	100	100	100	100	70	60	60	60	100	100	50	100
T4	200	250	200	250	60	80	80	60	100	100	100	100	100	50	50	50
T5	60	60	80	80	70	80	80	70	80	70	80	70	60	50	50	50
T6	20	20	20	20	25	25	25	25	45	30	50	35	20	20	20	20
T7	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
T8	100	100	100	150	80	60	60	50	100	100	100	50	100	100	100	150

*kton for Example 2.

Table 7. Crude Concentration Ranges in Tanks and CDUs for Examples 2–24

Example	Crude	Concentration Range (Min–Max)	Tank & CDU
2	C1–C3	1.00–1.00	T1,T2,T3 & T4–T5
	C2,C4,C6	0.00–1.00	T6
3	C1–C8	0.00–1.00	T1–T8
	C5–C8	0.00–1.00	CDUI & CDU2
	C1	0.15–0.85	CDU3
	C2–C4	0.00–1.00	CDU3
4–9 &	C1–C4	0.00–1.00	T1, T6–T8 & CDU3
21 – 24	C5–C8	0.00–1.00	T2–T5, CDU1 & CDU2
10–11,	C1–C8	0.00–1.00	T1–T8
16 & 18	C1–C4	0.10–090	CDU3
	C5–C8	0.10–0.90	CDUI & CDU2
12–13	C1–C8	0.05–0.95	T1–T8
	C1–C3	0.10–0.90	CDU3
	C4	0.00–1.00	CDU3
	C5–C8	0.10–0.90	CDUI & CDU2
14	C1–C8	0.05–0.95	T1–T8
	C1–C4	0.06–0.94	CDU3
	C5–C8	0.06–0.94	CDUI & CDU2
15	C1–C4	0.05–0.95	T1, T6–T8 & CDU3
	C5–C8	0.05–0.95	T2–T5
	C5–C8	0.08–0.92	CDUI & CDU2
17	C1–C8	0.00–1.00	T1–T8
	C1–C4	0.05–0.95	CDU3
	C5–C8	0.05–0.95	CDUI & CDU2
19	C1–C4	0.00–1.00	T1, T6–T8 & CDU3
	C5–C8	0.00–1.00	T2–T5, CDUI & CDU2
20	C1–C8	0.02–0.98	T1–T8
	C1–C4	0.05–0.95	CDU3
	C5–C8	0.05–0.95	CDUI & CDU2

representative of the actual scenarios in most refineries. The scheduling horizon is 2 days (seven 8-h periods) in Example 2, 3 days (nine 8-h periods) in Example 1, 7 days in Examples 3–6; 14 days in Examples 7–21; and 20 days in Examples 22–24. All examples have one SBM, except Example 2 that has four jetties but no SBM. Example 2 has 17 single-parcel vessels, 7 storage tanks, and two CDUs. Examples 3–6 have two VLCCs (8 parcels), 8 tanks, and 3 CDUs. Examples 7–15 and 17–19 have three VLCCs (12 parcels), 8 tanks, and 3 CDUs. Examples 16 and 20–21 have 3 jetties, 15 parcels, 8 tanks, and 3 CDUs. Finally, Examples 22–24 have four VLCCs (16 parcels), 8 tanks, and 3 CDUs, Except for Examples 14 and 15, the first vessel arrives at time zero in all examples. It should be clear that our test examples vary widely in structure, size, scale, and complexity and are representative of industrial scenarios. We used CPLEX 9.0/GAMS 21.4[4] on a Dell workstation PWS650 (Intel® Xeon™ CPU 3.06 GHz, 3.5 GB memory) running Windows XP.

Table 10 shows the solution statistics for Examples 1–21. Note that DICOPT/GAMS,[4] BARON/GAMS,[4] LA, and RA fail to solve most examples. In contrast, RRA and RRA-P are able to solve all examples. Thus, RRA and RRA-P are far more robust than the other algorithms.

Interestingly, while BARON/GAMS[4] can guarantee global solutions, it can solve Examples 1 and 2 only, and even for them, it requires huge computational times. Although DICOPT/GAMS[4] cannot guarantee global solutions, it is able to solve Examples 5 and 6, but requires huge computation times. In addition, it also gives inferior solutions compared

Table 8. Transfer Rates, Processing Limits, Operation Costs, Crude Margins, and Demands for Examples 2–24

	Flow Rate Limits (kbbl or kton*/period)		Demurrage (k$/period) or	Changeover	Inventory	Inventory	Unloading	Desirted		Margin ($/bbl or Yuan/ton*)					
Example	Parcel–Tank Min-Max	Tank-CDU Min-Max	Sea Waiting Cost* (k Yuan/period)	Loss (k$ or k Yuan*/instance)	Penalty ($/bbl/ period)	Cost (Yuan/ ton/period)	Cost (k Yuan/ period)	Safety Stock (kbbl)	Crude	Ex 2	Ex 3-4, 7-14 & 17-19	Ex 5	Ex 6	Ex 15	Ex 16, 20-21 & 22-24
2	0–5	0–5	5	1	–	0.05	7	–	C2	1	1.70	1.70	1.50	1.75	1.75
3–6	10–400	20–45	25	10	0.20	–	–	1500	C3	1	1.50	1.50	1.50	1.55	1.85
7–14, 17–19	10–400	20–40	25	10	0.02	–	–	1500	C4	1	1.60	1.60	1.50	1.80	1.25
15	10–450	20–100	20	15	0.05	–	–	1600	C5	1	1.45	1.45	1.50	1.45	1.45
16	10–250	20–50	15	5	0.20	–	–	1200	C6	1	1.60	1.60	1.50	1.70	1.65
20–21	10–250	20–50	15	5	0.20	–	–	1200	C7	–	1.55	1.55	1.50	1.60	1.55
22–24	10–300	10–80	15	5	0.20	–	–	1200	C8	–	0.05	1.50	1.50	1.65	1.60

	Processing Limits (kbbl or kton*/period)					Total Demand (kbbl)									
CDU	Ex 2	Ex 3–6	Ex 7–14	Ex 16-21	Ex 15 & 22–24	Ex 3–6	Ex 7–11	Ex 12	Ex 13	Ex 14	Ex 15	Ex 16	Ex 17 & 18	Ex 19–21	Ex 22–24
CDU 1	2–8	20–45	20–40	20–40	20–60	750	1000	900	1000	900	960	1000	1000	1000	1600
CDU 2	1–3	20–45	20–40	20–40	20–60	750	1000	900	1000	900	960	1000	1000	1000	1600
CDU 3	–	20–45	20–40	20–40	20–60	750	1000	900	900	900	960	1000	900	1000	1600

*For Example 2.

Table 9a. Specific Gravities, Sulfur Contents, Nitrogen Contents, Carbon Residues for Crudes and Acceptable Ranges for Feeds to CDUs

	Specific Gravity		Sulfur								Nitrogen		Carbon Residue	
Crude & CDU	Ex 17 [18 & 20]	Ex 19 & [21]	Ex 1	Ex 3,5,6 & [4]	Ex 7 & [9–11, 13]	Ex 8 & [15]	Ex 12 & [14]	Ex 16 & [22–24]	Ex 17 & [18 &20]	Ex 19 & 21	Ex 17 [18 & 20]	Ex 19 & 20	Ex 17 [18 & 20]	Ex 19 & 21
C1	1.2576 [1.1477]	1.2057	0.01	0.0020	0.0020	0.0020 [0.0010]	0.0020	0.0050	0.0135 [0.0320]	0.0095	32.00 [30.00]	55.00	0.0620 [0.0320]	0.0450
C2	1.2646 [1.1546]	1.2339	0.01	0.0025	0.0025	0.0025	0.0025	0.0080 [0.0020]	0.0115 [0.0280]	0.0085	21.00 [25.00]	45.00	0.0450 [0.0270]	0.0420
C3	1.2466 [1.1703]	1.2113	0.02	0.0015	0.0015	0.0015 [0.0035]	0.0015	0.0040	0.0150 [0.0276]	0.0080	45.00 [15.00]	50.00	0.0750 [0.0200]	0.0436
C4	1.2599 [1.1852]	1.2749	0.01	0.0060	0.0060	0.0060	0.0060	0.0060	0.0120 [0.025]	0.0090	25.00 [10.00]	40.00	0.0500 [0.0150]	0.0350
C5	1.0892 [1.0800]	1.0375	–	0.0120	0.0120	0.0120 [0.0180]	0.0120	0.0150 [0.0120]	0.0075 [0.0180]	0.0250	79.00 [98.00]	93.00	0.2400 [0.1200]	0.1880
C6	1.1207 [1.0959]	1.0615	–	0.0130	0.0130	0.0130 [0.0150]	0.0130 [0.0100]	0.0100	0.0050 [0.0165]	0.0235	62.00 [95.00]	88.00	0.1000 [0.0850]	0.1730
C7	1.1105 [1.105]	1.0664	–	0.0090	0.0130 [0.0090]	0.0090	0.0090	0.00200	0.0070 [0.0135]	0.0255	73.00 [85.00]	84.00	0.1800 [0.0800]	0.1540
C8	1.1148 [1.1124]	1.0968	–	0.0150	0.0150	0.0150 [0.0120]	0.0150	0.0160	0.0065 [0.0120]	0.0210	65.00 [82.00]	78.00	0.1300 [0.0700]	0.1260
CDU1 Min	1.0000	1.0000	0.00	0.0010	0.0010	0.0010	0.0013	0.0100	0.0050 [0.0130]	0.0200	60.00 [80.00]	75.00	0.1000 [0.0500]	0.1000
Max	1.1200 [1.1100]	1.0850 [1.0920]	0.01	0.0130	0.135	0.0135 [0.0165]	0.0145	0.165	0.0071 [0.0170]	0.0242	75.00 [96.00]	92.00	0.2000 [0.1000]	0.1800
CDU2 Min	1.0000	1.0000	0.01	0.0010	0.0010	0.0010	0.0010	0.0100	0.0050	0.0200 [0.0125]	60.00 [80.00]	75.00	0.1000 [0.0500]	0.1000
Max	1.200 [1.1100]	1.0900	0.02	0.0125	0.0130	0.0130 [0.0160]	0.140 [0.0160]	0.0150 [0.0172]	0.0070	0.0245	78.00 [96.50]	91.50	0.2200 [0.1150]	0.1850
CDU3 Min	1.0000 [1.1000]	1.2000	–	0.0010	0.0010	0.0010	0.0010	0.0040 [0.0010]	0.0100	0.0060 [0.0250]	20.00 [10.00]	10.00	0.0100 [0.0100]	0.0100
Max	1.2625 [1.1780]	1.27400	–	0.0035 [0.0030]	0.0040	0.0030 [0.0045]	0.0050	0.0075 [0.0045]	0.0138 [0.0290]	0.0092	40.00 [28.00]	54.00	0.0720 [0.0300]	0.0440

Data in brackets are for corresponding [Examples].

Table 9b. Pour Points, Freeze Points, Flash Points, Smoke Points, Ni Contents, and Reid Vapour Pressures for Crudes and Acceptable Ranges for Feeds to CDUs

	Pour Point		Freeze Point		Flash Point		Smoke Point		Ni		Reid Vapour Pressure	
Crude & CDU	Ex 17 [18 & 20]	Ex 19 & 21	Ex 17 [18 & 20]	Ex 19 & 21	Ex 17 [18 & 20]	Ex 19 & 21	Ex 17 [18 & 20]	Ex 19 & 21	Ex 17 [18 & 20]	Ex 19 & 21	Ex 17 [18 & 20]	Ex 19 & 21
C1	13.3290 [8.3994]	58.0549	36.3479 [133.2897]	270.2996	341.6801 [19.7932]	207.7017	588.3175 [538.0525]	548.1218	0.900 [0.095]	0.075	133.1031 [162.1270]	153.6366
C2	27.7594 [14.0162]	12.0466	10.7753 [102.0263]	211.3251	337.8010 [28.0269]	551.5897	706.1862 [561.3248]	588.8047	0.760 [0.085]	0.062	92.4005 [138.0196]	120.4380
C3	10.3347 [25.2344]	21.8409	54.8962 [77.0306]	248.0304	357.1811 [38.9152]	311.3055	557.2127 [578.8451]	567.6004	1.000 [0.082]	0.050	158.3207 [119.7402]	144.8884
C4	18.8718 [58.0549]	10.3347	26.2774 [46.0414]	168.4381	341.4362 [58.3937]	661.2327	608.9218 [632.1359]	626.5365	0.850 [0.070]	0.035	116.6061 [110.6941]	113.5842
C5	2.5140 [3.9516]	5.1896	812.1963 [701.2393]	1412.5240	2.1225 [2.1719]	16.5062	419.5659 [478.6314]	431.4538	12.500 [0.030]	19.000	28.6244 [73.9050]	24.1774
C6	9.3209 [5.4896]	4.6626	712.1419 [589.7132]	1286.6348	4.5253 [2.7114]	21.3079	536.3974 [530.977]	455.4485	5.400 [0.0235]	18.300	19.4676 [67.2142]	22.5324
C7	3.9516 [7.9708]	48.4716	757.3304 [536.1761]	1015.0334	2.6029 [3.1074]	29.5074	427.1329 [561.3248]	477.3611	10.300 [0.012]	17.500	26.2276	21.1838
C8	7.1733 [10.3347]	7.5624	744.1575 [512.9720]	768.6957	3.3367 [3.6938]	39.4486	457.4626 [578.8451]	503.5443	7.900 [0.0092]	16.700	24.1774 [48.0583]	13.8983
CDU1 Min	2.5000 [1.0000]	4.0000	700.0000 [500.0000]	700.0000	2.0000 [2.0000]	15.0000	400.0000 [450.0000]	400.0000	5.000 [0.001]	15.000	15.0000 [40.0000]	10.0000
Max	9.0000 [10.0000]	45.0000	810.000 [700.0000]	1405.0000	4.5000 [3.5000]	39.0000	530.0000 [560.0000]	475.0000	12.200 [0.027]	18.800	28.3000 [70.0000]	24.0000
CDU2 Min	2.5000 [2.0000]	4.0000	700.0000 [500.0000]	700.0000	2.0000 [2.0000]	15.0000	400.0000 [450.0000]	400.0000	5.000 [0.001]	15.000	15.0000 [40.0000]	10.0000
Max	9.2000 [10.2000]	48.0000	810.0000 [690.0000]	1410.0000	4.4800 [3.6000]	39.2000	520.0000 [570.0000]	470.0000	12.100 [0.028]	18.600	28.5000 [72.0000]	23.9000
CDU3 Min	10.0000 [8.0000]	10.0000	10.0000 [40.0000]	150.0000	300.0000 [15.0000]	200.0000	500.0000	500.0000	0.500 [0.050]	0.010	90.0000 [100.0000]	100.0000
Max	27.50000 [50.0000]	58.0000	54.0000 [130.0000]	270.0000	350.0000 [58.0000]	650.0000	700.0000 [620.0000]	600.0000	0.980 [0.092]	0.072	155.0000 [160.0000]	150.0000

Data in brackets are for corresponding [Examples].

Table 9c. Asphaltenes, Aromatics, Paraffins, Naphthenes, and Viscosities for Crudes and Acceptable Ranges for Feeds to CDUs

	Asphaltenes		Aromatics		Paraffins		Naphthenes		Viscosities	
Crude & CDU	Ex 17 [18 & 20]	Ex 19 & 21	Ex 17 [18& 20]	Ex 19 & 21	Ex 17 [18 & 20]	Ex 19 & 21	Ex 17 [18 & 20]	Ex 19 & 21	Ex 17 [18 & 20]	Ex 19 & 21
C1	0.0350 [0.1400]	0.0850	0.0892 [0.4500]	0.2972	0.7273 [0.3140]	0.3844	0.2892 [02360]	0.3414	71.7767 [79.2854]	76.8625
C2	0.0250 [0.1250]	0.0650	0.0465 [0.4200]	0.2793	0.7366 [0.3400]	0.4222	0.2835 [0.2400]	0.3203	64.5435 [78.7725]	76.2073
C3	0.0150 [0.1200]	0.0500	0.0642 [0.3500]	0.2756	0.6341 [0.3650]	0.3614	0.2281 [0.2850]	0.3022	0.3022 [78.5639]	75.7175
C4	0.0200 [0.1100]	0.0700	0.0635 [0.3030]	0.2713	0.7335 [0.4000]	0.4004	0.2901 [0.2970]	0.3443	68.1196 [78.4192]	76.5457
C5	0.1150 [0.0870]	0.2000	0.3216 [0.2980]	0.5216	0.3282 [0.3540]	0.2400	0.2767 [0.3480]	0.2384	83.1872 [76.9637]	82.6218
C6	0.0900 [0.0820]	0.1890	0.3130 [0.2960]	0.4942	0.4035 [0.3750]	0.3244	0.2200 [0.3290]	0.2302	79.3574 [76.6536]	81.5636
C7	0.1350 [0.0500]	0.1750	0.4437 [0.2760]	0.4577	0.3500 [0.4220]	0.2756	0.2085 [0.3020]	0.2407	82.5847 [75.7175]	81.1988
C8	0.1005 [0.0420]	0.1500	0.3599 [0.2500]	0.4317	0.3892 [0.4500]	0.3016	0.1990 [0.2950]	0.2439	81.5982 [75.4539]	80.3514
CDU1 min	0.0900 [0.0400]	0.1500	0.3000 [0.2500]	0.4000	0.3000 [0.3500]	0.2400	0.1800 [0.2500]	0.2000	70.0000 [75.0000]	80.0000
Max	0.1300 [0.0850]	0.1960	0.4200 [0.2960]	0.5000	0.4000 [0.4500]	0.3200	0.2700 [0.3400]	0.2420	83.0000 [76.7000]	82.5000
CDU2 Min	0.0900 [0.0400]	0.1500	0.3000 [0.2000]	0.4000	0.3000 [0.3500]	0.2400	0.1800 [0.2500]	0.2000	70.0000 [75.0000]	80.0000
Max	0.1320 [0.0840]	0.1950	0.4000 [0.2950]	0.5200	0.3950 [0.4400]	0.3200	0.2750 [0.3450]	0.2410	82.9000 [76.8000]	82.6000
CDU3 Min	0.0150 [0.1000]	0.0500	0.0400 [0.3000]	0.2500	0.6000 [0.3000]	0.3500	0.2000	0.3000	60.0000 [70.0000]	70.0000
Max	0.0320 [01360]	0.0800	0.0850 [0.4350]	0.2950	0.7350 [0.3900]	0.4200	0.2900 [0.2950]	0.3440	74.5000 [79.0000]	76.8000

Data in brackets are for corresponding [Examples].

Table 10. Solution Statistics for Various Algorithms/Codes

Example	Statistics	DICOPT	BARON	LA	RA	RRA	RRA-P	RRA-P1	IRRA-P1	RLA
1	CPU Time (s)	200	1824	–	–	2	2	2	–	1
	Profit (k$)	N/A	5069.94	N/A	N/A	5069.94	5069.94	5069.94	5069.94	5069.94
2	CPU Time (s)	–	6001	–	–	3	3	3	–	3
	Profit (k$)	N/A	1.0145E+08	N/A	N/A	1.0119E+08	1.0119E+08	1.0119E+08	1.0121E+08	1.0119E+08
3	CPU Time (s)	42689	60000	–	–	255	223	258	–	20263
	Profit (k$)	N/A	N/A	N/A	N/A	3417.29	3437.62	3456.67	3466.43	3367.10
4	CPU Time (s)	30669	60000	–	–	174	56	64	–	802
	Profit (k$)	N/A	N/A	N/A	N/A	3422.12	3391.40	3427.90	3432.44	3360.96
5	CPU Time (s)	22384	60000	–	–	152	79	149	–	912
	Profit (k$)	3000.13	N/A	N/A	N/A	3065.86	2993.80	3086.59	3128.00	3054.10
6	CPU Time (s)	17061	60000	–	–	277	28	40	–	201
	Profit (k$)	3295.00	N/A	N/A	N/A	3315.00	3325.00	3345.00	3355.00	3250.00
7	CPU Time (s)	–	60000	–	–	1694	7511	7614	–	1978
	Profit (k$)	N/A	N/A	N/A	N/A	4514.60	4594.03	4622.76	4639.40	4604.09
8	CPU Time (s)	38033	60000	–	–	6354	47	322	–	5252
	Profit (k$)	N/A	N/A	N/A	N/A	4533.35	4590.41	4605.16	4623.16	4549.67
9	CPU Time (s)	55226	60000	–	–	1796	52	110	–	1453
	Profit (k$)	N/A	N/A	N/A	N/A	4567.58	4642.37	4644.75	4670.13	4599.14
10	CPU Time (s)	33404	60000	–	–	5407	1566	1598	–	800
	Profit (k$)	N/A	N/A	N/A	N/A	4578.57	4611.29	4611.40	4630.38	4575.80
11	CPU Time (s)	40155	60000	–	–	5494	999	1484	–	3878
	Profit (k$)	N/A	N/A	N/A	N/A	4542.78	4588.19	4611.25	4615.79	4507.42
12	CPU Time (s)	49143	60000	–	–	8583	188	226	–	2802
	Profit (k$)	N/A	N/A	N/A	N/A	4150.61	4146.09	4165.28	4177.43	4130.48
13	CPU Time (s)	51538	60000	–	–	12584	5208	5416	–	19427
	Profit (k$)	N/A	N/A	N/A	N/A	4414.09	4429.52	4443.48	4451.45	4393.72
14	CPU Time (s)	70000	60000	–	6864	6864	373	693	–	58831
	Profit (k$)	N/A	N/A	N/A	4045.79	4045.79	4016.65	4078.24	4100.39	3946.39
15	CPU Time (s)	166131	60000	–	–	14589	1958	2395	–	31455
	Profit (k$)	N/A	N/A	N/A	N/A	4424.47	4468.43	4508.06	4539.64	4514.32
16	CPU Time (s)	68157	60000	–	–	14815	5860	5976	–	10425
	Profit (k$)	N/A	N/A	N/A	N/A	4718.75	4716.21	4745.88	4770.26	4684.60
17	CPU Time (s)	72672	60000	–	–	3394	411	476	–	19641
	Profit (k$)	N/A	N/A	N/A	N/A	4442.74	4492.33	4492.99	4516.79	4453.49
18	CPU Time (s)	56123	60000	–	–	9497	1576	1616	–	51267
	Profit (k$)	N/A	N/A	N/A	N/A	4450.70	4450.36	4466.38	4492.75	4416.77
19	CPU Time (s)	35403	60000	–	4639	4639	328	899	–	8446
	Profit (k$)	N/A	N/A	N/A	4600.02	4600.02	4617.17	4640.52	4653.24	4580.67
20	CPU Time (s)	40356	60000	–	–	19962	827	901	–	14092
	Profit (k$)	N/A	N/A	N/A	N/A	4695.91	4715.17	4744.35	4747.38	4719.93
21	CPU Time (s)	63002	60000	–	2988	2988	244	336	–	26496
	Profit (k$)	N/A	N/A	N/A	4730.45	4730.45	4735.18	4753.10	4762.09	4688.54

N/A = a feasible solution was either not obtained at all or not obtained within the specified time.

to RRA. As we can see from Examples 14, 19 and 21, RRA reduces to RA, when RA can find feasible solutions.

The solution times for RRA range from 28 CPU min to 5.5 h for large examples (7–21). For examples (17–21) with 15 quality specifications, they range from 49.8 min to 5.5 h. In contrast, the solution times for RRA-P range from 47.0 min to 2.1 h for Examples 7–21. Except for Example 7, RRA-P is much faster than RRA and reduces solution times by anywhere from 0% to 99.3%. For instance, RRA-P requires 1.6 h for Example 16 compared to 4.1 h for RRA. As expected, RRA-P is faster than RRA especially for large examples (22–24). We get a profit of 7370.13 k$ within 16 min for Example 22, 7349.32 k$ in 31 min for Example 23, and 7426.09 k$ in 14 min for Example 24, while RRA and other algorithms fail to solve these problems. This clearly shows the merit of using our proposed partial relaxation strategy and demonstrates that RRA-P is in fact much faster and more robust than RRA. However, note that the quality of RRA-P solution need not be better than that of RRA. For instance, the best profit for RRA-P is 4016.65 k$ compared to 4045.79 k$ for RRA in Example 14.

Solution Quality

Having achieved substantial improvements in robustness and solution efficiency, we now turn our attention to schedule quality. As mentioned at the outset, while it is not our intention to seek guaranteed globally optimal solutions in this work, we try our best to get the best solutions possible and more importantly would like to develop a way to estimate the quality of our solutions. To this end, we first discuss how we can improve the schedule given by RRA-P.

Let **S** denote the solution from RRA-P. We can obtain two series of feasible schedules (i.e. with no composition discrepancy) from **S** by solving MILPs and BLPs repeatedly. First, we take the values of the binary variables from **S**, and fix the binary variables in our exact MIBLP (involving Eqs. 1 and 2) to those values to get a BLP. Clearly, a solution **S**1 to this BLP is a schedule with no composition discrepancy. We extract the compositions of tanks from **S**1 and fix them in our MIBLP to get an MILP. We continue this alternating series of MILP and BLP, until the solutions of successive BLPs converge. Now, instead of solving a BLP first, we could also solve an MILP

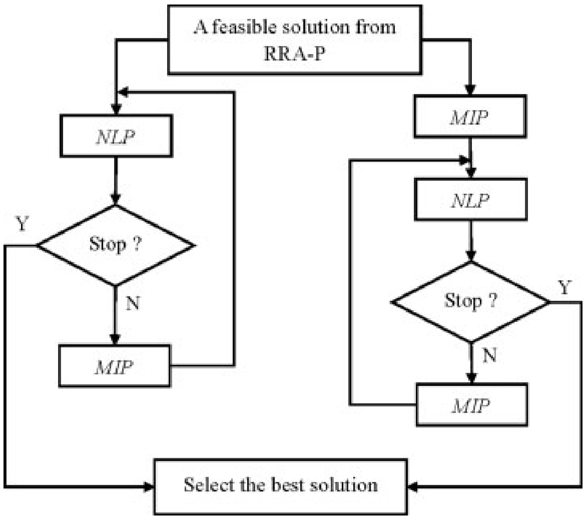

Figure 4. Flow chart for RRA-P1 (Partial Relaxation Refinement Strategy).

based on **S** and obtain another series of solutions. Figure 4 shows the complete iterative improvement procedure. The procedure normally takes about 8–10 (MILP+BLP) solutions and we take the best of all these solutions. We call this enhanced RRA-P as RRA-P1. Table 10 shows that RRA-P1 improves the solutions from RRA-P by an average of 0.60% in all examples and they are better than those from RRA by an average of 0.90%. Furthermore, the additional (MILP+BLP) solutions do not demand excessive additional computing effort. For illustration, we present the best crude schedule from RRA-P1 for Example 16 in Table 11.

While RRA-P1 improves the quality from a given solution from RRA-P, we can also use RRA-P1 repeatedly to get alternate and better solutions. Given the best solution from the previous iterations of RRA-P1, we impose an integer cut to prohibit the best combination of XT_{it} from recurring. However, this cannot guarantee a better solution. If we do get a better solution, then we impose a lower bound on the profit during the last phase of RRA-P1, where we solve MILPs and BLPs. If we do not get a better solution, then imposing a lower bound on profit will most likely result in no solution, so we merely use an integer cut to eliminate this solution and return to RRA-P1. We call this algorithm IRRA-P1. IRRA-P1 is computationally more expensive than RRA-P1 by a factor of around 10, and we must see if it improves quality substantially. We solved Examples 1–21 with IRRA-P1 to compare with RRA-P1. Table 10 shows that IRRA-P1 improves solution quality by 0–1.34% for all examples and 0.37% on an average. Although attractive, this improvement may not be worth the additional substantial computational effort of IRRA-P1.

Upper Bound on Profit

The algorithms presented so far in this paper carry no guarantee of optimality with respect to the original and full MIBLP. It is reasonable to ask how close their solutions are

Table 11. Operation Schedule from RRA-P1 for Example 16

	Crude Amount [to CDU No.] (from Vessel No.) in kbbl for Period											
Tank	1	2	3	4	5	6	7	8	9	10	11	12
1	−20.0[3]	−20.0[3]	−20.0[3]	−20.0[3]	−20.0[3]	−10.2[3]	−23.2[3]					
2	+100.0(2)	+10.0(3)		−20.0[2]	−20.0[2]	−20.0[2]	−20.0[2]	−20.0[2]	−20.0[2]	−20.0[2]	−20.0[2]	−20.0[2]
3	−20.0[2]	−20.0[2]	−20.0[2]	+125.0(6)		+120.0(8)						
4	−27.7[1]	−23.5[1]	−20.0[1]	−23.0[1]	−20.0[1]	−20.0[1]	−20.0[1]	−22.9[1]	−26.4[1]	−30.3[1]	−34.9[1]	−29.7[1]
5		+90.0(3)										
6	+10.0(1)		+80.0(4) +10.0(5)									
7				+18.3(5) +10.0(7)	+90.0(7)			−26.7[3]	−30.7[3]	−35.3[3]	−30.0[3]	−25.5[3]
8			+10.0(4) +96.7(5)									
Tank	13	14	15	16	17	18	19	20	21	22	23	24
1									+10.0(9)	+110.0(11)		+10.0(14)
2	−20.0[2]	−20.0[2]	−20.0[2]	−20.0[2]	−20.0[2]	−20.0[2]	−20.0[2]	−20.0[2]	−20.0[2]	−20.0[2]	−20.0[2]	−20.0[2]
3									−20.0[1]	−20.0[1]	−20.0[1]	−20.0[1]
4	−34.0[1]	−39.2[1]	−40.0[1]	−40.0[1]	−34.0[1]	−28.9[1]	−24.6[1]	−20.9[1]	+130.0(10)		+100.0(13)	
5												
6										+10.0(11)	+100.0(12)	
7												+80.0(14)
8	−21.7[3]	−20.0[3]	−20.0[3]	−20.0[3]	−20.0[3]	−20.0[3]	−20.0[3]	−20.0[3]	−20.0[3]	−20.0[3]	−20.0[3]	−20.0[3]
Tank	25	26	27-31	32	33	34	35	36	37	38	39	40-42
1						−20.3[3]	−22.2[3]	−25.5[3]	−28.9[3]	−32.7[3]	−37.0[3]	−40.0[3]
2	+125.0(15)											
3	−20.0[1]	−20.0[1]										
4	−20.0[2]	−20.0[2]	−20.0[2]	−20.8[2]	−23.9[2]	−27.5[2]	−31.6[2]	−36.3[2]	−40.0[2]	−40.0[2]	−40.0[2]	−40.0[2]
5			−20.0[1]	−20.0[1]	−20.0[1]	−20.0[1]	−20.0[1]	−20.0[1]	−20.0[1]	−20.0[1]	−20.0[1]	−20.0[1]
8	−20.0[3]	−20.0[3]	−20.0[3]	−20.0[3]	−20.0[3]							

"−" sign represents delivery to [CDU], "+" sign represents receipt from (Parcels).

to the global optima of the original MIBLP. Of course, a global optimization algorithm for this problem would be desirable, but considering the large sizes of practical problems and the need for quick solutions, that will require considerable effort and is best left for the future. However, even without such an algorithm, we can get reasonable conservative estimates of solution quality based on a theoretical upper bound on the profit. Recall that in the first MILP of RRA, we linearize the bilinear constraints (Eqs. 1 and 2) for most periods after the arrival period of the first vessel. For periods before the arrival of the first vessel, the bilinear constraints (Eqs. 1 and 2) become linear because the initial tank compositions are known. Since the first MILP of RRA is a linear relaxation of **F**, its optimal solution is a valid upper bound for **F**. However, the first MILP of RRA is the largest and the most difficult to solve and often cannot be solved to zero relative gap for medium and large-size problems. Therefore, we use the best possible integer solution from the first MILP of RRA as the upper bound, because the best possible integer solution is a valid upper bound on the optimal solution. Because the linearization is not very tight, we need other constraints to improve this upper bound. To this end, we now present several novel tightening constraints. The only reason why these novel constraints are not in RRA and RRA-P is that their inclusion makes MILPs intractable for most problems.

Apart from the initial periods in which all tank compositions are known ($= f_{ic0}$) in the first iteration, RRA approximates Eqs. 1 and 2 in all the remaining periods. Our linear estimators of Eqs. 1 and 2 are not as tight as the best known linearizations of McCormick[12] for bilinear constraints. Thus, our first tightening constraints are the following derived from Relaxation Linearization Technology.[12]

$$FCTU_{iuct} \ge f_{ict} FTU^L_{iu} + xt^L_{ic} FTU_{iut} - xt^L_{ic} FTU^L_{iu} - FTU^L_{iu}(1 - Y_{iut}) \quad (i,u) \in \boldsymbol{IU}, (i,c) \in \boldsymbol{IC} \tag{11a}$$

$$FCTU_{iuct} \ge f_{ict} FTU^U_{iu} + xt^U_{ic} FTU_{iut} - xt^U_{ic} FTU^U_{iu} \quad (i,u) \in \boldsymbol{IU}, (i,c) \in \boldsymbol{IC} \tag{11b}$$

$$FCTU_{iuct} \le f_{ict} FTU^U_{iu} + xt^L_{ic} FTU_{iut} - xt^L_{ic} FTU^U_{iu} \quad (i,u) \in \boldsymbol{IU}, (i,c) \in \boldsymbol{IC} \tag{11c}$$

$$FCTU_{iuct} \le f_{ict} FTU^L_{iu} + xt^U_{ic} FTU_{iut} - xt^U_{ic} FTU^L_{iu} + FTU^L_{iu}(1 - Y_{iut}) \quad (i,u) \in \boldsymbol{IU}, (i,c) \in \boldsymbol{IC} \tag{11d}$$

$$VCT_{ict} \ge f_{ict} V^L_i + xt^L_{ic} V_{it} - xt^L_{ic} V^L_i \quad (i,c) \in \boldsymbol{IC} \tag{12a}$$

$$VCT_{ict} \ge f_{ict} V^U_i + xt^U_{ic} V_{it} - xt^U_{ic} V^U_i \quad (i,c) \in \boldsymbol{IC} \tag{12b}$$

$$VCT_{ict} \le f_{ict} V^U_i + xt^L_{ic} V_{it} - xt^L_{ic} V^U_i \quad (i,c) \in \boldsymbol{IC} \tag{12c}$$

$$VCT_{ict} \le f_{ict} V^L_i + xt^U_{ic} V_{it} - xt^U_{ic} V^L_i \quad (i,c) \in \boldsymbol{IC} \tag{12d}$$

where xt^L_{ic} and xt^U_{ic} are the limits on the composition of crude c in tank i, FTU^L_{iu} and FTU^U_{iu} are the limits on the amount of crude charge per period from tank i to CDU u, and V^L_i and V^U_i are the limits on crude inventory in tank i. Note that these are not written for initial periods in which the tank compositions are known to be f_{ic0}, but only for those periods in which they are unknown.

In most scenarios, not all tanks may receive crude, when the first ship arrives. Thus, their compositions will remain as f_{ic0} even after that. However, the first iteration of RRA uses exact linearizations of Eqs. 1 and 2 only for those periods in which the tank compositions are f_{ic0} without a doubt. We can relax this requirement further by forcing a tank composition to be f_{ic0}, until it receives a crude. We can detect the transfer of crude to a tank by looking at the value of $\sum_{\tau \le t} XT_{i\tau}$. As long as this remains zero, the composition of tank i must remain f_{ic0}. To enforce this idea, we use the following two constraints:

$$FTU^U_{iu} xt^U_{ic} \sum_{\tau \le t} XT_{i\tau} + FCTU_{iuct} \ge FTU_{iut} f_{ic0} \quad (i,u) \in \boldsymbol{IU}, (i,c) \in \boldsymbol{IC} \tag{13a}$$

$$FTU^U_{iu} xt^U_{ic} \sum_{\tau \le t} XT_{i\tau} + FTU_{iut} f_{ic0} \ge FCTU_{iuct} \quad (i,u) \in \boldsymbol{IU}, (i,c) \in \boldsymbol{IC} \tag{13b}$$

Similar constraints can also be written for f_{ict} in those periods, where the tank composition is unknown:

$$f_{ict} \ge f_{ic0} - xt^U_{ic} \sum_{\tau \le t} XT_{i\tau} \quad (i,c) \in \boldsymbol{IC} \tag{14a}$$

$$f_{ict} \le f_{ic0} + xt^U_{ic} \sum_{\tau \le t} XT_{i\tau} \quad (i,c) \in \boldsymbol{IC} \tag{14b}$$

If a tank i receives no crude during a period t, then its composition must remain constant. In other words,

$$f_{ict} \ge f_{ic(t-1)} - xt^U_{ic} XT_{it} \quad (i,c) \in \boldsymbol{IC} \tag{15a}$$

$$f_{ict} \le f_{ic(t-1)} + xt^U_{ic} XT_{it} \quad (i,c) \in \boldsymbol{IC} \tag{15b}$$

Furthermore, the individual crude fractions in each tank must sum to 1.

$$\sum_c f_{ict} = 1 \quad (i,c) \in \boldsymbol{IC} \tag{16}$$

If only one tank i feeds CDU u during period t, then the crude quality in tank i at the end of t must satisfy the crude quality constraints for CDU u.

$$V^U_i \theta^L_{ku} \left(1 - Y_{iut} + \sum_{ii \neq i} Y_{(ii)ut} \right) + \sum_c VCT_{ict} \theta_{kc} \ge V_{it} \theta^L_{ku} \quad (ii,u) \in \boldsymbol{IIU}, (i,u) \in \boldsymbol{IU}, (i,c) \in \boldsymbol{IC} \tag{17a}$$

$$\sum_c VCT_{ict} \theta_{kc} \le V_{it} \theta^U_{ku} + \left(\sum_c V^U_i xt^U_{ic} \theta_{kc} \right) \left(1 - Y_{iut} + \sum_{ii \neq i} Y_{(ii)ut} \right) \quad (ii,u) \in \boldsymbol{IIU}, (i,u) \in \boldsymbol{IU}, (i,c) \in \boldsymbol{IC} \tag{17b}$$

$$\theta^L_{ku}\left(\sum_c V^U_i xt^U_{ic}\rho_c\right)\left(1 - Y_{iut} + \sum_{ii\neq i} Y_{(ii)ut}\right) + \sum_c VCT_{ict}\rho_c\theta_{kc} \geq \theta^L_{ku}\left(\sum_c VCT_{ict}\rho_c\right)$$
$$(ii,u)\in \boldsymbol{IIU}, (i,u)\in \boldsymbol{IU}, (i,c)\in \boldsymbol{IC} \quad (17c)$$

$$\sum_c VCT_{ict}\rho_c\theta_{kc} \leq \theta^U_{ku}\left(\sum_c VCT_{ict}\rho_c\right) + \left(\sum_c V^U_i xt^U_{ic}\rho_c\theta_{kc}\right)\left(1 - Y_{iut} + \sum_{ii\neq i} Y_{(ii)ut}\right)$$
$$(ii,u)\in \boldsymbol{IIU}, (i,u)\in IU, (i,c)\in \boldsymbol{IC} \quad (17d)$$

where $\boldsymbol{IIU} = \{(ii,u)\,|\,\text{tank}\,ii\,\text{can feed CDU}\,u\}$.

Similarly, we can also obtain the following by using f_{ict}.

$$\sum_c f_{ict}\theta_{kc} \geq \theta^L_{ku} - \theta^L_{ku}\left(1 - Y_{iut} + \sum_{ii\neq i} Y_{(ii)ut}\right)$$
$$(ii,u)\in \boldsymbol{IIU}, (i,u)\in \boldsymbol{IU}, (i,c)\in \boldsymbol{IC} \quad (18a)$$

$$\sum_c f_{ict}\theta_{kc} \leq \theta^U_{ku} + \left(\sum_c xt^U_{ic}\theta_{kc}\right)\left(1 - Y_{iut} + \sum_{ii\neq i} Y_{(ii)ut}\right)$$
$$(ii,u)\in \boldsymbol{IIU}, (i,u)\in \boldsymbol{IU}, (i,c)\in \boldsymbol{IC} \quad (18b)$$

$$\sum_c f_{ict}\rho_c\theta_{kc} \geq \theta^L_{ku}\left(\sum_c f_{ict}\rho_c\right) - \theta^L_{ku}\left(\sum_c xt^U_{ic}\rho_c\right)\left(1 - Y_{iut} + \sum_{ii\neq i} Y_{(ii)ut}\right)$$
$$(ii,u)\in \boldsymbol{IIU}, (i,u)\in \boldsymbol{IU}, (i,c)\in \boldsymbol{IC} \quad (18c)$$

$$\sum_c f_{ict}\rho_c\theta_{kc} \leq \theta^U_{ku}\left(\sum_c f_{ict}\rho_c\right) + \left(\sum_c xt^U_{ic}\rho_c\theta_{kc}\right)\left(1 - Y_{iut} + \sum_{ii\neq i} Y_{(ii)ut}\right)$$
$$(ii,u)\in \boldsymbol{IIU}, (i,u)\in \boldsymbol{IU}, (i,c)\in \boldsymbol{IC} \quad (18d)$$

If only one tank i feeds a CDU u during t, then its concentration must satisfy the crude quality constraints of CDU u,

$$f_{ict} \geq xc^L_{cu} - xc^L_{cu}\left(1 - Y_{iut} + \sum_{ii} Y_{(ii)ut}\right)$$
$$(ii,u)\in \boldsymbol{IIU}, (i,u)\in \boldsymbol{IU}, (i,c)\in \boldsymbol{IC} \quad (19a)$$

$$f_{ict} \leq xc^U_{cu} + xt^U_{ic}\left(1 - Y_{iut} + \sum_{ii} Y_{(ii)ut}\right)$$
$$(ii,u)\in \boldsymbol{IIU}, (i,u)\in \boldsymbol{IU}, (i,c)\in \boldsymbol{IC} \quad (19b)$$

$$VCT_{ict} \geq V_{it}xc^L_{cu} - V^U_i xc^L_{cu}\left(1 - Y_{iut} + \sum_{ii} Y_{(ii)ut}\right)$$
$$(ii,u)\in \boldsymbol{IIU}, (i,u)\in \boldsymbol{IU}, (i,c)\in \boldsymbol{IC} \quad (19c)$$

$$VCT_{ict} \leq V_{it}xc^U_{cu} + V^U_i xt^U_{ic}\left(1 - Y_{iut} + \sum_{ii} Y_{(ii)ut}\right)$$
$$(ii,u)\in \boldsymbol{IIU}, (i,u)\in \boldsymbol{IU}, (i,c)\in \boldsymbol{IC} \quad (19d)$$

where xc^L_{cu} and xc^U_{cu} are limits on the composition of crude c in feed to CDU u.

While Eqs. 12a–19d are nonredundant, many of them are big-M constraints. The sheer number of these constraints makes the problem too large to be practical for routine use. This is why we only use these constraints for obtaining tight upper bounds only and not inside RRA-P1.

Deviations from Upper Bounds

We solve Examples 1–21 by adding Eqs. 12a–19d to obtain upper bounds (Table 12). Their addition makes the medium- to large-size problems hard to solve to zero relative gaps. Therefore, we have to solve them within some small relative gaps (e.g. 3% for medium-size and 5% for large-size problems). In such cases, we use the best possible integer solution as the upper bound. Clearly, these are conservative estimates of upper bounds. However, from Table 12, we see that Eqs. 12a–19d do produce upper bounds lower than those obtained from the first iterations of RRA.

We assess solution quality as a percent deviation of the best solution from its upper bound. The deviations are below 3% and 6% for medium-size and large-size problems respectively except Example 5 whose deviation is about 8.8%. While the deviation for Example 1 is 1.59%, the solution from RRA-P1 is indeed the global optimum as confirmed by BARON/GAMS.[4] The solution for Example 2 is very near the global optimum, because the deviation is about 0.40%.

Table 12. The Upper Bound for Examples 1–21

Example	Profit (k$)	Upper Bound 1	Upper Bound 2	Solution Deviation
1	5069.94	5152.09	5185.63	0.0159
2	1.0121E+08	1.0161E+08	1.0161E+08	0.0040
3	3466.43	3554.60	3580.34	0.0248
4	3432.44	3537.27	3570.67	0.0296
5	3128.00	3429.59	3508.74	0.0879
6	3355.00	3373.00	3375.00	0.0053
7	4639.40	4759.79	4788.96	0.0253
8	4623.16	4767.26	4803.97	0.0302
9	4670.13	4797.80	4820.27	0.0266
10	4630.38	4780.07	4782.97	0.0313
11	4615.79	4776.10	4786.96	0.0336
12	4177.43	4298.61	4306.18	0.0282
13	4451.45	4614.17	4627.54	0.0353
14	4100.39	4290.00	4299.29	0.0442
15	4539.64	4802.58	4810.46	0.0547
16	4770.26	4900.82	4911.14	0.0266
17	4516.79	4639.73	4654.29	0.0265
18	4492.75	4620.00	4626.04	0.0275
19	4653.24	4804.46	4811.10	0.0315
20	4747.38	4912.52	4920.63	0.0336
21	4762.09	4949.71	4963.65	0.0379

Profit = the solution from IRRA-P1.
Upper bound 1 = the best possible integer solution obtained from the first iteration of RRA with eqs. 12–19.
Upper bound 2 = the best possible integer solution obtained from the first iteration of RRA.
Solution deviation = the relative gap between profit and upper bound 1.

It is possible to explain the large deviation (8.8%) for Example 5. As we can see from Tables 4–9, the margin for crude 8 is very different from those of crudes 1–7. Since the MILP for the upper bound problem does involve relaxation and cannot guarantee a solution free from composition discrepancy, the solver has the freedom to feed crudes 1–7 preferentially over crude 8 during the upper bound problem. This naturally makes the upper bound and causes a large deviation. Indeed, the results of Example 6 lend further support to our explanation. The crude margins (Tables 4–9) are very similar in Example 6, and the deviation is indeed small (0.53%).

NLP-Based Strategy

It is clear that the linear approximation of bilinear constraints is the root cause of composition discrepancy. The algorithms discussed so far used the rolling-horizon type procedure of Reddy et al.[3,9] to avoid discrepancy. An alternate strategy that does not "decompose" the problem along the time dimension is to solve an alternating series of MILP and NLP as proposed by Li et al.[7] This strategy is very similar to the final MILP-NLP refinement used in RRA-P1. Li et al.[7] solve an MILP approximation (MIP I) in the first iteration. They examine this solution to see if the concentration of crudes in each storage tank is the same as that of the feed from that tank. They call this as condition I in Li et al.[7] If condition I is satisfied, then an optimal solution is found and their procedure terminates. Otherwise, they fix the integer map from MIP I into the original MINLP model to get an NLP, whose solution ensures composition consistency. If the gap between the objectives of NLP and MIP I satisfies a tolerance (condition II), then the procedure terminates. Otherwise, the tank compositions obtained from the NLP are fixed in the original MINLP model to get a new MILP (MIP II in Li et al.[7]). If the objective value from MIP II is better than that from NLP and the integer map obtained from MIP II is different from the fixed integer map in NLP (condition III), then the integer map from MIP II is fixed into the original MINLP model to get another NLP, and the iterations are repeated. Otherwise, the procedure again terminates. However, as pointed out by Reddy et al.,[3,9] this algorithm fails, whenever the integer map from MIP I yields an infeasible NLP.

Some key ideas described and used earlier in RRA-P, namely integer cuts, tightening constraints, and slack variables, can also be employed successfully in the NLP-based algorithm (LA) of Li et al.[7] To see if such a strategy can be more efficient or superior, we modified LA to obtain a revised algorithm (RLA), and tested it along with other algorithms in this paper. RLA adds Eqs. 11–13 and 17 to tighten the MILP approximation in RRA and uses the same integer cuts as in RRA. We call this modified MILP approximation in RLA as MIP-1.

It is clear that a composition discrepancy may still exist in a solution of MIP-1, which will lead to an infeasible NLP. We used two slack variables in Eq. 7 to get an NLP solution to avoid such infeasibility. Here, in addition, we use two more positive slack variables (s^{+}_{ukt} and s^{-}_{ukt}) corresponding to Eqs. 4 and 5 in the NLP. These relate to the other source of infeasibility, namely the feed quality requirement. Thus, we replace Eqs. 4 and 5 in the NLP by Eqs. 20 and 21 and use the following objective in the NLP:

$$\begin{aligned}\text{Profit} = {} & \sum_i \sum_u \sum_c \sum_t FCTU_{iuct} CP_{cu} \\ & - \sum_v DC_v - COC \sum_u \sum_t CO_{ut} - \sum_t SC_t \\ & - M \sum_i \sum_u \sum_c \sum_t (u^{+}_{iuct} + u^{-}_{iuct}) \\ & - M \sum_u \sum_k \sum_t (s^{+}_{ukt} + s^{-}_{ukt})\end{aligned}$$

$$\theta^{L}_{ku} FU_{ut} - s^{-}_{ukt} \leq \sum_i \sum_c FCTU_{iuct} \theta_{kc} \quad (i,u) \in \boldsymbol{IU}, (i,c) \in \boldsymbol{IC} \tag{20a}$$

$$\sum_i \sum_c FCTU_{iuct} \theta_{kc} \leq s^{+}_{ukt} + \theta^{U}_{ku} FU_{ut} \quad (i,u) \in \boldsymbol{IU}, (i,c) \in \boldsymbol{IC} \tag{20b}$$

$$\theta^{L}_{ku} \left(\sum_i \sum_c FCTU_{iuct} \rho_c \right) - s^{-}_{ukt} \leq \sum_i \sum_c FCTU_{iuct} \rho_c \theta_{kc} \quad (i,u) \in \boldsymbol{IU}, (i,c) \in \boldsymbol{IC} \tag{21a}$$

$$\sum_i \sum_c FCTU_{iuct} \rho_c \theta_{kc} \leq s^{+}_{ukt} + \theta^{U}_{ku} \left(\sum_i \sum_c FCTU_{iuct} \rho_c \right) \quad (i,u) \in \boldsymbol{IU}, (i,c) \in \boldsymbol{IC} \tag{21b}$$

where M is a large number to force the slack variables to zero, whenever possible. If all the slack variables (u^{+}_{iuct}, u^{-}_{iuct}, s^{+}_{ukt}, and s^{-}_{ukt}) are zero, then the integer map obtained from MIP-1 is feasible. If not, we use the values of the slack variables to identify the sources of infeasibility, add the following slack cuts, and fix the tank concentrations obtained from the NLP (see Li et al.[7] for further details) in MIP-1 to obtain a revised MILP approximation (MIP-2R). Note that MIP-2R does not include the tightening constraints in MIP-1.

$$\sum_{i'} FCTU_{i'uct} \geq [u^{-}_{iuct}] \quad (i,u,c,t) \in \boldsymbol{IF11}_{iuct}, i' \in \boldsymbol{IF12}_{i(i')uct} \tag{22}$$

$$\sum_{i'} FCTU_{i'uct} \geq [u^{+}_{iuct}] \quad (i,u,c,t) \in \boldsymbol{IF21}_{iuct}, i' \in \boldsymbol{IF22}_{i(i')uct} \tag{23}$$

$$\sum_{ii} FTU_{(ii)ut} \geq \varepsilon \quad (ii,u) \in \boldsymbol{IIU}, ii \in \boldsymbol{IE1}_{(ii)ukt} \tag{24}$$

where ε is a small number, $[u^{-}_{iuct}]$ and $[u^{+}_{iuct}]$ are the values of slack variables u^{-}_{iuct} and u^{+}_{iuct} respectively, and sets $\boldsymbol{IF11}_{iuct}$, $\boldsymbol{IF12}_{i(i')uct}$ are defined dynamically as shown in Figure 5. If $f_{ict} > xc^{U}_{cu}$, then the NLP solver makes $[u^{-}_{iuct}]$ positive. This reduces $FCTU_{iuct}$ and keeps the composition of crude c in the feed to CDU u below xc^{U}_{cu}. In this case, tank i cannot feed CDU u alone, which is included in set $\boldsymbol{IF11}_{iuct}$. The possible modifications of the integer map related to tank i

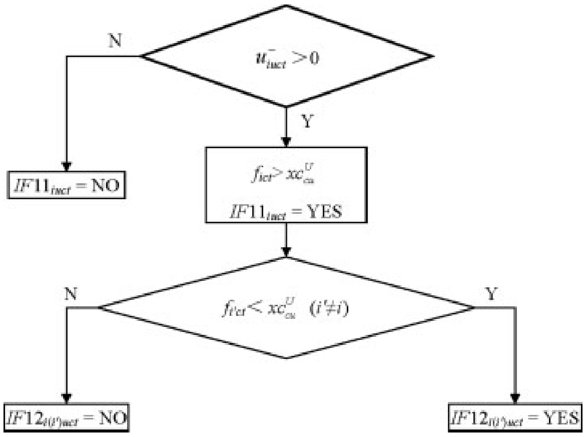

Figure 5. Definition of sets for slack cuts.

are: (a) set $Y_{iut} = 0$, so tank i cannot charge CDU u, (b) change the value of $Y_{i'ut}$ (i' denotes a tank that can charge CDU u and $f_{i'ct} < xc^{U}_{cu}$), so that other tanks can replace tank i or charge CDU u together with tank i during period t. In this way, the values of Y_{iut} and $Y_{i'ut}$ will change and FPT_{pit} and XT_{it} will change accordingly. In practice, setting $Y_{iut} = 0$ may cause infeasibility, because this excludes the possibility that tank i may charge CDU u together with other tanks. Thus, we should look for tanks i' with $f_{i'ct} < xc^{U}_{cu}$. These tanks are in $\boldsymbol{IF12}_{i(i')uct}$ $\boldsymbol{IF21}_{iuct}$ and $\boldsymbol{IF22}_{i(i')uct}$ are defined similarly, except that $[u^{+}_{iuct}]$ is positive ($f_{ict} < xc^{L}_{cu}$) and $f_{i'ct} > xc^{L}_{cu}$.

Equation 22 addresses the situation, when the concentration of crude c in tank i exceeds the upper acceptable limit for CDU u. When u^{-}_{iuct} in Eq. 7 is nonzero, the concentration of crude c in tank i exceeds the upper limit of concentration to CDU u. Clearly, this tank i cannot feed CDU u alone. In other words, it must charge CDU u together with some other tank/s in $\boldsymbol{IF12}_{i(i')uct}$. Thus, the total amount of crude from tanks in $\boldsymbol{IF12}_{i(i')uct}$ must be nonzero. Similarly, Eq. 23 addresses the situation, when the concentration of crude c in tank i is below the lower limit for CDU u.

Equation 24 handles the feed quality requirements for CDU u during t. When $s^{+}_{ukt} > 0$, property k in the feed to CDU u exceeds the upper limit for CDU u. To bring this within the limits for CDU u, we find tanks ii with crude qualities for property k below the upper limit for CDU u, and include them in $\boldsymbol{IE1}_{(ii)ukt}$ Then, to ensure that one or more of these tanks feed CDU u, we force the sum of the flows from these tanks during period t to be nonzero.

Figure 6 illustrates the procedure for RLA. We first solve MIP-1. Then, we check for composition discrepancy in the solution using condition I from Li et al.[7] If no composition discrepancy exists, then the algorithm stops. Otherwise, we fix the integer map and solve the NLP problem incorporating Eqs. 7, 20, and 21. If some slack variables are nonzero, then we identify sets $\boldsymbol{IF11}_{iuct}$, $\boldsymbol{IF12}_{i(i')uct}$, $\boldsymbol{IF21}_{iuct}$, $\boldsymbol{IF22}_{i(i')uct}$ and $\boldsymbol{IE1}_{(ii)ukt}$ as explained earlier and resolve MIP-2R involving Eqs. 22–24 and NLP repeatedly, until we get a feasible solution. If the slack variables are zero, then we have a feasible solution. Once we have a feasible solution, then we fix the tank concentrations in MIP-1 and remove redundant constraints (no discrepancy exists so all tightening constraints will be redundant) to get MIP-2 and do the iterative refinement as in RRA-P1. Note that the iteration conditions (I, II, and III) in Figure 6 refer to Li et al.[7]

Evaluation of RLA

Let us first illustrate RLA using Example 1. The MILP approximation with Eqs. 11a–11d, 12a–12d, 13a–d, and 17a–17d gives us our first integer map. This integer map results in an infeasible NLP. Therefore, we add the slack variables, and get $u^{-}_{4115} = 0.268$, $u^{-}_{4116} = 0.294$, $u^{-}_{4117} = 0.396$, $u^{-}_{4118} = 0.564$, $u^{+}_{4115} = 0.268$, $u^{+}_{4116} = 0.294$, $u^{+}_{4117} = 0.396$, $u^{+}_{4118} = 0.564$ by solving the NLP with slack variables. Using Figure 5, we get $\boldsymbol{IF11}_{\boldsymbol{iuct}}$={(4, 1, 1, 5), (4, 1, 1, 6), (4, 1, 1, 7), (4, 1, 1, 8)} and $\boldsymbol{IF12}_{i(i')uct}$={(4, 1, 1, 1, 5), (4, 1, 1, 1, 6), (4, 1, 1, 1, 7), (4, 1, 1, 1, 8)}. Then, we solve MIP-2R by using Eqs. 22–24 and fixing the tank concentrations obtained from the NLP. The solution of MIP-2R gives us an integer map that yields a feasible NLP with a profit of 5066.7 k$. Using this feasible solution, we get MIP-2, then do the iterative refinement as in RRA-P1 to get the final profit of 5069.94 k$.

To evaluate RLA more rigorously, we solved Examples 1–21 using RLA. Table 10 also shows the performance of RLA along with other algorithms. RLA does improve on LA (Li et al.[7]) and gives feasible solutions for all examples, but

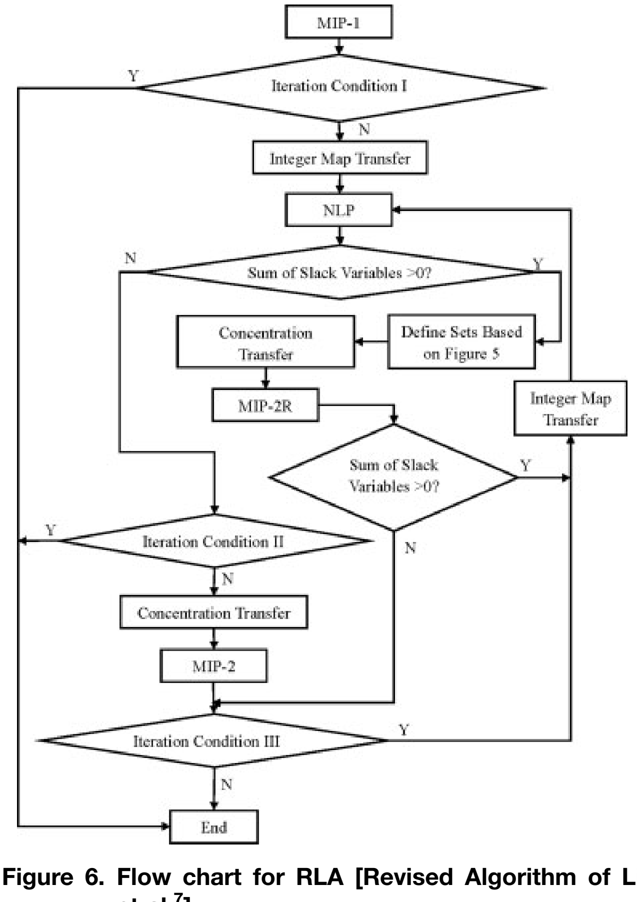

Figure 6. Flow chart for RLA [Revised Algorithm of Li et al.[7]].

requires much longer solution times than RRA-P1. This is because RLA must solve much larger MILPs due to Eqs. 11a–11d, 12a–12d, 13a–d, and 17a–17d. Surprisingly, RLA fails to match the solution quality of RRA-P1.

Conclusion

We revised the formulation and algorithm of Reddy et al.[3,9] for the difficult and nonconvex MINLP problem of scheduling crude oil operations involving blending in a refinery. While the revised algorithm is intended for a marine-access refinery, the algorithmic strategy is applicable to other types of refineries such as in-land refineries. Although the algorithm does not guarantee globally optimal schedules, it successfully solves problems with several ship/VLCC arrivals and up to 20 days of scheduling horizon, and gives schedules with profits within 6% of a conservative upper bound. It is superior to existing literature algorithms (Li et al.,[7] Reddy et al.,[3,9] and others) and general-purpose software (BARON/GAMS,[4] DICOPT/GAMS[4]) in several respects.

1. It enhances the practical utility of crude scheduling algorithms by identifying and modeling fifteen crude quality specifications currently used by the refinery industry. It reports relevant indexes and linear (weight-based or volume-based) blending correlations to address nonlinear crude properties.

2. It is far more robust in getting a good feasible schedule. While it successfully solved all the 24 industry-scale examples that we tested in this paper, other algorithms and software failed to solve most of them.

3. It is significantly faster. A clever partial relaxation strategy enables it to solve large problems, which others fail to solve in reasonable time, without compromising solution quality, causing any composition discrepancy, or requiring NLP solutions. It is also much faster (by a factor of nearly five on an average) than an NLP-based algorithm that we devised by improving the algorithm of Li et al.[7]

4. It improves schedule quality by employing an iterative refinement strategy. While for small problems, its solutions are very near-optimal, for medium-size examples, they are within 3% of a conservative upper bound on the profit

The proposed algorithm is timely and useful in this era of increasing crude prices and decreasing crude qualities. Further work is now appearing and is desirable on global optimization algorithms for solving this difficult scheduling problem,[13] and also on addressing disruptions and uncertainty in crude scheduling.[14–18]

Acknowledgments

The authors acknowledge the financial support for this work from The Agency for Science, Technology, and Research (A*Star) under grant 052 116 0074.

Notation

Sets

$\boldsymbol{IC}$ = set of pairs (tank i, crude c) that i can hold c
$\boldsymbol{IU}$ = set of pairs (tank i, CDU u) that i can feed u
$\boldsymbol{IIU}$ = set of pairs (tank ii, CDU u) that ii can feed u
$\boldsymbol{IF11}_{iuct}$ = dynamic set defined from the value of slack variable u_{iuct}^{-}
$\boldsymbol{IF12}_{i(i')uct}$ = dynamic set defined from the value of slack variable u_{iuct}^{-}
$\boldsymbol{IF21}_{iuct}$ = dynamic set defined from the value of slack variable u_{iuct}^{+}
$\boldsymbol{IF22}_{i(i')uc}$ = dynamic set defined from the value of slack variable u_{iuct}^{+}
$\boldsymbol{IE1}_{(ii)ukt}$ = dynamic set defined from the values of s_{ukt}^{-} and s_{ukt}^{+}
β = vessel-based blocks
α = composition-based blocks
η_{β} = first composition-based block in block β

Parameters

$\gamma_{u}^{L/U}$ = limits on period-period changes in crude feed flows to CDU u
$\theta_{ku}^{L/U}$ = limits on blending index for crude property k in the feed to CDU u
ρ_{c} = the density of crude c
θ_{kc} = specification index for property k of crude c
NZ = the number of terms in the first summation
$xt_{ic}^{L/U}$ = limits on the composition of crude c in tank i
$FTU_{iu}^{L/U}$ = limits on the amount of crude charge per period from tank i to CDU u
$V_{i}^{L/U}$ = limits on crude inventory of tank i
H = schedule horizon

Binary variables

XP_{pt} = 1 if parcel p is connected for transfer during period t
XT_{it} = 1 if tank i is connected to receive crude during period t
Y_{iut} = 1 if tank i feeds CDU u during period t

Continuous variables

FTU_{iut} = total amount of crude from tank i to CDU u during period t
$FCTU_{iuct}$ = the amount of crude c from tank i to CDU u during period t
V_{it} = total amount of crude in tank i at the end of period t
VCT_{ict} = the amount of crude c in tank i at the end of period t
f_{ict} = the fraction of crude c in tank i at the end of period t
FU_{ut} = total amount of crude fed to CDU u during period t
u_{iuct}^{+} = positive slack variables
u_{iuct}^{-} = positive slack variables
s_{ukt}^{-} = slack variables for property specification index constraints
s_{ukt}^{+} = slack variables for property specification index constraints

Literature Cited

1. Kelly JD, Mann JL. Crude oil blend scheduling optimization: an application with multi-million dollar benefits—part 1. *Hydrocarb Process.* 2003;82:47–51.
2. Kelly JD, Mann JL. Crude oil blend scheduling optimization: an application with multi-million dollar benefits—part 2. *Hydrocarb Process.* 2003;82:72–79.
3. Reddy PCP, Karimi IA, Srinivasan R. A novel solution approach for optimizing scheduling crude oil operations. *AIChE J.* 2004;50:1177–1197.
4. Brooke A, Kendrick D, Meeraus A, Raman R. GAMS: a user's guide. GAMS development corporation, 1998.
5. Lee H, Pinto JM, Grossmann IE, Park S. Mixed-integer linear programming model for refinery short-term scheduling of crude oil unloading with inventory management. *Ind Eng Chem Res.* 1996;35:1630–1641.
6. Jia Z, Ierapetritou MG, Kelly JD. Refinery short-term scheduling using continuous time formulation: crude oil operations. *Ind Eng Chem Res.* 2003;42:3085–3097.
7. Li WK, Hui CW, Hua B, Tong Z. Scheduling crude oil unloading, storage, and processing. *Ind Eng Chem Res* 2002;41:6723–6734.
8. Moro LFL, Pinto JM. Mixed-integer programming approach for short-term crude oil scheduling. *Ind Eng Chem Res.* 2004;43:85–94.
9. Reddy PCP, Karimi IA, Srinivasan R. A new continuous time formulation for scheduling crude oil operations. *Chem Eng Sci.* 2004;59:1325–1341.

10. Shah N. Mathematical programming techniques for crude oil scheduling. *Comput Chem Eng*. 1996;20:S1227–S1232.
11. Balas E, Jeroslow R. Canonical cuts on the unit hypercube. *SIAM J Appl Math*. 1972;23:61–69.
12. McCormick GP. Computability of global solutions to factorable nonconvex programs. Part I: Convex underestimating problems. *Math Program*. 1976;10:146–175.
13. Karuppiah R, Furman K, Grossmann IE. Global optimization for scheduling refinery crude oil operations. Presented in INFORMS Annual Meeting, Pittsburgh, PA, USA, Nov. 5–8, 2006.
14. Adhitya A, Srinivasan R, Karimi IA. Heuristic rescheduling of crude oil operations to manage abnormal supply chain events. *AIChE J*. 2007;53:397–422.
15. Adhitya A, Srinivasan R, Karimi IA. A model-based rescheduling framework for managing abnormal supply chain events. *Comput Chem Eng*. 2007;31:496–518.
16. Li J, Li WK, Karimi IA, Srinivasan R. Robust and efficient algorithm for optimizing crude oil operations. Presented in AIChE Annual Meeting, Cincinnati, OH, USA, Oct. 30–Nov. 4, 2005.
17. Li J, Karimi IA, Srinivasan R. Robust and partial relaxation algorithms for crude oil scheduling. Presented in the 3rd Sino-Japanese Optimization Meeting in the e-Business Era, Singapore, Oct. 31–Nov. 2, 2005.
18. Li J, Karimi IA, Srinivasan R. Robust scheduling of crude oil operations under demand and ship arrival uncertainty. Presented in AIChE Annual Meeting, San Francisco, CA, USA, Nov. 12–17, 2006.

Manuscript received Mar. 19, 2007, and revision received Jun. 11, 2007.