AIChE

Recipe Determination and Scheduling of Gasoline Blending Operations

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

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

DOI 10.1002/aic.11970
Published online October 26, 2009 in Wiley InterScience (www.interscience.wiley.com).

Gasoline is a major contributor to the profit of a refinery. Scheduling gasoline-blending operations is a critical and complex routine task involving tank allocation, component mixing, blending, product storage, and order delivery. Optimized schedules can maximize profit by avoiding ship demurrage, improving order delivery, minimizing quality give-aways, avoiding costly transitions and slop generation, and reducing inventory costs. However, the blending recipe and scheduling decisions make this problem a nonconvex mixed-integer nonlinear program (MINLP). In this article, we develop a slot-based MILP formulation for an integrated treatment of recipe, specifications, blending, and storage and incorporate many real-life features such as multipurpose product tanks, parallel nonidentical blenders, minimum run lengths, changeovers, piecewise constant profiles for blend component qualities and feed rates, etc. To ensure constant blending rates during a run, we develop a novel and efficient procedure that solves successive MILPs instead of a nonconvex MINLP. We use 14 examples with varying sizes and features to illustrate the superiority and effectiveness of our formulation and solution approach. The results show that our solution approach is superior to commercial solvers (BARON and DICOPT). © 2009 American Institute of Chemical Engineers *AIChE J*, 56: 441–465, 2010
Keywords: gasoline, recipe, blending, mixed-integer nonlinear programming, nonconvex, property index

Introduction

Ever-changing crude prices, deteriorating crude qualities, fluctuating demands for products, and growing environmental concerns are squeezing the profit margins of modern oil refineries like never before. Optimal scheduling of various operations in a refinery offers significant opportunities for saving costs and increasing profits. The overall refinery operations[1,2] involve three main segments, namely crude oil storage and processing, intermediate processing, and product blending. Scheduling of crude oil operations[3–5] has received the most attention so far. However, only limited work exists on the scheduling of product blending operations.

Gasoline is one of the most profitable products of a refinery and can account for as much as 60–70% of total profit. A refinery typically blends several gasoline cuts or fractions from various processes to meet its customer orders of varying specifications. However, this process involves nonlinear blending and complex combinatorics, and can easily result in suboptimal schedules and costly quality give-aways. The large numbers of orders, delivery dates, blenders, blend components, tanks, quality specifications, mixing timing, product

Correspondence concerning this article should be addressed to I. A. Karimi at cheiak@nus.edu.sg

certification, etc. make this problem highly complex and nonlinear. Optimal scheduling using advanced techniques of mixed-integer programming are imperative for avoiding ship demurrage, improving order delivery and customer satisfaction, minimizing quality give-aways, reducing transitions and slop generation, increasing productivity and asset utilization, exploiting low-quality cuts, etc. Therefore, scheduling of gasoline blending operations is important.

The early work focused mainly on gasoline blending planning rather than scheduling. It focused on the optimal blending of various intermediate fractions from the refinery and some additives to meet product quality specifications and demand requirements. Dewitt et al.[6] developed a decision support system named OMEGA (optimization method for the estimation of gasoline attributes) for gasoline blending operations. They used detailed nonlinear models for predicting gasoline attributes. Rigby et al.[7] improved OMEGA to a multi-period blending model named StarBlend. Li et al.[8] integrated CDU, FCC, and product blending models into refinery planning and used a linear correlation for octane number (ON) prediction in gasoline blending and a nonlinear correlation for pour point prediction in diesel blending. Some commercial tools such as Aspen Blend, Aspen PIMS-MBO, Aspen's ORION, and Honeywell's BLEND are also restricted to product-blending planning problems and do not involve detailed scheduling decisions.

For scheduling product-blending operations, Pinto et al.[1] developed an MILP model for scheduling refinery production involving blending operations. Although they considered transitions in blending pipelines, they did not ensure constant blending rates and did not enforce minimum run lengths[9] for blend runs. In addition, they used linear blending correlations for key elements, whereas did not handle most nonlinear product properties such as sulfur. Joly and Pinto[10] also developed an MINLP formulation for scheduling fuel oil/asphalt production. This also involved blending operations, but they made the same assumptions as Pinto et al.[1]

Glismann and Gruhn[11] developed a two-level decomposition approach to integrate short-term scheduling with blend recipe optimization. They first solve a nonlinear programming (NLP) problem to obtain product recipes and quantities, and then use mixed-integer linear programming (MILP) to obtain a schedule for the blending operation for the fixed product recipes. Jia and Ierapetritou[2] proposed a continuous-time event-based MILP formulation for scheduling gasoline blending and distribution operations simultaneously. They allowed multipurpose product tanks, one product tank delivering multiple orders, and multiple product tanks delivering one order. However, their model lacked some key operational features such as multiple parallel nonidentical blenders, variable recipes, and product specifications. Moreover, they allowed a blender to feed multiple product tanks at the same time, which may not be a normal practice. However, more importantly, their formulation gives infeasible solutions, and allows a product tank to hold several products at a time (See Appendix A for details).

Recently, Mendez et al.[12] presented both discrete-time and slot-based continuous-time models for the simultaneous optimization of blending and short-term scheduling. Their model allowed parallel identical blenders and determined optimal blend recipes. However, they employed nonlinear correlations for some product specifications, which resulted in a nonconvex MINLP. To solve this MINLP, they proposed an iterative algorithm that first uses linear correlations to obtain component volume fractions in each blend and computes the correction factor "bias" between the nonlinear and linear estimates of product specifications. Then, the algorithm uses this "bias" to amend the linear correlations, until all product specifications meet their limits. Thus, their algorithm solves linearized MILPs. However, they did not consider multipurpose product tanks, and did not ensure constant blend rates or minimum run lengths.

Unlike other commercial planning tools mentioned earlier, Honeywell's Production Scheduler is a tool for product blending operations. It incorporates several features such as recipe, product quality, certification delay, filling a tank to "full" before switching to another tank, and multipurpose component tanks. However, it uses decomposition and heuristic methods to obtain a feasible schedule and nonlinear correlations to predict product qualities.[13–15]

The above literature review suggests that previous work considered only pieces of the full gasoline-blending problem. Furthermore, the work that did address product specifications used nonlinear correlations, which makes the problem more difficult. An integrated treatment of recipe, blending, and scheduling is missing. In this article, we address most of the drawbacks of the existing work, and develop formulations that incorporate several real-life operation features such as multipurpose product tanks, parallel nonidentical blenders, constant rates during blending runs, minimum run lengths, changeovers, linear property indices,[5,16] piecewise constant profiles for blend component qualities and feed rates, etc. We allow the blend component flow rates and qualities to be piecewise constant over the horizon by means of a multi-period formulation. Although the formulation is nonlinear and nonconvex, when we enforce a constant rate during each blending run, we develop a novel schedule-adjustment procedure that solves only MILPs and no MINLP.

We begin with a detailed problem statement. Based on this statement, we develop a single-period mathematical formulation. Following that, we propose a novel procedure that addresses the nonlinearity arising from forcing constant blending rates. Then, we extend the single-period formulation to the multi-period scenario. Next, we illustrate our proposed schedule adjustment procedure with a small example, and evaluate our model and procedure with 13 additional examples. Lastly, we compare our procedure with commercial MINLP solvers (DICOPT/GAMS[17] and BARON/GAMS[17]).

Problem Statement

Consider a gasoline-blending unit (GBU) in a typical refinery (Figure 1). It employs I component tanks ($i = 1, 2, \ldots, I$), B blenders ($b = 1, 2, \ldots, B$), J product tanks ($j = 1, 2, \ldots, J$), and some lifting ports. The GBDU uses I components ($i = 1, 2, \ldots, I$) to make P possible products ($p = 1, 2, \ldots, P$). These components are gasoline fractions from various processes in a refinery such as CDU (crude distillation unit), FCCU (fluid catalytic cracking unit), FCRU (fluid catalytic reforming unit), AKU (alkanisation unit), IFU (isoforming unit), CHU (catalytic hydrogenation unit), ARU (aromatization unit), and various additives such as MTBE (methyl tert-butyl ether) and butane to enhance octane rating

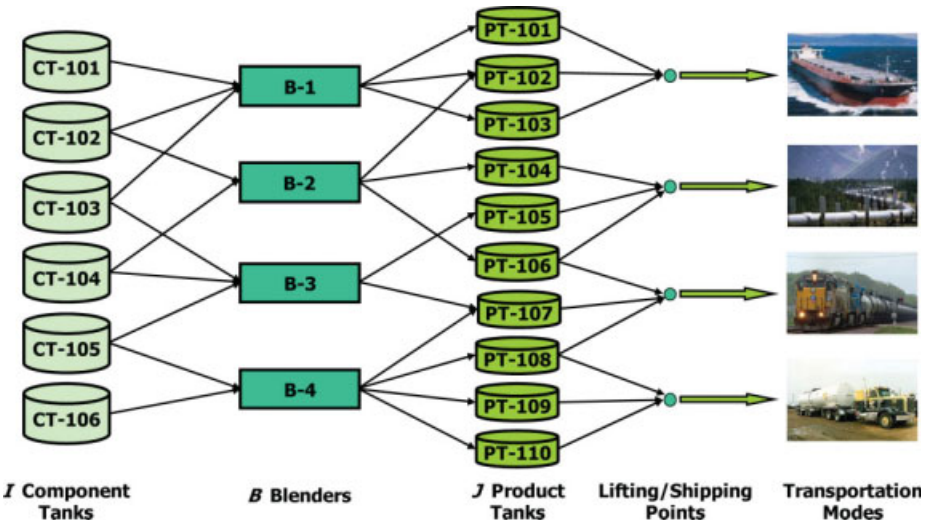

Figure 1. Schematic of gasoline blending operation.

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

or act as corrosion inhibitors or lubricators. Thus, different grades of gasoline contain different components. Each component has a pre-fixed, distinct, and known quality or specification, and component i is stored in its own dedicated component tank i. Note that even if several component tanks store one component, then we can treat them as one tank with no loss of generality. At time zero, it has O orders ($o = 1, 2, \ldots, O$) to fulfill during the coming scheduling horizon $[0, H]$. Each order o has a time window $[DD_o^L, DD_o^U]$ for delivery. An order may involve multiple products, but we assume with no loss of generality that each involves a single product, as each multi-product order can be broken into several single-product orders. Any delivery after DD_o^U incurs a demurrage cost (DM_o).

The quality of blend components is specified in terms of various property indices such as RBN (Research octane number index), RVI (Reid vapor pressure index), etc. The flow profiles over time of various blend components into respective component tanks are known a priori. The blend components from various component tanks are fed to the blenders in some proportions to make various products of desired quality at various times. The blenders are semi-continuous units that process products one at a time. The blended products from these units flow to assigned product tanks that may hold different products over time. The products from product tanks are loaded into vehicles or ships at appropriate times for the delivery of various orders.

The operation of this typical GBDU involves decisions such as recipe determination, allocation of component tanks to blenders, assignment of product tanks to products over time, and scheduling of blending, transfer, and delivery operations. With this, the gasoline-blending problem addressed in this article can be stated as:

Given:

1. A scheduling horizon $[0, H]$.
2. I components and their property indices.
3. I component tanks, their initial inventories, limits on their holdups, flow profiles of feeds into the tanks, and limits on the flows out of the tanks.
4. P products and specification limits on their property indices.
5. B blenders, the products that each blender can process, lower limits on the blend times of these products, and limits on their blending rates.
6. Product tanks, the products that each tank can store, limits on their holdups, the products and holdups at time zero, and delivery (lifting) rates for various products.
7. O orders, their constituent products, amounts, and delivery time windows.
8. Component costs, transition costs in blenders and product tanks, and demurrage costs for orders.

Determine:

1. The blenders that each component tank should feed over time, and the feed flow rates.
2. The products that each blender should produce over time, and the blending rates.
3. The products that each product tank should receive over time from various blenders, and the corresponding transfer rates.
4. The orders that each product tank should deliver over time, and their amounts.
5. The inventory profiles of various tanks (component and product).

Allowing:

1. A component tank may feed multiple blenders, and a blender may receive from multiple component tanks at the same time.
2. A blender may feed multiple product tanks during the scheduling horizon.
3. A product tank may deliver multiple orders at the same time.
4. Multiple tanks may deliver an order at the same time.

Subject to the operating rules:

1. A blender can process at most one product at any time. Once it begins processing a product, it must operate for some minimum time, before it can switch to another product.
2. A blender can feed at most one product tank at any time. In addition to being the industry practice, this helps

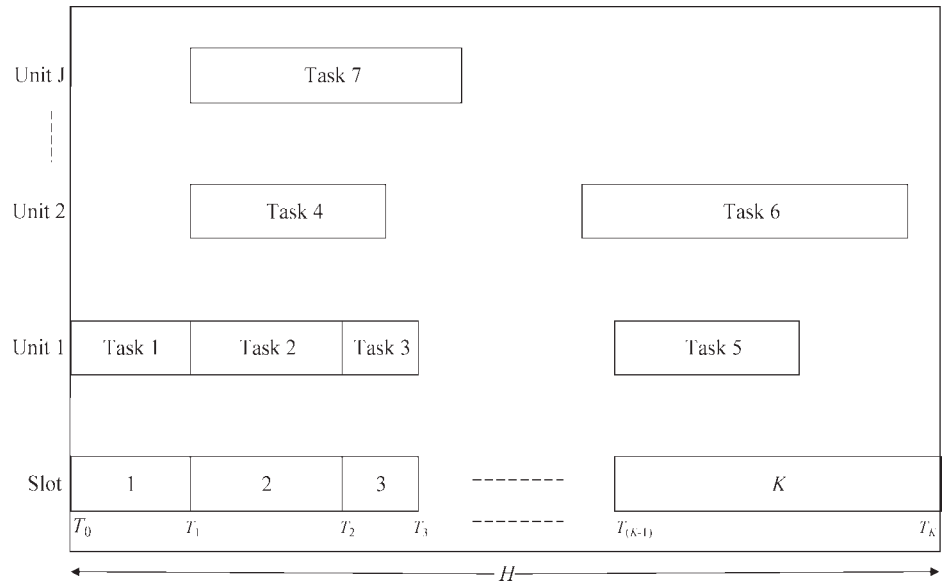

Figure 2. Schematic of slot design.

to decrease the number of tanks in use and increase their utilization.

3. A product tank cannot receive and deliver a product simultaneously. This is because some additional mixing time and/or product certification may usually be necessary after a product tank receives a product from blenders.

Assuming:

1. Flow rate profile of each component from the upstream process is piecewise constant.
2. Component inventories are sufficient for blending through the entire scheduling horizon.
3. Mixing in each blender is perfect.
4. Changeover times between products are negligible for both blenders and product tanks.
5. Each order involves only one product. As discussed earlier, each multi-product order can be decomposed into several single-product orders.
6. Each order is completed during the scheduling horizon.
7. The qualities of products in product tanks at time zero satisfy desired specs.
8. Time for mixing in product tanks is zero.
9. No additional time is required for product certification. This will be the case, when the certification begins sufficiently in advance during the blending process so that no delay is required between the receipt into and delivery from a product tank.

We now develop a MILP formulation for the above problem. However, for the sake of simplicity, we first consider the simplest scenario in which the flow rates of all components are constant over the entire scheduling horizon. This is also what most existing work assumes. In addition, most existing work considers a single or multiple identical blenders.

Single-Period MILP

We divide the horizon H into K ($k = 1, 2, \ldots, K$) process-slots[18] of variable lengths (SL_k). $k = 0$ denotes the time just before zero time. The process-slots are common to or synchronized across all units (tanks and blenders). Denoting T_0 as the start of the horizon and the end of slot $k = 0$, and T_k as the time at which slot k ends, we have

$$T_k = T_{(k-1)} + \mathrm{SL}_k \quad T_0 = 0, \quad 0 < k \leq K \tag{1}$$

with H as the upper bound of T_k for all $k > 0$. Figure 2 shows the schematic of our slot design. We assume that each new operation (except idling) on a unit (tank or blender) begins at the start of a slot, but may end at any time within a slot.

Throughout this article, each variable is defined with specific ranges of its indices, and each constraint, unless otherwise indicated, is written for all valid values of the indices of its constituent variables.

Blending and Storage

At any time, a blender must be either running or idle. When running, it must be connected to a product tank. If idle, then we connect it to a dummy product tank ($j = 0$). Thus, we have J real product tanks ($j = 1, 2, \ldots, J$) and one ($j = 0$) dummy product tank. Now, we define $\boldsymbol{BJ} = \{(b, j) \mid$ blender b can feed product tank $j\}$ and a binary variable (v_{bjk})

$$v_{bjk} = \begin{cases} 1 & \text{If blender } b \text{ feeds product tank } j \text{ during slot } k \\ 0 & \text{Otherwise} \end{cases}$$
$$(b,j) \in \boldsymbol{BJ}, \quad 0 \leq j \leq J, \quad 0 \leq k \leq K$$

We treat v_{bjk} ($j = 1, 2, \ldots, J$) as binary and v_{b0k} as 0–1 continuous variable.

Each blender must feed exactly one product tank (real or dummy) in each slot

$$\sum_{j=0}^{J} v_{bjk} = 1 \quad (b,j) \in \boldsymbol{BJ}, \quad 0 < k \leq K \tag{2}$$

Let G_{bjk} be the volume that blender b feeds product tank j during slot k. If blender b does not feed tank j during slot k, then G_{bjk} must be zero

$$G_{bjk} \le \mathrm{VP}_j^U v_{bjk} \quad (b,j) \in \boldsymbol{BJ}, \quad 0 < j \le J, \quad 0 \le k \le K \quad (3)$$

where, $\mathrm{VP}_j^{\mathrm{U}}$ is the maximum capacity of tank j.

To model the holdup in product tanks, we define $\boldsymbol{PJ} = \{(p, j) \mid \text{product tank } j \text{ can hold product } p\}$ and one binary variable (u_{jpk}) as follows:

$$u_{jpk} = \begin{cases} 1 & \text{If product tank } j \text{ holds product } p \text{ during slot } k \\ 0 & \text{Otherwise} \end{cases} \qquad (p,j) \in \boldsymbol{PJ}, \quad 0 < j \le J, \quad 0 \le k \le K$$

Note that $u_{jpk} = 1$ allows tank j to have a zero holdup of p during k. This saves us a variable for modeling the state of an empty tank. Thus, each tank must hold exactly one product in each slot

$$\sum_{p=1}^{P} u_{jpk} = 1 \quad (p,j) \in \boldsymbol{PJ}, \quad 0 < j \le J, \quad 0 < k \le K \quad (4)$$

Note that u_{jp0} must have an appropriate value based on what was inside tank j before time zero. To model product transitions in product tank j, we define a 0–1 continuous variable ue_{jk}

$$ue_{jk} = \begin{cases} 1 & \text{If tank } j \text{ switches products} \\ & \quad \text{at the end of slot } k \qquad 0 < j \le J, \quad 0 \le k \le K \\ 0 & \text{Otherwise} \end{cases}$$

$$ue_{jk} \ge u_{jpk} - u_{jp(k+1)} \quad (p,j) \in \boldsymbol{PJ}, \quad 0 \le k < K, \quad 0 < j \le J \quad (5a)$$

$$ue_{jk} \ge u_{jp(k+1)} - u_{jpk} \quad (p,j) \in \boldsymbol{PJ}, \quad 0 \le k < K, \quad 0 < j \le J \quad (5b)$$

We need not force $ue_{jk} = 0$, as we will impose a penalty for product changeovers in the objective. Now, a product transition cannot occur, unless the tank holdup (VP_{jk}, $0 \le \mathrm{VP}_{jk} \le \mathrm{VP}_j^U$) at the end of slot k is zero. Thus,

$$\mathrm{VP}_{jk} \le \mathrm{VP}_j^U \left(1 - ue_{jk}\right) \quad 0 < k < K \quad (6)$$

To model the blending operation, we define $\mathbf{BP} = \{(b, p) \mid \text{blender } b \text{ can process product } p\}$, and the following 0–1 continuous variables

$$x_{bpk} = \begin{cases} 1 & \text{If blender } b \text{ processes product } p \text{ during slot } k \\ 0 & \text{Otherwise} \end{cases} \qquad (b,p) \in \boldsymbol{BP}, \quad 0 \le k \le K$$

$$xe_{bk} = \begin{cases} 1 & \text{If blender } b \text{ ends the current} \\ & \quad \text{product run during slot } k \qquad 0 \le k < K \\ 0 & \text{Otherwise} \end{cases}$$

We now relate x_{bpk}, u_{jpk}, and v_{bjk} to force x_{bpk} to be binary. First, if blender b is feeding a real tank j and that tank is holding product p during slot k, then b must be processing p in slot k

$$x_{bpk} \ge u_{jpk} + v_{bjk} - 1 \qquad (b,p) \in \boldsymbol{BP}, \quad (b,j) \in \boldsymbol{BJ}, (p,j) \in \boldsymbol{PJ}, \quad 0 < j \le J, \quad 0 < k \le K \quad (7a)$$

Similarly, if b is processing p and feeding j during slot k, then j must hold p during k.

$$u_{jpk} \ge x_{bpk} + v_{bjk} - 1 \qquad (b,p) \in \boldsymbol{BP}, (b,j) \in \boldsymbol{BJ}, \quad (p,j) \in \boldsymbol{PJ}, \quad 0 < j \le J, \quad 0 < k \le K \quad (7b)$$

Note that Eq. 7 are not written for p that j cannot store, i.e. $(p, j) \notin \boldsymbol{PJ}$. For such products, we write

$$x_{bpk} + \sum_{j \notin \boldsymbol{PJ}} v_{bjk} \le 1 \qquad (b,p) \in \boldsymbol{BP}, \quad (b,j) \in \boldsymbol{BJ}, \quad 0 < j \le J, \quad 0 < k \le K \quad (8a)$$

$$\sum_{p \notin \boldsymbol{PJ}} x_{bpk} + v_{bjk} \le 1 \qquad (b,p) \in \boldsymbol{BP}, \quad (b,j) \in \boldsymbol{BJ}, \quad 0 < j \le J, \quad 0 < k \le K \quad (8b)$$

If a blender is not idle, then it must process exactly one product. In other words,

$$\sum_{p=1}^{P} x_{bpk} + v_{b0k} = 1 \quad (b,p) \in \boldsymbol{BP}, \quad 0 < k \le K \quad (9)$$

Equations 2, 4, 7, and 9 make x_{bpk} binary (Proof in Appendix B). Note that we must assign proper values for x_{bp0} based on the product that blender b was processing before time zero.

Using x_{bpk}, we compute xe_{bk} as follows. If a blender processes the same product in two consecutive slots (k and $k + 1$), then the current blend run cannot end at slot k, and vice versa. In other words,

$$xe_{bk} + x_{bpk} + x_{bp(k+1)} \le 2 \quad (b,p) \in \boldsymbol{BP}, \quad 0 \le k < K \quad (10)$$

On the contrary, if a product does not continue in the next slot, then its run must end at slot k

$$xe_{bk} \ge x_{bpk} - x_{bp(k+1)} \quad (b,p) \in \boldsymbol{BP}, \quad 0 \le k < K \quad (11a)$$

$$xe_{bk} \ge x_{bp(k+1)} - x_{bpk} \quad (b,p) \in \boldsymbol{BP}, \quad 0 \le k < K \quad (11b)$$

Next, we define RL_{bk} as the length of the current run of blender b at the end of slot k, if the run does not end during slot k, and zero otherwise. In other words,

$$\mathrm{RL}_{bk} = \begin{cases} \text{Current run length} & \text{If the current run of blender } b \text{ does not end during slot } k \\ 0 & \text{Otherwise} \end{cases} \qquad 1 \le b < B, \quad 0 \le k \le K$$

Thus, $\mathrm{RL}_{b0} = 0$, if a run has ended at time zero, otherwise it is the current run length at time zero. To compute RL_{bk}, we write

$$\mathrm{RL}_{bk} \le \mathrm{RL}_{b(k-1)} + \mathrm{SL}_k \qquad 0 < k \le K \tag{12}$$

$$\mathrm{RL}_{bk} \le H(1 - xe_{bk}) \qquad 0 \le k \le K \tag{13}$$

Then, to ensure a minimum length ($\mathrm{RL}^{\mathrm{L}}_{bp}$) for each blend run, we demand

$$\mathrm{RL}_{b(k-1)} + \mathrm{SL}_k + \mathrm{RL}^{L}_{b}(1 - xe_{bk}) \ge \sum_{p=1}^{P} \mathrm{RL}^{L}_{bp} x_{bpk} \qquad (b,p) \in \boldsymbol{BP}, \quad 0 < k \le K \tag{14}$$

where, $\mathrm{RL}^{\mathrm{L}}_{b} = \max_{p}\left(\mathrm{RL}^{\mathrm{L}}_{bp}\right)$.

Blending requires components. Let q_{ibk} be the volume of component i used by blender b during slot k. Recall that G_{bjk} is the volume that blender b feeds product tank j during slot k. Then, the volume (Q_{bk}) processed in blender b during slot k is

$$Q_{bk} = \sum_{i=1}^{I} q_{ibk} \qquad 0 < k \le K \tag{15a}$$

$$Q_{bk} = \sum_{j=1}^{J} G_{bjk} \quad (b,j) \in \boldsymbol{BJ}, \quad 0 < k \le K \tag{15b}$$

If blender b is idle during slot k, then this volume must be zero

$$Q_{bk} \le M_b(1 - v_{b0k}) \qquad 0 < k \le K \tag{16}$$

where, M_b is the most volume that blender b can process during a slot. There are several ways of estimating M_b. One estimate is HF^{U}_{b}. Another is the maximum number in the maximum capacities of all product tanks because a blender can feed at most one product tank in a slot. The minimum possible estimate should be used as M_b. In other words, $M_b = \min\left\{H F^{\mathrm{U}}_{b}, \max_{j}\left(\mathrm{VP}^{\mathrm{U}}_{j}\right)\right\}$.

If blender b is not idle during slot k, then Q_{bk} must be limited by the maximum processing rate (F^{U}_{b}) of blender b

$$Q_{bk} \le F^{U}_{b}\mathrm{SL}_k \qquad 0 < k \le K \tag{17a}$$

On the other hand, it must also respect the minimum processing rate (F^{L}_{b}) of blender b for each product, unless the current run is ending during slot k. In other words,

$$Q_{bk} + F^{L}_{b}H(v_{b0k} + xe_{bk}) \ge F^{L}_{b}\mathrm{SL}_k \qquad 0 < k \le K \tag{17b}$$

To compute the volume of product processed during a run up to slot k, we define CQ_{bk} analogous to RL_{bk} as follows:

$$\mathrm{CQ}_{bk} = \begin{cases} \text{Volume processed} & \text{If blender } b \text{ does not end its run during slot } k \\ 0 & \text{Otherwise} \end{cases}$$

$$\mathrm{CQ}_{bk} \le \mathrm{CQ}_{b(k-1)} + Q_{bk} \qquad 0 < k \le K \tag{18}$$

$$\mathrm{CQ}_{bk} \le F^{U}_{b}H(1 - xe_{bk}) \qquad 0 < k \le K \tag{19}$$

$$\mathrm{CQ}_{b(k-1)} + Q_{bk} + F^{L}_{b}\mathrm{RL}^{L}_{b}(1 - xe_{bk}) \ge F^{L}_{b}\sum_{p=1}^{P} \mathrm{RL}^{L}_{bp} x_{bpk} \qquad (b,p) \in \boldsymbol{BP}, \quad 0 < k \le K \tag{20}$$

Note that Eq. 17 allows the blending rate to vary from slot to slot during a run. Normally, this is not done in practice. However, enforcing this makes the formulation nonlinear and nonconvex. Therefore, we have decided to deal with this issue later.

Product Quality

Clearly, each blend run must ensure product quality. Several gasoline properties such as ON, Reid vapor pressure (RVP), specific gravity (SG), sulfur (S), benzene (B), aromatics (A), olefin (O), residue (R), existent gum (EG), flammability limit (FL), oxygenates (OX), phosphorous (PH), oxidation stability (OS), copper corrosion (CC), silver corrosion (SC), initial boiling point (IBP), final boiling point (FBP), manganese (Mg), etc. are used in practice. Many of these properties (e.g. RVP) involve highly nonlinear mixing rules. However, as noted by Li et al.,[5] a linear blending index usually exists and is used for almost every hydrocarbon property with nonlinear mixing correlations. These blending indices are linearly additive on either volume or weight basis. Table 1 lists several gasoline properties, indices, and their additive bases. Let θ_{is} be the known blending index for a property s of component i, ρ_i be the density of component i, $[\theta^{\mathrm{L}}_{ps}, \theta^{\mathrm{U}}_{ps}]$ be the desired limits on property s of product p, and $\rho_{\max}$ be the maximum possible density among all products. Then, the following ensure the desired product quality

$$\sum_{i=1}^{I} q_{ibk}\theta_{is} \ge Q_{bk}\theta^{L}_{ps} - M_b\left\{\theta^{L}_{ps} - \min_{p}\left(\theta^{L}_{ps}\right)\right\}(1 - x_{bpk}) \qquad (b,p) \in \boldsymbol{BP}, \quad 0 < k \le K \tag{21a}$$

$$\sum_{i=1}^{I} q_{ibk}\theta_{is} \le Q_{bk}\theta^{U}_{ps} + M_b\left\{\max_{p}\left(\theta^{U}_{ps}\right) - \theta^{U}_{ps}\right\}(1 - x_{bpk}) \qquad (b,p) \in \boldsymbol{BP}, \quad 0 < k \le K \tag{21b}$$

$$\sum_{i=1}^{I} q_{ibk}\rho_i\theta_{is} \ge \left(\sum_{i=1}^{I} q_{ibk}\rho_i\right)\theta^{L}_{ps} - M_b\left\{\theta^{L}_{ps} - \min_{p}\left(\theta^{L}_{ps}\right)\right\}\rho_{\max}(1 - x_{bpk}) \qquad (b,p) \in \boldsymbol{BP}, \quad 0 < k \le K \tag{22a}$$

$$\sum_{i=1}^{I} q_{ibk}\rho_i\theta_{is} \le \left(\sum_{i=1}^{I} q_{ibk}\rho_i\right)\theta^{U}_{ps} + M_b\left\{\max_{p}\left(\theta^{U}_{ps}\right) - \theta^{U}_{ps}\right\}\rho_{\max}(1 - x_{bpk}) \qquad (b,p) \in \boldsymbol{BP}, \quad 0 < k \le K \tag{22b}$$

where, Eq. 21 is for volume-based indices, and Eq. 22 for weight-based indices.

Table 1. Gasoline Properties Corresponding Indices, and Correlations

Gasoline Property*	Blending Index	Addition Basis	Index Correlation
Research Octane Number (RON)	RBN	Volume	RON + 11.5 (0 ≤ RON ≥ 85) Exp(0.0135 RON + 3.422042) (RON > 85)
Reid Vapor pressure (RVP Bar)	RVI	Volume	Exp(1.14 log (100 RVP))
Specific-gravity (SG)	DNI	Volume	1/SG
Sulfur (S ppm)	SULI	Weight	Weighted average
Benzene (B)	BI	Volume	Volumetric average
Aromatics (A)	AROI	Volume	Volumetric average
Olefin (O)	OI	Volume	Volumetric average
Residue (R)	RI	Volume	Volumetric average
Existent Gum (EG mg/l00 ml)	EGI	Volume	Volumetric average
Flammability limit (FL)	FLI	Volume	Volumetric average
Oxygenates (OX)	OXI	Weight	Weighted average
Phosphorous (PH g/gal)	PHI	Volume	Volumetric average
Oxidation Stability (OS minutes)	OSI	N/A	N/A
Copper corrosion (CC)	CCI	N/A	N/A
Silver Corrosion (SC)	SCI	N/A	N/A
Initial boiling point (IBP°C)	IBPI	N/A	N/A
Final boiling point (FBP°C)	FBPI	N/A	N/A
Manganese (MG)	MGI	Volume	Volumetric average

*Index correlations from Singapore Petroleum Company (SPC).

Often, a practitioner may impose limits $[r_{pi}^{\mathrm{L}}, r_{pi}^{\mathrm{U}}]$ on the volume fraction of component i in product p. In such a case, we use

$$Q_{bk} r_{pi}^{L} - M_b \left\{ r_{pi}^{L} - \min_p \left(r_{pi}^{L} \right) \right\} (1 - x_{bpk}) \le q_{ibk} \le Q_{bk} r_{pi}^{U} + M_b \left\{ \max_p \left(r_{pi}^{U} \right) - r_{pi}^{U} \right\} (1 - x_{bpk}) \quad (b,p) \in \boldsymbol{BP}, \quad 0 < k \le K \quad (23a, b)$$

Order Delivery

We define $\boldsymbol{JO} = \{(j, o) \mid$ product tank j can deliver order $o\}$ and one binary variable (z_{jok}) as follows:

$$z_{jok} = \begin{cases} 1 & \text{If product tank } j \text{ is delivering order } o \text{ during slot } k \\ 0 & \text{Otherwise} \end{cases} \quad (j,o) \in \boldsymbol{JO}, \quad 0 < j \le J, \quad 0 < k \le K$$

Since each order must be filled during the scheduling horizon, there must be at least one delivery for each order

$$\sum_{j=1}^{J} \sum_{k=1}^{K} Z_{jok} \ge 1 \quad (j,o) \in \boldsymbol{JO} \quad (24)$$

A tank j cannot receive and deliver products simultaneously

$$v_{bjk} + z_{jok} \le 1 \quad (b,j) \in \boldsymbol{BJ}, \quad (j,o) \in \boldsymbol{JO}, \quad 0 < j \le J, \quad 0 < k \le K \quad (25)$$

If tank j is delivering order o during slot k, then it must be holding the product corresponding to that order in both slots $(k - 1)$ and k. This is true, because a tank cannot receive and deliver at the same time, so it must be holding the product in $(k - 1)$, before it delivers in slot k

$$z_{jok} \le u_{jpk} (p,j) \in \boldsymbol{PJ}, (j,o) \in \boldsymbol{JO}, (o,p) \in \boldsymbol{OP}, \quad 0 < j \le J, \quad 0 < k \le K \quad (26a)$$

$$z_{jok} \le u_{jp(k-1)} (p,j) \in \boldsymbol{PJ}, (j,o) \in JO, (o,p) \in \boldsymbol{OP}, \quad 0 < j \le J, \quad 0 < k \le K \quad (26b)$$

where, $\boldsymbol{OP} = \{(o, p) \mid$ order o is for product $p\}$. Recall that each order has a single product.

If a product tank j switches products at the end of a slot k, then its holdup must be zero. Thus, it cannot deliver any order in $(k + 1)$

$$z_{jo(k+1)} + ue_{jk} \le 1 \quad (j,o) \in \boldsymbol{JO}, \quad 0 < j \le J, \quad 0 \le k < K \quad (27)$$

Let DQ_{jok} be the volume of order o delivered by tank j during slot k. If tank j is not delivering o during k, then the delivery amount must be zero:

$$\mathrm{DQ}_{jok} \le \mathrm{TQ}_o z_{jok} \quad (j,o) \in \boldsymbol{JO}, \quad 0 < j \le J, \quad 0 < k \le K \quad (28)$$

where, TQ_o is the required amount for order o. In general, depending on its pump/valve infrastructure, each tank will have some limits on the delivery rates of orders. Let us assume that the maximum possible delivery rate for an order o is DR_{jo} and the maximum cumulative rate for all orders is $\mathrm{DR}_j^{\mathrm{U}}$. Then, we have

$$\mathrm{DQ}_{jok} \le \mathrm{DR}_{jo} \mathrm{SL}_k \quad (j,o) \in \boldsymbol{JO}, \quad 0 < j \le J, \quad 0 < k \le K \quad (29a)$$

$$\sum_{o \in JO} \mathrm{DQ}_{jok} \le \mathrm{DR}_j^{U} \mathrm{SL}_k \quad 0 < j \le J, \quad 0 < k \le K \quad (29b)$$

To ensure full delivery for each order, we use

$$\sum_{j=1}^{J}\sum_{k=1}^{K} \mathrm{DQ}_{jok} = \mathrm{TQ}_o \qquad (j,o) \in \boldsymbol{JO} \tag{30}$$

Recall that each order o has a delivery window ($\mathrm{DD}_o^{\mathrm{L}}$, $\mathrm{DD}_o^{\mathrm{U}}$). A tank j cannot begin delivering o before $\mathrm{DD}_o^{\mathrm{L}}$:

$$T_{k-1} \geq \mathrm{DD}_o^{L} z_{jok} \quad (j,o) \in \boldsymbol{JO}, \quad 0 < j \leq J, \quad 0 < k \leq K \tag{31}$$

If an order is not fully delivered before $\mathrm{DD}_o^{\mathrm{U}}$, then we have a delivery delay given by

$$d_o \geq T_{k-1} + \frac{\mathrm{DQ}_{jok}}{\mathrm{DR}_{jo}} - \mathrm{DD}_o^{U} - H\left(1 - z_{jok}\right) \qquad (j,o) \in \boldsymbol{JO}, \quad 0 < j \leq J, \quad 0 < k \leq K \tag{32}$$

Recall that all orders are completed within the scheduling horizon. Therefore, the delivery delay (d_o) of order o must have the upper bound of ($\mathrm{H}-\mathrm{DD}_o^{\mathrm{U}}$).

Note that we allowed order delivery to be intermittent from a tank. As we mentioned for blending, we propose a simple adjustment procedure later to correct this situation.

Inventory Balance

Let V_{ik} ($V_i^{\mathrm{L}} \leq V_{ik} \leq V_i^{\mathrm{U}}$) denote the inventory of component i at the end of slot k and ($V_i^{\mathrm{L}} \leq V_i \leq V_i^{\mathrm{U}}$) denote the inventory of component i at the end of scheduling horizon. Within any period, the feed rate (F_i) of i from upstream units is constant, so

$$V_{ik} = V_{i(k-1)} + F_i \mathrm{SL}_k - \sum_{b=1}^{B} q_{ibk} \quad 0 < k \leq K \tag{33a}$$

$$V_i = V_{ik} + F_i(H - T_K) \tag{33b}$$

Similarly, for product tank j, we have

$$\mathrm{VP}_{jk} = \mathrm{VP}_{j(k-1)} + \sum_{b=1}^{B} G_{bjk} - \sum_{o=1}^{O} \mathrm{DQ}_{jok} \qquad (j,o) \in \boldsymbol{JO}, \quad 0 < k \leq K \tag{34}$$

Transitions in Blenders

At the beginning, we know the amount of an order and its product. Then, we can calculate the total amount of each product needed. Having the initial amount of each product, we then know whether the product is needed to process in the blender or not. Let N denote the number of products that are needed to process in blenders during the scheduling horizon and N_b as the number of blenders. Recall that we use xe_{bk} as the product transition in blender b at the end of slot k and xe_{bK} is equal to one. We calculate the total minimum transitions during the scheduling horizon as follows:

$$\sum_{b=1}^{B}\sum_{k=1}^{K-1} xe_{bk} \geq N - N_b \tag{35}$$

Objective Function

For similar problems, Mendez et al.[12] maximized total profit, whereas Jia and Ierapetritou[2] minimized the makespan. Neither considered the transition costs for blenders and product tanks. For a given set of orders, we feel that minimizing the total operating cost including material (component), demurrage, transition, and backorder costs is more meaningful in practice. Assuming that the transition costs are product and sequence-independent (this is reasonable as product qualities are quite similar), our scheduling objective is

$$\text{Minimize TC} = \sum_{i=1}^{I}\sum_{b=1}^{B}\sum_{k=1}^{K} c_i\, q_{ibk} + \sum_{b=1}^{B}\sum_{k=1}^{K-1} \mathrm{CB}_b\, xe_{bk} + \sum_{j=1}^{J}\sum_{k=1}^{K-1} \mathrm{CT}_j\, ue_{jk} + \sum_{o=1}^{O} \mathrm{DM}_o\, d_o \tag{36}$$

where, c_i is the price (\$ per unit volume) of component i, CB_b is the cost (\$ per occurrence) of transition on blender b, CT_j is the cost (\$ per occurrence) of transition in product tank j, and DM_o is the demurrage cost (\$ per unit time) of order o.

This completes our single-period model (**SPM**) for scheduling blending operations, which comprises Eqs. 1–36. As mentioned before, it allows the blending rate to vary from slot to slot and order delivery to be discontinuous, which is undesirable in practice. Therefore, we need a procedure to adjust the solution from **SPM** to obtain a realistic schedule.

Schedule Adjustment

The optimal solution from **SPM** gives us the values of x_{bpk}, xe_{bk}, v_{bjk}, SL_k, RL_{bk}, Q_{bk}, and CQ_{bk}. We use $[x_{bpk}]$, $[xe_{bk}]$, $[\mathrm{SL}_k]$, $[\mathrm{RL}_{bk}]$, $[Q_{bk}]$, $[\mathrm{CQ}_{bk}]$, and $[v_{b0k}]$, respectively, to denote the optimal values of x_{bpk}, xe_{bk}, SL_k, RL_{bk}, Q_{bk}, CQ_{bk}, and v_{b0k} obtained from SPM. Note that $[\mathrm{RL}_{bk}]$ and $[\mathrm{CQ}_{bk}]$, respectively, may not be the correct length and volume of product processed by blender b when a run is in progress. Therefore, we compute the correct values for run lengths (CRL_{bk}) and volumes (CCQ_{bk}) as follows:

$$\mathrm{CRL}_{bk} = \mathrm{CCQ}_{bk} = 0 \qquad \text{if } [xe_{bk}] = 1 \tag{37a, b}$$

$$\mathrm{CRL}_{bk} = \mathrm{CRL}_{b(k-1)} + [\mathrm{SL}_k] \qquad \text{if } [xe_{bk}] = 0 \tag{38a}$$

$$\mathrm{CCQ}_{bk} = \mathrm{CCQ}_{b(k-1)} + [Q_{bk}] \qquad \text{if } [xe_{bk}] = 0 \tag{38b}$$

Then, we compute the total volume (TCQ_{bk}) processed by a blender in a run as

$$\mathrm{TCQ}_{bk} = 0 \qquad \text{if } [xe_{bk}] = 0 \tag{39a}$$

$$\mathrm{TCQ}_{bk} = \mathrm{CCQ}_{b(k-1)} + [Q_{bk}] \qquad \text{if } [xe_{bk}] = 1 \tag{39b}$$

Using the above parameters, we compute the blending rate (R_{bk}) for each blending run at the slot where it ends

$$R_{bk} = \max\left(F_b^L, \frac{\mathrm{CCQ}_{b(k-1)} + [Q_{bk}]}{\mathrm{CRL}_{b(k-1)} + [\mathrm{SL}_k]}\right) \text{ for } k \text{ with } [xe_{bk}] = 1 \text{ and } [v_{b0k}] = 0 \tag{40}$$

Then, we set R_{bk} for all slots within each run to be the same as the one computed above. Thus, if a run spans slots 3–6 inclusive, then we set $R_{bk} = R_{b6}$ for $k = 3$–5.

Now, to obtain a realistic schedule with the constant blend rates computed above, we fix x_{bpk}, xe_{bk}, and v_{b0k}. This allows us to fix, remove, or change some variables and constraints in SPM.

Equations 16–20 become

$$Q_{bk} = 0 \quad \text{for } (b,k) \text{ with } [v_{b0k}] = 1 \tag{41a}$$

$$Q_{bk} = R_{bk}\mathrm{SL}_k \quad \text{for } (b,k) \text{ with } [xe_{bk}] = [v_{b0k}] = 0 \tag{41b}$$

$$Q_{bk} \le R_{bk}\,\mathrm{SL}_k \quad \text{for } (b,k) \text{ with } [xe_{bk}] = 1 \text{ and } [v_{b0k}] = 0 \tag{41c}$$

$$\mathrm{CQ}_{bk} = 0 \quad \text{for } (b,k) \text{ with } [xe_{bk}] = 1 \tag{42a}$$

$$\mathrm{CQ}_{bk} = \mathrm{CQ}_{b(k-1)} + Q_{bk} \quad \text{for } (b,k) \text{ with } [xe_{bk}] = [v_{b0k}] = 0 \tag{42b}$$

$$\mathrm{CQ}_{b(k-1)} + Q_{bk} \ge \mathrm{TCQ}_{bk} \quad \text{for } (b,k) \text{ with } [xe_{bk}] = 1 \text{ and } [v_{b0k}] = 0 \tag{42c}$$

Fixing the values of x_{bpk}, xe_{bk}, and v_{b0k}, Eqs. 2, 7–8, 21–23, and 36 become

$$\sum_{j=1}^{J} v_{bjk} = 1 - [v_{b0k}] \quad (b,j) \in \mathbf{BJ},\ 0 < j \le J,\ 0 < k \le K \tag{43}$$

$$[x_{bpk}] \ge u_{jpk} + v_{bjk} - 1 \qquad (b,p) \in \boldsymbol{BP}, (b,j) \in \boldsymbol{BJ}, (p,j) \in \boldsymbol{PJ}, \quad 0 < j \le J,\ < k \le K \tag{44a}$$

$$u_{jpk} \ge [x_{bpk}] + v_{bjk} - 1 \qquad (b,p) \in \boldsymbol{BP}, (b,j) \in \boldsymbol{BJ}, (p,j) \in \boldsymbol{PJ}, \quad 0 < j \le J, \quad 0 < k \le K \tag{44b}$$

$$[x_{bpk}] + \sum_{j \notin PJ} v_{bjk} \le 1 \qquad (b,p) \in \boldsymbol{BP}, (b,j) \in \boldsymbol{BJ}, \quad 0 < j \le J, \quad 0 < k \le K \tag{45a}$$

$$\sum_{p \notin PJ} [x_{bpk}] + v_{bjk} \le 1 \qquad (b,p) \in \boldsymbol{BP}, (b,j) \in \boldsymbol{BJ}, \quad 0 < j \le J, \quad 0 < k \le K \tag{45b}$$

$$\sum_{i=1}^{I} q_{ibk}\theta_{is} \ge Q_{bk}\theta^{L}_{ps} - M_b\left\{\theta^{L}_{ps} - \min_p\left(\theta^{L}_{ps}\right)\right\}\left(1 - [x_{bpk}]\right) \quad \text{for } (b,k) \text{ with } [v_{b0k}] = 0, (b,p) \in \boldsymbol{BP}, \quad 0 < k \le K \tag{46a}$$

$$\sum_{i=1}^{I} q_{ibk}\theta_{is} \le Q_{bk}\theta^{U}_{ps} + M_b\left\{\max_p\left(\theta^{U}_{ps}\right) - \theta^{U}_{ps}\right\}\left(1 - [x_{bpk}]\right) \quad \text{for } (b,k) \text{ with } [v_{b0k}] = 0, (b,p) \in \boldsymbol{BP}, \quad 0 < k \le K \tag{46b}$$

$$\sum_{i=1}^{I} q_{ibk}\rho_i\theta_{is} \ge \left(\sum_{i=1}^{I} q_{ibk}\rho_i\right)\theta^{L}_{ps} - M_b\left\{\theta^{L}_{ps} - \min_p\left(\theta^{L}_{ps}\right)\right\}\rho_{\max}\left(1 - [x_{bpk}]\right) \quad \text{for } (b,k) \text{ with } [v_{b0k}] = 0, \quad (b,p) \in \boldsymbol{BP}, \quad 0 < k \le K \tag{47a}$$

$$\sum_{i=1}^{I} q_{ibk}\rho_i\theta_{is} \le \left(\sum_{i=1}^{I} q_{ibk}\rho_i\right)\theta^{U}_{ps} + M_b\left\{\max_p\left(\theta^{U}_{ps}\right) - \theta^{U}_{ps}\right\}\rho_{\max}\left(1 - [x_{bpk}]\right) \quad \text{for } (b,k) \text{ with } [v_{b0k}] = 0, \quad (b,p) \in \boldsymbol{BP}, \quad 0 < k \le K \tag{47b}$$

$$Q_{bk}r^{L}_{pi} - M_b\left\{r^{L}_{pi} - \min_p\left(r^{L}_{pi}\right)\right\}\left(1 - [x_{bpk}]\right) \le q_{ibk} \le Q_{bk}r^{U}_{pi} + M_b\left\{\max_p\left(r^{U}_{pi}\right) - r^{U}_{pi}\right\}\left(1 - [x_{bpk}]\right) \quad \text{for } (b,k) \text{ with } [v_{b0k}] = 0, (b,p) \in \boldsymbol{BP}, \quad 0 < k \le K \tag{48a, b}$$

$$\mathrm{TC} = \sum_{i=1}^{I}\sum_{b=1}^{B}\sum_{k=1}^{K} c_i\,q_{ibk} + \sum_{b=1}^{B}\sum_{k=1}^{K-1}\mathrm{CB}_b[xe_{bk}] + \sum_{j=1}^{J}\sum_{k=1}^{K-1}\mathrm{CT}_j\,ue_{jk} + \sum_{o=1}^{O}\mathrm{DM}_o d_o \tag{49}$$

The revised model (RSPM) comprises Eqs. 1, 3–6, 15, 24–34, and 41–49, whose solution ensures that blending campaigns have constant blend rates that are within the limits on the blending rates and minimum run lengths at the same time. Appendix C gives the proof.

The schedule from RSPM may still show intermittent delivery of orders (Figure 3). In Figure 3, product tanks PT-101 and PT-102 deliver orders O1 and O2 intermittently. When the delivery is over contiguous slots, then this can be easily revised by simply delivering at a constant rate until the entire order, which is distributed over contiguous slots, is fully delivered. Figure 4 shows such a revised schedule for Figure 3, where deliveries of O1 and O2 are uninterrupted. Figure 5 shows the complete algorithm for the adjustment procedure.

The models and procedures discussed so far were for a single period with constant feed rates to component tanks.

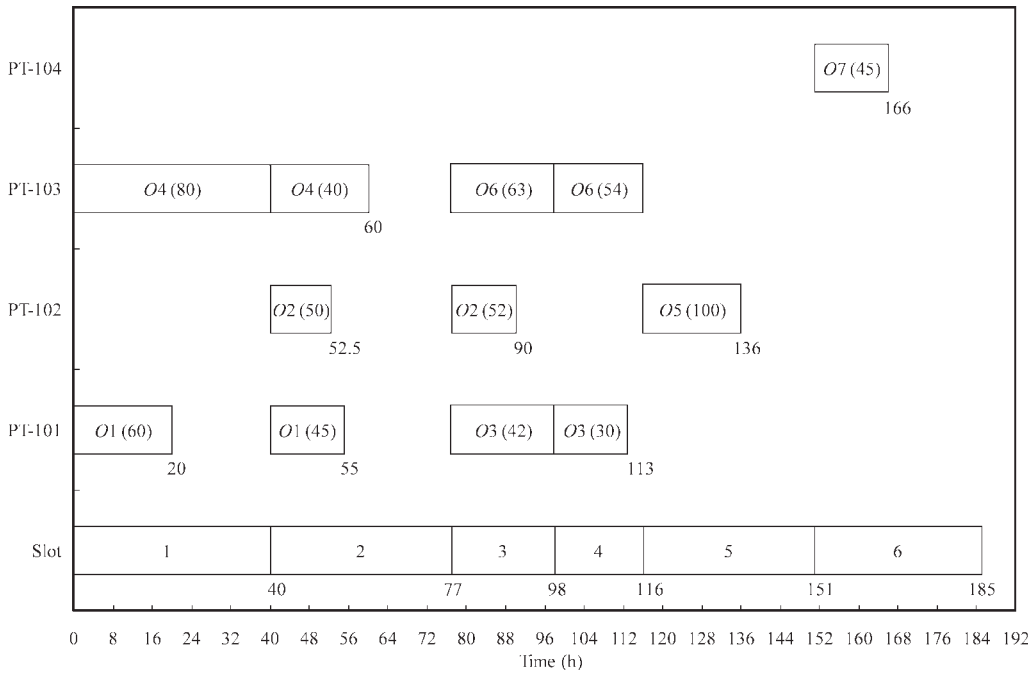

Figure 3. An example schedule to illustrate intermittent delivery of orders O1 and O2 by PT-101 and PT-120, respectively.

The extension of this to a multi-period scenario, where the entire horizon can be divided into multiple periods of constant feed rates, is straightforward as we see next.

Multi-Period Formulation (MPM)

Given the rate profiles of feeds into component tanks, we divide the entire scheduling horizon into T periods ($t = 1, 2, \ldots, T$) of lengths H_t such that the flow rates of components are constant within each period and $H = H_1 + H_2 + \ldots + H_t$. Let F_{it} be the flow rate of component i in period t. We follow the approach used by Karimi and McDonald[9] in their second model (M2). Thus, we divide each period into several slots of unknown lengths. Let $\boldsymbol{TK} = \{(t, k) \mid \text{slot } k \text{ is in period } t\}$. For this model, we first fix some T_k to be the period ends. For instance, if periods 1, 2, and 3 have three slots each, then

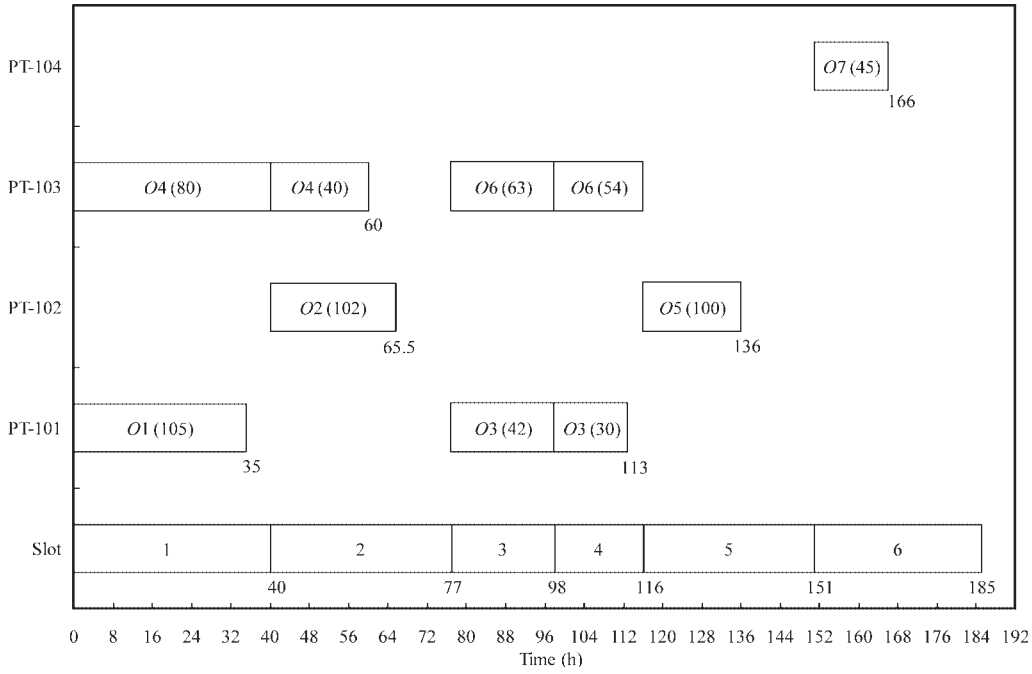

Figure 4. The schedule of Figure 3 revised by our algorithm where PT-101 and PT-102 deliver O1 and O2 continuously.

we set $T_3 = H_1$, $T_6 = H_1 + H_2$, and $T_9 = H_1 + H_2 + H_3$ with the upper bound of T_k being H. Clearly, Eq. 1 is also effective for this model (MPM), whereas Eq. 33 becomes

$$V_{ik} = V_{i(k-1)} + \sum_{i \in TK} F_{it}\mathrm{SL}_k - \sum_{b=1}^{B} q_{ibk} \qquad 0 < k \le K \quad (50a)$$

$$V_i = V_{iK} + F_{iT}(H - T_K) \quad (50b)$$

In addition, component properties may vary from period to period. For instance, refinery may often generate slops of lower quality, which can be used in blending. We assume that the component properties are known and constant during each period, but may vary with periods. Let θ_{ist} denote the known blending index for a property s of component i during period t, ρ_{it} be the density of component i during period t, and $\rho_t^{\max}$ be the maximum possible density among all products during period t. Then, Eqs. 21–22 change to

$$\sum_{i=1}^{I} q_{ibk}\theta_{ist} \ge Q_{bk}\theta_{ps}^{L} - M_b \cdot \left\{\theta_{ps}^{L} - \min_p\left(\theta_{ps}^{L}\right)\right\}\left(1 - x_{bpk}\right) \quad (b,p) \in \boldsymbol{BP}, (t,k) \in \boldsymbol{TK},\ 0 < k \le K \quad (51a)$$

$$\sum_{i=1}^{I} q_{ibk}\theta_{ist} \le Q_{bk}\theta_{ps}^{U} + M_b \cdot \left\{\min_p\left(\theta_{ps}^{U}\right) - \theta_{ps}^{U}\right\}\left(1 - x_{bpk}\right) \quad (b,p) \in \boldsymbol{BP}, (t,k) \in \boldsymbol{TK},\ 0 < k \le K \quad (51b)$$

$$\sum_{i=1}^{I} q_{ibk}\rho_{it}\theta_{ist} \ge \left(\sum_{i=1}^{I} q_{ibk}\rho_{it}\right)\theta_{ps}^{L} - M_b \cdot \left\{\theta_{ps}^{L} - \min_p\left(\theta_{ps}^{L}\right)\right\}\rho_1^{\max} \cdot \left(1 - x_{bpk}\right) \quad (b,p) \in \boldsymbol{BP}, (t,k) \in \boldsymbol{TK},\ 0 < k \le K \quad (52a)$$

$$\sum_{i=1}^{I} q_{ibk}\rho_{it}\theta_{ist} \le \left(\sum_{i=1}^{I} q_{ibk}\rho_{it}\right)\theta_{ps}^{U} + M_b \cdot \left\{\max_p\left(\theta_{ps}^{U}\right) - \theta_{ps}^{U}\right\}\rho_1^{\max} \cdot \left(1 - x_{bpk}\right) \quad (b,p) \in \boldsymbol{BP}, (t,k) \in \boldsymbol{TK},\ 0 < k \le K \quad (52b)$$

Thus, MPM comprises Eqs. 1–20, 23–32, 34–36, and 50–52.

For RMPM, Eqs. 46–47 become

$$\sum_{i=1}^{I} q_{ibk}\theta_{ist} \ge Q_{bk}\theta_{ps}^{L} - M_b\left\{\theta_{ps}^{L} - \min_p\left(\theta_{ps}^{L}\right)\right\}\left(1 - \left[x_{bpk}\right]\right) \quad \text{for}(b,k) \text{ with } [v_{b0k}] = 0, (b,p) \in \boldsymbol{BP}, (t,k) \in \boldsymbol{TK},\ 0 < k \le K \quad (53a)$$

$$\sum_{i=1}^{I} q_{ibk}\theta_{ist} \le Q_{bk}\theta_{ps}^{U} + M_b\left\{\max_p\left(\theta_{ps}^{U}\right) - \theta_{ps}^{U}\right\}\left(1 - \left[x_{bpk}\right]\right) \quad \text{for}(b,k) \text{ with } [v_{b0k}] = 0, (b,p) \in \boldsymbol{BP}, (t,k) \in \boldsymbol{TK},\ 0 < k \le K \quad (53b)$$

$$\sum_{i=1}^{I} q_{ibk}\rho_{it}\theta_{ist} \ge \left(\sum_{i=1}^{I} q_{ibk}\rho_{it}\right)\theta_{ps}^{L} - M_b\left\{\theta_{ps}^{L} - \min_p\left(\theta_{ps}^{L}\right)\right\}\rho_t^{\max}\left(1 - \left[x_{bpk}\right]\right) \quad \text{for } (b,k) \text{ with } [v_{b0k}] = 0, (b,p) \in \boldsymbol{BP}, (t,k) \in \boldsymbol{TK},\ 0 < k \le K \quad (54a)$$

$$\sum_{i=1}^{I} q_{ibk}\rho_{it}\theta_{ist} \le \left(\sum_{i=1}^{I} q_{ibk}\rho_{it}\right)\theta_{ps}^{U} + M_b\left\{\max_p\left(\theta_{ps}^{U}\right) - \theta_{ps}^{U}\right\}\rho_t^{\max}\left(1 - \left[x_{bpk}\right]\right) \quad \text{for } (b,k) \text{ with} [v_{b0k}] = 0, (b,p) \in \boldsymbol{BP}, (t,k) \in \boldsymbol{TK},\ 0 < k \le K \quad (54b)$$

Then RMPM comprises Eqs. 1, 3–6, 15, 24–32, 34, 41–45, 48–50, and 53–54.

Note that the optimal solution from SPM or MPM provides a lower bound for this MINLP problem, because it allows blending rate to vary from slot to slot in each run. Any feasible solution from RSPM or RMPM is an upper bound. If a feasible solution from RSPM or RMPM is identical to the optimal solution from SPM, then it must be globally optimal.

Next, we illustrate our solution approach in detail using a very small example (Example 1) and then evaluate our solution approach using additional thirteen larger examples.

Example 1

This example involves five orders (O1–O5), three grades (products P1–P3), nine component tanks (CT-101 to CT-109), one blender, five product tanks (PT-101 to PT-105), and one product property (Octane number). Components from atmospheric distillation, FCCU (Fluid Catalytic Cracking Unit), FCRU (Fluid Catalytic Reforming Unit), AKU (Alkanisation Unit), IFU (Isoforming Unit), CHU (Catalytic Hydrogenation Unit), ARU (Aromatization Unit), and various additives such as MTBE (Methyl Tert-Butyl Ether) and Butane are stored in the nine component tanks. The blender's operating range is [2, 15 kbbl/h] with a minimum run length of 6 h for each product. At time zero, the blender is idle. PT-101, PT-102, and PT-103 can store P2 and P3, whereas PT-104 and PT-105 can store P1 and P2. At time zero, PT-101 holds 90.20 kbbl of P3, PT-102 is empty, PT-103 holds 14.08 kbbl of P2, PT-104 holds 28.49 kbbl of P2, and PT-105 holds 20.20 kbbl of P1. Tables 2–8 list other data. The scheduling horizon is 72 h with constant flows

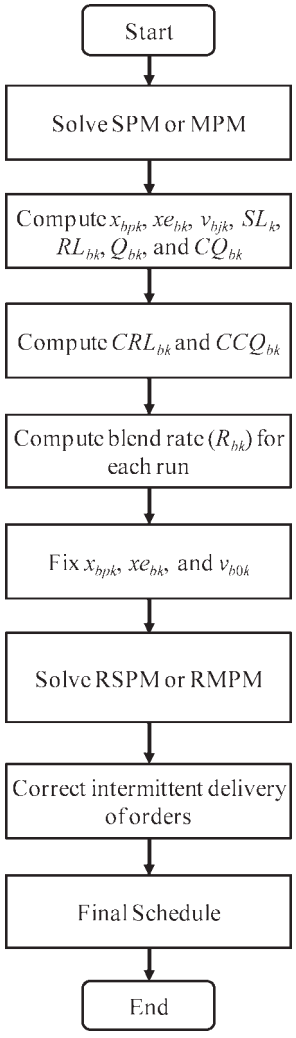

Figure 5. Flowchart for the schedule adjustment procedure.

into component tanks over that period, so our single-period model is appropriate. We solve it on a Dell precision PWS690 (Intel Xeon^R 5160 with CPU 3 GHZ and 16 GB memory) running Windows XP using CPLEX 10.0.1/GAMS 22.2.[17]

The optimal solution (Figure 6) has a cost of 5149.73 k$. The model needed 4 slots and 0.77 CPU s. P3 is not processed at all, because its initial inventory is sufficient to satisfy all orders. The blender has two runs of 24 and 34.78 h durations. The first run spans slot 1 and processes 23.43 kbbl of P2. The second run spans slots 2–4, and processes 22.05, 93.75, and 0.00 kbbl of P1, respectively. Thus, it processes 23.43 kbbl of P2 in run 1 and 115.80 kbbl of P1 in run 2. These give us the following blending rates.

$$R_{11} = \max[23.43/24, 2] = 2.0 \text{ kbbl/h}$$

$$R_{12} = R_{13} = R_{14} = \max[115.80/34.78, 2] = 3.329 \text{ kbbl/h}$$

Using the above blend rates, we solve RSPM to obtain the optimal solution of 5149.73 kbbl within 0.06 CPU s.

Table 2. Order Data for Examples 1–9

		Amount (kbbl)								Delivery Rate (kbbl/h)								Delivery Window							
o	p	1	2	3	4–5	6	7–8	9		1	2	3	4–5	6	7–8	9		1	2	3	4–5	6	7–8	9	
O1	*P1*	11	10	10	10	10	10	10		3	5	5	5	5	5	5		[0,12]	[0,24]	[0,24]	[0,24]	[0,24]	[0,24]	[0,24]	
O2	*P2*	3	3	3	3	3	3	3		3	3	3	3	3	3	3		[0,12]	[0,24]	[0,24]	[0,24]	[0,24]	[0,24]	[0,24]	
O3	*P2*	60	3	3	3	3	3	3		5	3	3	3	3	3	3		[24,48]	[24,50]	[0,24]	[0,24]	[0,24]	[0,24]	[0,24]	
O4	*P1*	125	10	10	10	10	10	10		5	5	5	5	5	5	5		[24,72]	[24,50]	[0,24]	[0,24]	[0,24]	[0,24]	[0,24]	
O5	*P2*	3	3	3	3	3	3	3		3	5	3	3	3	3	3		[24,48]	[48,72]	[24,48]	[24,48]	[24,48]	[24,48]	[24,48]	
O6	*P1*	–	10	10	10	10	10	10		–	5	5	5	5	5	5		–	[48,72]	[24,48]	[24,48]	[24,48]	[24,48]	[24,48]	
O7	*P2*	–	3	3	3	3	3	3		–	3	3	3	3	3	3		–	[48,120]	[24,48]	[24,48]	[24,48]	[24,48]	[24,48]	
O8	*P1*	–	120	100	100	90	100	100		–	5	5	5	5	5	5		–	[118,190]	[118,190]	[118,190]	[118,190]	[118,190]	[118,190]	
O9	*P2*	–	3	3	3	3	3	3		–	3	3	3	3	3	3		–	[144,168]	[144,168]	[144,168]	[144,168]	[144,168]	[144,168]	
O10	*P4*	–	150	150	150	150	150	150		–	5	5	5	5	5	5		–	[150.5,185.5]	[150.5,185.5]	[150.5,185.5]	[150.5,185.5]	[150.5,185.5]	[150.5,185.5]	
O11	*P3*	–	–	20	20	45	60	60		–	–	5	5	5	5	5		–	–	[144,168]	[144,168]	[144,168]	[144,168]	[144,168]	
O12	*P2*	–	–	30	30	30	20	20		–	–	5	5	5	5	5		–	–	[24,48]	[24,48]	[24,48]	[24,48]	[24,48]	
O13	*P4*	–	–	–	60	45	60	60		–	–	–	5	5	5	5		–	–	–	[0,56]	[0,56]	[0,56]	[0,56]	
O14	*P3*	–	–	–	10	15	15	20		–	–	–	5	5	5	5		–	–	–	[48,72]	[48,72]	[48,72]	[48,72]	
O15	*P2*	–	–	–	20	15	20	20		–	–	–	4	4	4	4		–	–	–	[0,72]	[0,75]	[0,72]	[0,72]	
O16	*P2*	–	–	–	–	10	20	20		–	–	–	–	4	5	5		–	–	–	–	[48,72]	[48,72]	[48,72]	
O17	*P1*	–	–	–	–	10	10	10		–	–	–	–	5	5	5		–	–	–	–	[48,96]	[48,72]	[48,72]	
O18	*P1*	–	–	–	–	10	10	10		–	–	–	–	5	5	5		–	–	–	–	[48,96]	[48,72]	[48,72]	
O19	*P2*	–	–	–	–	–	60	60		–	–	–	–	–	5	5		–	–	–	–	–	[0,50]	[0,50]	
O20	*P2*	–	–	–	–	–	40	40		–	–	–	–	–	5	5		–	–	–	–	–	[144,168]	[144,168]	
O21	*P1*	–	–	–	–	–	–	30		–	–	–	–	–	–	5		–	–	–	–	–	–	[96,120]	
O22	*P5*	–	–	–	–	–	–	40		–	–	–	–	–	–	5		–	–	–	–	–	–	[144,168]	
O23	*P3*	–	–	–	–	–	–	20		–	–	–	–	–	–	5		–	–	–	–	–	–	[144,168]	

Table 3. Order Data for Examples 10–14

	Product		Amount (kbbl)					Delivery Rate (kbbl/h)					Delivery Window				
o	12	The Rest	10	11	12	13	14	10	11	12	13	14	10	11	12	13	14
$O1$	$P1$	$P1$	10	10	10	10	10	5	5	5	5	5	[0,24]	[0,24]	[0,24]	[0,24]	[0,24]
$O2$	$P2$	$P2$	3	3	3	3	3	3	3	3	3	3	[0,24]	[0,24]	[0,24]	[0,24]	[0,24]
$O3$	$P2$	$P2$	3	3	3	3	3	3	3	3	3	3	[0,24]	[0,24]	[0,24]	[0,24]	[0,24]
$O4$	$P1$	$P1$	10	10	10	10	10	5	5	5	5	5	[0,24]	[0,24]	[0,24]	[0,24]	[0,24]
$O5$	$P2$	$P2$	3	3	3	3	3	3	3	3	3	3	[24,48]	[24,48]	[24,48]	[24,48]	[24,48]
$O6$	$P1$	$P1$	10	10	10	10	10	5	5	5	5	5	[24,48]	[24,48]	[24,48]	[24,48]	[24,48]
$O7$	$P2$	$P2$	3	3	3	3	3	3	3	3	3	3	[24,48]	[24,48]	[24,48]	[24,48]	[24,48]
$O8$	$P1$	$P1$	100	100	100	100	100	5	5	5	5	5	[118,190]	[118,190]	[118,190]	[118,190]	[118,190]
$O9$	$P2$	$P2$	3	3	3	3	3	3	3	3	3	3	[144,168]	[144,168]	[144,168]	[144,168]	[144,168]
$O10$	$P4$	$P4$	150	150	100	150	150	5	5	5	5	5	[150.5,185.5]	[150.5,185.5]	[150.5,185.5]	[150.5,185.5]	[150.5,185.5]
$O11$	$P3$	$P3$	60	60	60	60	60	5	5	5	5	5	[144,168]	[144,168]	[144,168]	[144,168]	[144,168]
$O12$	$P2$	$P2$	20	20	20	20	20	5	5	5	5	5	[24,48]	[24,48]	[24,48]	[24,48]	[24,48]
$O13$	$P4$	$P4$	60	60	60	60	60	5	5	5	5	5	[0,56]	[0,56]	[0,56]	[0,56]	[0,56]
$O14$	$P3$	$P3$	20	20	15	20	20	5	5	5	5	5	[48,72]	[48,72]	[48,72]	[48,72]	[48,72]
$O15$	$P2$	$P2$	20	20	20	20	20	4	4	4	4	4	[0,72]	[0,72]	[0,72]	[0,72]	[0,72]
$O16$	$P2$	$P2$	20	20	20	20	20	5	5	5	5	5	[48,72]	[48,72]	[48,72]	[48,72]	[48,72]
$O17$	$P1$	$P1$	10	10	10	10	10	5	5	5	5	5	[48,72]	[48,72]	[48,72]	[48,72]	[48,72]
$O18$	$P1$	$P1$	10	10	10	10	10	5	5	5	5	5	[48,72]	[48,72]	[48,72]	[48,72]	[48,72]
$O19$	$P2$	$P2$	60	60	60	60	60	5	5	5	5	5	[0,50]	[0,50]	[0,50]	[0,50]	[0,50]
$O20$	$P2$	$P2$	40	40	40	40	40	5	5	5	5	5	[144,168]	[144,168]	[144,168]	[144,168]	[144,168]
$O21$	$P5$	$P1$	30	30	30	30	30	5	5	5	5	5	[96,120]	[96,120]	[96,120]	[96,120]	[96,120]
$O22$	$P5$	$P5$	40	40	40	40	40	5	5	5	5	5	[144,168]	[144,168]	[144,168]	[144,168]	[144,168]
$O23$	$P3$	$P3$	20	20	20	20	20	5	5	5	5	5	[144,168]	[144,168]	[144,168]	[144,168]	[144,168]
$O24$	$P5$	$P5$	6	6	6	6	6	3	3	3	3	3	[96,120]	[96,120]	[96,120]	[96,120]	[96,120]
$O25$	$P5$	$P5$	20	20	20	20	20	5	5	5	5	5	[144,168]	[144,168]	[144,168]	[144,168]	[144,168]
$O26$	$P3$	$P1$	–	10	30	10	10	–	4	4	4	4	–	[0,76]	[144,168]	[0,76]	[0,76]
$O27$	$P3$	$P4$	–	20	20	20	20	–	5	4	5	5	–	[120,144]	[72,96]	[120,144]	[120,144]
$O28$	$P4$	$P1$	–	25	3	25	25	–	5	3	5	5	–	[120,144]	[72,96]	[120,144]	[120,144]
$O29$	$P4$	$P5$	–	10	15	10	10	–	5	3	5	5	–	[120,144]	[96,120]	[120,144]	[120,144]
$O30$	$P1$	$P4$	–	15	15	15	15	–	5	3	5	5	–	[120,144]	[96,120]	[120,144]	[120,144]
$O31$	$P2$	$P1$	–	–	15	15	15	–	–	5	5	5	–	–	[96,120]	[120,144]	[120,144]
$O32$	$P5$	$P1$	–	–	20	20	20	–	–	2	5	5	–	–	[96,120]	[144,168]	[144,168]
$O33$	$P1$	$P4$	–	–	20	20	20	–	–	5	5	5	–	–	[0,76]	[144,168]	[144,168]
$O34$	$P3$	$P4$	–	–	20	20	20	–	–	5	5	5	–	–	[120,144]	[168,192]	[168,192]
$O35$	$P3$	$P5$	–	–	30	30	30	–	–	5	5	5	–	–	[120,144]	[168,192]	[168,192]
$O36$	–	$P2$	–	–	–	3	3	–	–	–	3	3	–	–	–	[168,192]	[168,192]
$O37$	–	$P1$	–	–	–	10	10	–	–	–	5	5	–	–	–	[168,192]	[168,192]
$O38$	–	$P1$	–	–	–	40	40	–	–	–	5	5	–	–	–	[168,192]	[168,192]
$O39$	–	$P4$	–	–	–	10	10	–	–	–	5	5	–	–	–	[168,192]	[168,192]
$O40$	–	$P5$	–	–	–	10	10	–	–	–	5	5	–	–	–	[168,192]	[168,192]
$O41$	–	$P1$	–	–	–	–	15	–	–	–	–	5	–	–	–	–	[168,192]
$O42$	–	$P2$	–	–	–	–	20	–	–	–	–	3	–	–	–	–	[168,192]
$O43$	–	$P3$	–	–	–	–	15	–	–	–	–	5	–	–	–	–	[144,168]
$O44$	–	$P5$	–	–	–	–	20	–	–	–	–	4	–	–	–	–	[168,192]
$O45$	–	$P4$	–	–	–	–	10	–	–	–	–	5	–	–	–	–	[96,120]

Figure 7 shows the optimal schedule from RSPM. Because the solutions from RSPM and SPM have the same cost, we have a guaranteed global optimal solution. Each order is delivered before its due date, so no demurrage incurs in this example. The blender has one product transition (P2-P1) after slot 1, and PT-104 transitions from P2 to P1 after slot 2. PT-104 delivers O3 and O5 simultaneously during slot 2, but there are no intermittent deliveries. Thus, Figure 7 is a realistic optimal schedule for this example.

Now, we consider two periods of 40 and 32 h in the scheduling horizon. Figure 8 shows the feed rate profiles to component tanks. The feed rates to CT-101 through CT-109 in the first period are 1.2, 0.8, 1.2, 1.2, 0.5, 0.8, 0.0, 0.0, and 1.0 kbbl/h, respectively. In the second period, they are 0.8, 0.6, 0.6, 0.8, 0.5, 0.6, 0.5, 0.5, and 0.0 kbbl/h, respectively. We use three slots for the first period and two slots for the second. The optimal solution remains the same as 5149.73 k$, but the solution time increases to 12.6 CPU s due to the presence of one more slot in MPM.

Detailed evaluation

Tables 2–8 show the data for Examples 2–14 that we use for a detailed evaluation of our methodology. We use nine components, eleven product tanks, and 192-h (8-day) scheduling horizon. Examples 2–7 have one blender, Examples 8–12 have two, and Examples 13 and 14 have three blenders. We use 10–45 orders in these examples as shown in Tables 2–8. We have designed our test examples with widely varying structure, size, scale, and complexity, and they mimic the real-life industrial scenarios very well. For all examples except Example 5, all blenders are idle at time zero. In

Table 4. Product and Component tank Data for Examples 1–14

Tank	Initial Contain		Initial Stock (kbbl)		Capacity (kbbl)		Storable Products				Maximum Delivery Rate/Feed Rate (kbbl/h)						
	1	2–14	1	2–14	1	2–14	1	2–8	9–11	13–14	12	1	2–5	6	7–8	9–10	11	12	13–14
PT-101	*P3*	*P3*	90.20	30.00	100	150	*P2, P3*	*P2, P3*	*P2, P3, P5*	*P2, P3, P5*	*P2, P3, P5*	15	20	25	20	30	30	30	30
PT-102	*P3*	*P3*	0.00	0.00	100	150	*P2, P3*	*P2, P3*	*P2, P3, P5*	*P2, P3, P5*	*P2, P3, P5*	15	20	25	20	30	30	30	30
PT-103	*P2*	*P2*	14.08	14.08	150	150	*P2, P3*	*P2, P3*	*P2, P3, P5*	*P2, P3, P5*	*P2, P3, P5*	15	20	25	20	30	30	30	30
PT-104	*P2*	*P4*	28.49	25.00	100	200	*P1, P2*	*P2, P4*	*P2, P4*	*P2, P4*	*P2, P5*	15	20	25	20	30	30	30	30
PT-105	*P1*	*P2*	20.20	28.49	100	200	*P1, P2*	*P2, P3*	*P2, P5*	*P2, P5*	*P2, P3, P5*	15	20	25	20	30	30	30	30
PT-106	–	*P2*	–	57.59	–	150	–	*P2, P3*	*P2, P5*	*P2, P5*	*P2, P3, P5*	–	20	25	20	30	30	30	30
PT-107	–	*P1*	–	13.79	–	200	–	*P1, P4*	*P1, P4*	*P1, P4*	*P1, P4*	–	20	25	20	30	30	30	30
PT-108	–	*P1*	–	12.36	–	150	–	*P1, P4*	*P1, P4*	*P1, P4*	*P1, P4*	–	20	25	20	30	30	30	30
PT-109	–	*P4*	–	23.96	–	200	–	*P1, P4*	*P1, P4*	*P1, P4*	*P1, P4*	–	20	25	20	30	30	30	30
PT-110	–	*P1*	–	60.00	–	150	–	*P1, P4*	*P1, P4*	*P1, P4*	*P1, P4*	–	20	25	20	30	30	30	30
PT-111	–	*P1*	–	12.36	–	150	–	*P1, P4*	*P1, P4*	*P1, P4*	*P1, P4*	–	20	25	20	30	30	30	30
CT-101	*C1*	*C1*	27.38	26.46	200	250	–	–	–	–	–	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
CT-102	*C2*	*C2*	34.58	67.90	200	300	–	–	–	–	–	1.00	0.50	0.50	0.50	0.50	0.50	0.50	0.50
CT-103	*C3*	*C3*	59.44	59.44	250	300	–	–	–	–	–	1.20	1.00	1.00	1.00	1.00	1.00	1.00	1.00
CT-104	*C4*	*C4*	28.29	44.44	250	300	–	–	–	–	–	1.20	1.00	1.00	1.00	1.00	1.00	1.00	1.00
CT-105	*C5*	*C5*	10.59	10.59	200	200	–	–	–	–	–	1.00	0.80	0.80	0.80	0.80	0.70	0.80	0.70
CT-106	*C6*	*C6*	19.53	19.53	200	250	–	–	–	–	–	1.00	0.50	0.50	0.50	0.50	0.50	0.50	0.50
CT-107	*C7*	*C7*	27.30	46.91	100	250	–	–	–	–	–	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.50
CT-108	*C8*	*C8*	49.34	49.47	100	250	–	–	–	–	–	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.50
CT-109	*C9*	*C9*	13.84	44.58	200	250	–	–	–	–	–	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Example 5, the blender is processing P1 before time zero. Although we list several gasoline properties in Table 1, we consider only nine of them in our examples. These are ON, RVP, S, B, A, O, SG, FL, and OX. Our model can easily incorporate other gasoline properties. We solve all examples on a Dell precision PWS690 (Intel Xeon^R CPU 3 GHz, 16 GB RAM) running Windows XP using solver CPLEX 10.0.1/GAMS22.2.[17]

Tables 9 and 10 give the computational performance of SPM and MPM. Our procedure obtains guaranteed global optimal solutions for Examples 1–5, because both RSPM and SPM give the same solution. We were unable to get optimal solutions or prove their optimality for Example 6–14, so we set upper limits on CPU times for these examples as shown in Tables 9 and 10. For instance, we use 3 h (10,800 s) for SPM and 1 h (3600 s) for RSPM of Examples 6–10. As we can see from Tables 9 and 10, solutions are typically obtained quite quickly, but proving optimality takes much time. For instance, SPM gives a solution of 5213.88 k$ for Example 6 in just 591 CPU s, but spends the remaining 2.84 h to prove optimality. In fact, it fails to prove optimality even after 3 days of CPU time.

Table 11 gives the RMIP values and best solutions from SPM for Examples 1–14. The best solutions, which are the better lower bounds than the RMIP values, do not improve much over time for Examples 7–14. For instance, SPM gives an RMIP and best solution of 7797.02 k$ for Example 7, but fails to prove optimality even after 3 h. Similarly, the RMIP value from SPM for Example 9 is 10,003.92 k$. After 3 h, the best solution is 10,013.90 k$, which is an improvement of just 0.10%. The lower bounds for SPM and MPM improve very slowly for large problems. This makes it difficult to solve them to optimality. A possible reason for this increased complexity is the transitions in the product tanks. They result in very low RMIP values, which are difficult to improve. To illustrate this point, consider Example 12 with 35 orders. This example allows some product tanks to hold one more product compared with other examples. For instance, PT-105 can hold P2, P3, and P5 in Example 12, whereas only two products in other examples. This example needs 33 h of CPU time to obtain solutions with a relative gap of 8.55%. This suggests that examples with dedicated product tanks should probably run much faster.

RSPM improves the qualities of SPM solutions by 1.2% on an average for Examples 9, 12, 13, and 14. For instance, SPM gives 16,536.00 k$ for Example 13, whereas RSPM gives 16,146.52 k$. However, it does not do so for Examples 10 and 11. For instance, SPM gives 11,543.60 k$ for Example 10, but RSPM gives 11,629.09 k$.

Our observations from Table 10 are similar to those from Table 9. For illustration, we give the final schedules for Examples 5 (15 orders) and 14 (45 orders) in Figures 9 and 10, respectively. The following features of these schedules are noteworthy.

1) In Figure 9, although the blender has processed 150 kbbl of P1 at time zero for 10 h, it still continues with P1 for another 43.14 h during slot 1.

2) There is no demurrage in Figure 9. Figure 10 has demurrage. For instance, O5 is delivered in slot 2, and incurs a delay of 0.33 h compared with its due date of 48 h.

Table 5. Component and Product Property Indices for Examples 1–14

RBN			RVI		SULI		BI		AROI	
1	2–8	9–14	2–8	9–14	2–8	9–14	2–8	9–14	2–8	9–14
86.5000	86.5000	86.5000	140.4650	140.4650	80.0000	80.0000	0.7800	0.7800	25.0000	25.0000
103.6600	103.6600	103.6600	68.9213	68.9213	40.0000	40.0000	0.9800	0.9800	31.7000	31.7000
111.3500	111.3500	111.3500	87.6804	87.6804	0.0000	0.0000	1.2000	1.2000	48.0000	48.0000
113.9300	113.9300	113.9300	51.4659	51.4659	5.0000	5.0000	0.0000	0.0000	0.0000	0.0000
94.5000	94.5000	94.5000	175.5886	175.5886	0.0000	0.0000	0.0960	0.0960	0.0000	0.0000
118.1600	118.1600	118.1600	19.9115	19.9115	0.0800	0.0800	0.0050	0.0050	0.0000	0.0000
144.6800	144.6800	144.6800	12.5522	12.5522	7.5000	7.5000	0.0078	0.0078	0.0500	0.0500
150.6600	150.6600	150.6600	110.5925	110.5925	2.0000	2.0000	0.2500	0.2500	19.2000	19.2000
92.5000	92.5000	92.5000	436.3383	436.3383	30.0000	30.0000	0.0920	0.0920	24.0000	24.0000
[110.45, $+\infty$]	[110.45, $+\infty$]	[110.45, $+\infty$]	[15, 170]	[15, 170]	[0, 45]	[0, 45]	[0, 0.86]	[0, 0.86]	[0, 35.00]	[0, 35.00]
[111.95, $+\infty$]	[111.95, $+\infty$]	[111.95, $+\infty$]	[15, 170]	[15, 170]	[0, 50]	[0, 50]	[0, 0.92]	[0, 0.92]	[0, 36.00]	[0, 36.00]
[108.97, $+\infty$]	[108.97, $+\infty$]	[108.97, $+\infty$]	[15, 170]	[15, 170]	[0, 44]	[0, 44]	[0, 0.94]	[0, 0.94]	[0, 42.00]	[0, 42.00]
–	[103.24, $+\infty$]	[103.24, $+\infty$]	[15, 170]	[15, 170]	[0, 50]	[0, 50]	[0, 0.90]	[0, 0.90]	[0, 40.00]	[0, 40.00]
–	–	[115.01, $+\infty$]	–	[15, 170]	–	[0, 48]	–	[0, 0.93]	–	0, 40.00]
	OI			DNI		FLI			OXI	
4	5–8	9–14	4–8	9–14	4–8	9–14	4–5	7–8	9–11	12–14
1.0000	1.0000	1.0000	1.4850	1.4850	3.4500	3.4500	0.2500	0.2500	0.2500	0.2500
23.8000	23.8000	23.8000	1.3340	1.3340	6.2500	6.2500	0.7500	0.7500	0.7500	0.7500
0.8500	0.8500	0.8500	1.2200	1.2200	2.3600	2.3600	2.0000	2.0000	2.0000	2.0000
0.0000	0.0000	0.0000	1.5800	1.5800	3.5600	3.5600	1.2500	1.2500	1.2500	1.2500
0.4000	0.4000	0.4000	1.4980	1.4980	1.9600	1.9600	0.0800	0.0800	0.0800	0.0800
0.7200	0.7200	0.7200	1.4360	1.4360	3.6500	3.6500	0.0000	0.0000	0.0000	0.0000
0.00038	0.00038	0.00038	1.1500	1.1500	2.9600	2.9600	0.0005	0.0005	0.0005	0.0005
0.1500	0.1500	0.1500	1.3480	1.3480	5.4600	5.4600	18.2000	18.2000	18.2000	18.2000
0.0600	0.0600	0.0600	1.6050	1.6050	7.9500	7.9500	0.8500	0.8500	0.8500	0.8500
[0, 20.00]	[0, 20.00]	[0, 20.00]	[1.190, 1.667]	[1.190, 1.667]	[1.4, 7.60]	[1.4, 7.60]	[0, 1.85]	[0, 2.80]	[0, 2.80]	[0, 2.80]
[0, 18.00]	[0, 18.00]	[0, 18.00]	[1.199, 1.667]	[1.199, 1.667]	1.4, 7.25]	[1.4, 7.25]	[0, 1.90]	[0, 2.75]	[0, 2.75]	[0, 2.75]
[0, 20.00]	[0, 20.00]	[0, 20.00]	[1.182, 1.667]	[1.182, 1.667]	[1.4, 7.20]	[1.4, 7.20]	[0, 2.10]	[0, 2.90]	[0, 2.90]	[0, 2.90]
[0, 18.00]	[0, 18.00]	[0, 18.00]	[1.190, 1.667]	[1.190, 1.667]	[1.4, 7.50]	[1.4, 7.50]	[0, 2.00]	[0, 2.70]	[0, 2.70]	[0, 2.70]
–	–	[0, 20.00]	–	[200, 1.667]	–	[1.4, 7.40]	–	–	[0, 3.00]	[0, 3.00]

Table 6. Allowable Composition Ranges for Components in Products of Examples 1–14

	*C*1			*C*2			*C*3			*C*4					
Product	1	2–8	9–14	1	2–8	9–14	1	2–8	9–14	1	2–4	5–7	8	9–12	13–14
*P*1	[0, 0.22]	[0, 0.22]	[0, 0.22]	[0.1, 1]	[0.10, 1]	[0.10, 1]	[0, 1]	[0, 1]	[0, 1]	[0, 0.40]	[0, 0.40]	[0, 0.40]	[0, 0.40]	[0, 0.40]	[0, 0.40]
*P*2	[0, 0.24]	[0, 0.24]	[0, 0.24]	[0.1, 1]	[0.10, 1]	[0.10, 1]	[0, 1]	[0, 1]	[0, 1]	[0, 0.45]	[0, 0.45]	[0, 0.45]	[0, 0.45]	[0, 0.45]	[0, 0.45]
*P*3	[0, 0.25]	[0, 0.25]	[0, 0.25]	[0.1, 1]	[0.10, 1]	[0.10, 1]	[0, 1]	[0, 1]	[0, 1]	[0, 0.43]	[0, 0.43]	[0, 0.43]	[0, 0.43]	[0, 0.43]	[0, 0.43]
*P*4	–	[0, 0.24]	[0, 0.24]	–	[0.10, 1]	[0.10, 1]	–	[0, 1]	[0, 1]	–	[0, 0.44]	[0, 0.44]	[0, 0.44]	[0, 0.44]	[0, 0.44]
*P*5	–	–	[0, 0.30]	–	–	[0.15, 1]	–	–	[0, 1]	–	–	–	–	[0, 0.40]	[0, 0.40]

	*C*5			*C*6			*C*7			*C*8			*C*9		
Product	1	2–8	9–14	1	2–8	9–14	1	2–8	9–14	1	2–8	9–14	1	2–8	9–14
*P*1	[0, 0.25]	[0, 0.25]	[0, 0.25]	[0, 0.20]	[0, 0.20]	[0, 0.20]	[0, 0.25]	[0, 0.25]	[0, 0.25]	[0, 0.30]	[0, 6.30]	[0, 0.30]	[0, 0.15]	[0, 0.15]	[0, 0.15]
*P*2	[0, 0.25]	[0, 0.25]	[0, 0.25]	[0, 0.22]	[0, 0.22]	[0, 0.22]	[0, 0.25]	[0, 0.25]	[0, 0.25]	[0, 0.30]	[0, 0.30]	[0, 0.30]	[0, 0.18]	[0, 0.18]	[0, 0.18]
*P*3	[0, 0.25]	[0, 0.25]	[0, 0.25]	[0, 0.18]	[0, 0.18]	[0, 0.18]	[0, 0.25]	[0, 0.25]	[0, 0.25]	[0, 0.30]	[0, 0.30]	[0, 0.30]	[0, 0.20]	[0, 0.20]	[0, 0.20]
*P*4	–	[0, 0.25]	[0, 0.25]	–	[0, 0.20]	[0, 0.20]	–	[0, 0.25]	[0, 0.25]	–	[0, 0.30]	[0, 0.30]	–	[0, 0.16]	[0, 0.16]
*P*5	–	–	[0, 0.25]	–	–	[0, 0.20]	–	–	[0, 0.25]	–	–	[0, 0.30]	–	–	[0, 0.17]

Table 7. Blender and Economic Data for Examples 1–14

	RL_{bo}h and CQ_{bo}, kbbl					Minimum and Maximum Blending Rate (kbbl/h)							Allowable Product				
Blender	1–4	5	6–7	8–12	13–14	1	2	3–6	7	8–9	10–12	13–14	1	2–7	8	9–12	13–14
B-101	0 and 0	10 and 150	0 and 0	0 and 0	0 and 0	2.0–15	1.5–15	1.5–20	1.5–25	1.5–25	1.5–30	1.5–30	*P*1–*P*3	*P*1–*P*4	*P*1–*P*4	*P*1–*P*5	*P*1–*P*5
B-102	–	–	–	0 and 0	0 and 0	–	–	–	–	1.5–25	1.5–30	1.5–30	–	–	*P*1–*P*4	*P*1–*P*5	*P*1–*P*5
B-103	–	–	–	–	0 and 0	–	–	–	–	–	–	1.5–25	–	–	–	–	*P*1–*P*5

Minimum Run Length of Each Product in Blender (h)

	*P*1			*P*2				*P*3			*P*4				*P*5		
	1–7	8–12	13–14	1–7	8–12	13	14	1–7	8–12	13–14	2–7	8–11	12	13–14	9–11	12	13–14
B–101	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	5	5
B-102	–	6	6	–	6	6	6	–	6	6	–	6	6	6	6	5	5
B-103	–	–	6	–	–	6	6	–	–	6	–	–	–	6	–	–	5

	Ex	*C*1	*C*2	*C*3	*C*4	*C*5	*C*6	*C*7	*C*8	*C*9	Ex	Demurrage Cost(k$/h)	Scheduling Horizon (h)
Cost ($/bbl)	1	20	40	55	45	25	50	58	60	30	1	2.1	72
	2–14	20	24	30	25	22	27	50	50	22.5	2–14	2.5	192

Transition	Ex	*PT*-101	*PT*-102	*PT*-103	*PT*-104	*PT*-105	*PT*-106	*PT*-107	*PT*-108	*PT*-109	*PT*-110	*PT*-111	Transition Cost in blender (k$/instance)
Cost (k$/instance)	1	9.8	9.8	14.5	9.8	9.8	–	–	–	–	–	–	20
2–14	14.5	14.5	14.5	19	19	14.5	19	14.5	19	14.5	14.5		

Table 8. Periods Slots and Fee Flow Rates to Component Tanks for Examples 1–14

Example	Period	Period Duration	Slot	Feed Flow Rate to Component Tank (kbbl/h)								
				CT–101	*CT*–102	*CT*–103	*CT*–104	*CT*–105	*CT*–106	*CT*–107	*CT*–108	*CT*–109
1	1	40	1–3	1.2	0.8	1.2	1.2	0.5	0.8	0.0	0.0	1.0
	2	32	4–5	0.8	0.6	0.6	0.8	0.5	0.6	0.5	0.5	0.0
2	1	60	1–2	1.2	0.8	1.2	1.2	0.5	0.8	0.0	0.0	1.0
	2	132	3–5	0.8	0.6	0.6	0.8	0.5	0.6	0.5	0.5	0.0
3–5	1	100	1–3	1.2	0.8	1.2	1.2	0.5	0.8	0.0	0.0	1.0
	2	92	4–5	0.8	0.6	0.6	0.8	0.5	0.6	0.5	0.5	0.0
6	1	120	1–4	1.2	0.8	1.2	1.2	0.7	0.8	0.0	0.0	1.0
	2	72	5–6	0.8	0.6	0.6	0.8	0.5	0.6	0.5	0.5	0.0
7	1	80	1–3	1.2	0.8	1.2	1.2	0.7	0.8	0.0	0.0	1.0
	2	70	4–6	0.8	0.6	0.6	0.8	0.5	0.6	0.5	0.5	0.0
	3	42	7–9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	1	80	1–3	1.2	0.8	1.2	1.2	0.5	0.8	0.0	0.0	1.0
	2	70	4–6	0.8	0.6	0.6	0.8	0.5	0.6	0.5	0.5	0.0
	3	42	7–8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	1	80	1–3	1.0	0.5	1.0	1.0	0.5	0.5	0.0	0.0	1.0
	2	70	4–7	0.8	0.6	0.6	0.8	0.5	0.6	0.5	0.5	0.0
	3	42	8–9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10	1	80	1–3	1.0	0.5	1.0	1.0	0.8	0.5	0.0	0.0	1.0
	2	60	4–7	0.8	0.6	0.6	0.8	0.5	0.6	0.5	0.5	0.0
	3	52	8–10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11	1	80	1–4	1.0	0.5	1.0	1.0	0.7	0.5	0.0	0.0	1.0
	2	60	5–9	0.8	0.6	0.6	0.8	0.5	0.6	0.5	0.5	0.0
	3	52	10–14	0.5	0.5	0.5	0.5	0.5	0.5	0.0	0.0	0.5
12	1	50	1–5	1.0	0.5	1.0	1.0	0.8	0.5	0.0	0.0	1.0
	2	50	6–0	0.8	0.6	0.6	0.8	0.5	0.6	0.5	0.5	0.0
	3	50	11–14	0.5	0.5	0.5	0.5	0.5	0.5	0.0	0.0	0.5
	4	42	15–17	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
13	1	50	1–5	1.0	0.5	1.0	1.0	0.7	0.5	0.5	0.5	1.0
	2	50	6–0	0.8	0.6	0.6	0.8	0.5	0.6	0.5	0.5	0.0
	3	50	11–15	0.5	0.5	0.5	0.5	0.5	0.5	0.0	0.0	0.5
	4	42	16–18	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
14	1	50	1–5	1.0	0.5	1.0	1.0	0.7	0.5	0.5	0.5	1.0
	2	50	6–10	0.8	0.6	0.6	0.8	0.5	0.6	0.5	0.5	0.0
	3	50	11–14	0.5	0.5	0.5	0.5	0.5	0.5	0.0	0.0	0.5
	4	42	15–17	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

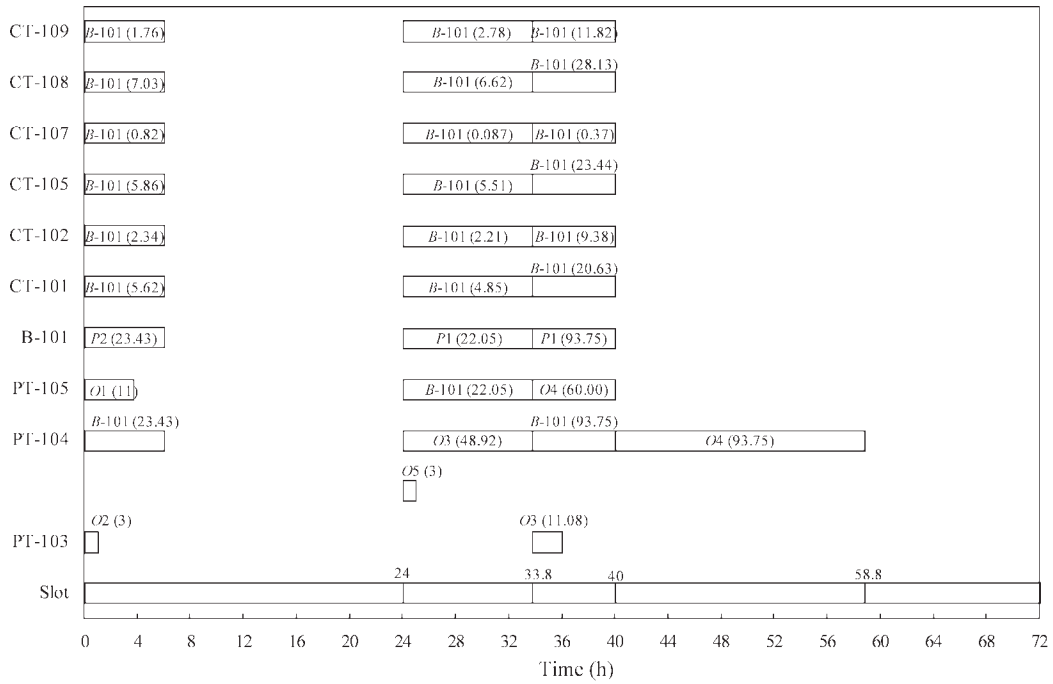

Figure 6. Optimal schedule for Example 1 (5 orders) from SPM.

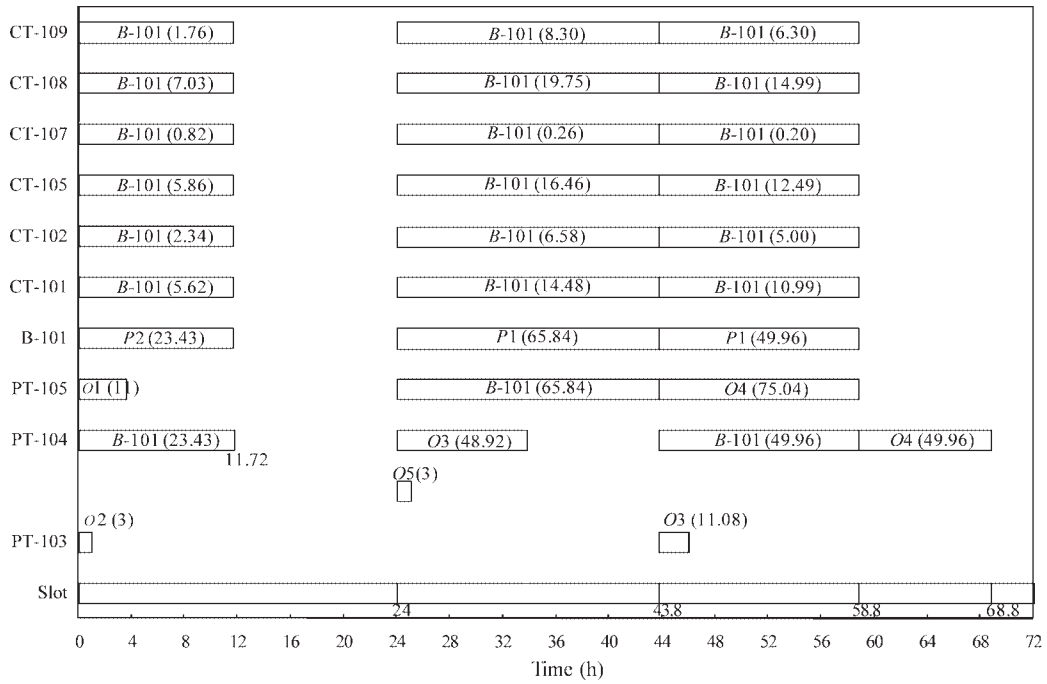

Figure 7. Optimal schedule for Example 1 (5 orders) from RSPM.

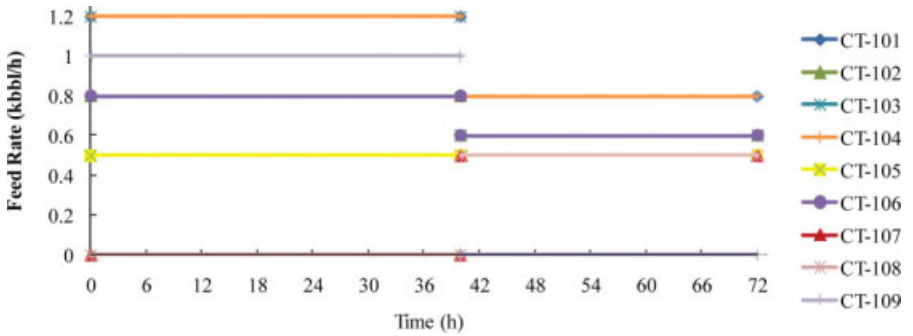

Figure 8. Feed rate profiles of blend components from component tanks for Example 1.

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

Table 9. Computational Performance of SPM

Example	Order	Discrete Continuous Slot	Discrete Variables	Continuous Variables	Constraints	Non-Zero Elements	CPU Time for SPM (s)	CPU Time for RSPM (s)	Cost from SPM (k$)	Cost from RSPM (k$)
1	5	4	100	269	1098	3094	0.77	0.06	5149.73	5149.73
2	10	4	252	598	2847	9037	48.3	0.28	3658.11	3658.11
3	12	4	315	633	3199	10074	177	0.14	3159.12	3159.12
4	15	4	372	723	3855	12871	641	3.50	4556.67	4556.67
5	15	4	372	723	3867	12891	215	0.53	4556.67	4556.67
6	18	5	576	950	5478	18068	591*	0.31	5213.88	5213.88
7	20	6	775	1197	7175	23,454	10,800†	2.31	8100.35	8100.35
8	20	5	682	1164	7664	25,990	10,800†	4.81	8100.35	8100.35
9	23	8	1251	1890	13,922	48,082	10,800†	3600†	10,803.23	10,750.49
10	25	9	1503	2209	16,492	56,490	10,800†	3600†	11,543.60	11,629.09
11	30	12	2328	3216	24,755	83,253	36,000†	4065	13,444.14	13,476.48
12	35	16	3846	4963	39,520	129,757	108,000†	10,800†	15,053.85	15,016.66
13	40	17	4414	5966	51,775	171,916	108,000†	10,800†	16,536.00	16,146.52
14	45	16	4532	6077	52,785	173,176	108,000†	10,800†	18,547.36	18,233.61

CPU time limit for SPM is set at 10,800 s for Examples 1–10, 36,000 s for Example 11, and 108,000 s for Examples 12–14.
CPU time limit for RSPM is set at 3600 s for Examples 1–10, and 10,800 s for Examples 11–14.
*The time when final solution is found, but termination at CPU time limit.
†Reached CPU time limit.

Table 10. Computational Performance of MPM

Example	Order	Discrete Variables	Continuous Variables	Constraints	Non-Zero Elements	CPU Time for MPM (s)	CPU Time for RMPM (s)	Cost from MPM (k$)	Cost from RMPM (k$)
1	5	130	328	1384	3912	12.5	0.14	5149.73	5149.73
2	10	329	729	3600	11,423	137	0.16	3658.11	3658.11
3	12	411	769	4048	12,736	82.6	0.14	3179.12	3179.12
4	15	486	877	4880	16,255	495	0.16	4556.67	4556.67
5	15	486	877	4892	16,279	64.8	0.39	4556.67	4556.67
6	18	712	1114	6631	21,841	202*	0.37	5213.88	5213.88
7	20	1219	1725	10,922	35,585	10800†	0.89	8100.35	8100.35
8	20	1159	1779	12,464	42,115	10800†	33.6	8100.35	8114.85
9	23	1423	2107	15,708	54,201	10800†	10.8	10613.65	10613.65
10	25	1867	2663	20,244	69,251	10800†	3600†	11432.21	11488.35
11	30	2328	3216	24,755	83,258	36000†	10800†	13347.63	13622.18
12	35	4099	5258	42,026	137,904	108000†	597	15374.24	15321.74
13	40	4686	6302	54,857	182,076	108000†	10800†	16323.84	16653.07
14	45	4830	6439	56,127	184,066	108000†	256.3	18778.33	18781.78

Note: CPU time limit for MPM is set at 10800 s for Examples 1–10, 36000 s for Example 11, and 108000 s for Examples 12–14.
CPU time limit for RMPM is set at 3600 s for Examples 1–10 and 10800 s for Examples 11–14.
*The time when final solution is found, but termination at CPU time limit.
†Reached CPU time limit.

Table 11. RMIPs and Best Possible Solution for Example 1–14 from SPM

Example	RMIP from SPM (k$)	Best Possible Solution from SPM (k$)	Dev-1 (%)	Cost from SPM (k$)	Cost from RSPM (k$)	Dev-2 (%)
1	4988.52	5149.73	3.23	5149.73	5149.73	0.00
2	3570.65	3658.11	2.45	3658.11	3658.11	0.00
3	3105.08	3159.12	1.74	3159.12	3159.12	0.00
4	4501.78	4556.67	1.22	4556.67	4556.67	0.00
5	4501.78	4556.67	1.22	4556.67	4556.67	0.00
6	4407.80	5125.99	16.29	5213.88	5213.88	1.69
7	7797.02	7797.02	0.00	8100.35	8100.35	3.74
8	7777.02	7777.02	0.00	8100.35	8100.35	3.99
9	10,003.92	10,013.90	0.10	10,803.23	10,750.49	6.85
10	10,608.50	10,616.19	0.07	11,543.60	11,629.09	8.71
11	12,472.99	12,486.17	0.11	13,444.14	13,476.48	7.35
12	13,731.95	13,732.02	0.00	15,053.85	15,016.66	8.55
13	14,682.32	14,695.91	0.09	16,536.00	16,146.52	8.98
14	16,583.22	16,586.76	0.02	18,547.36	18,233.61	9.03

Dev-1: The relative gap between columns 2 and 3.
Dev-2: The relative gap between columns 3 and 6.

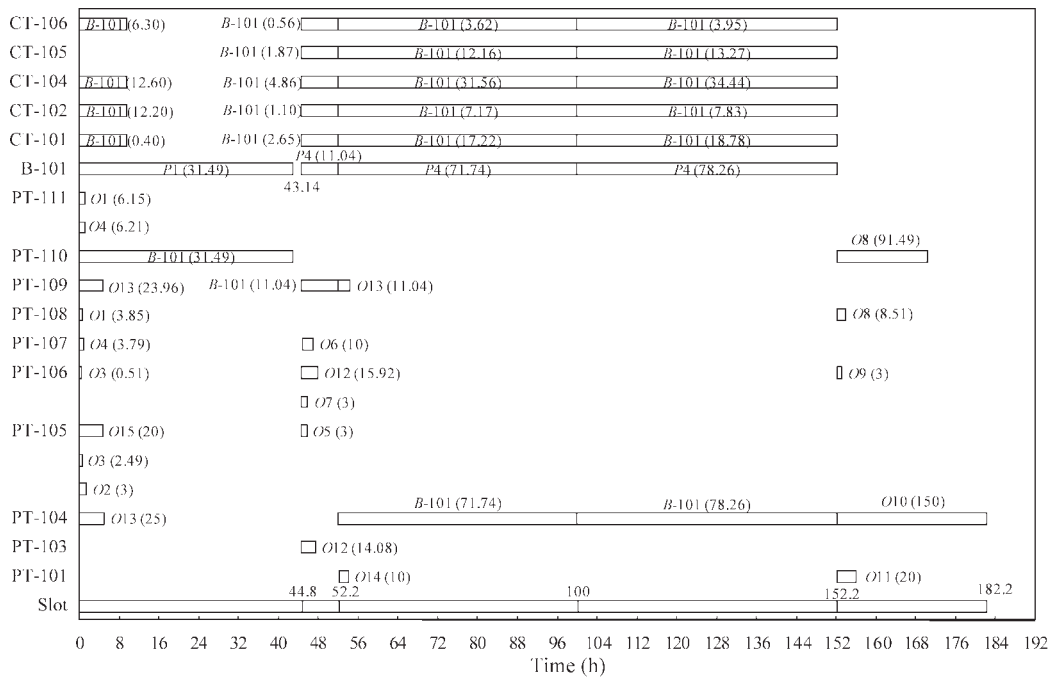

Figure 9. Optimal schedule for Example 5 (15 orders) from RMPM.

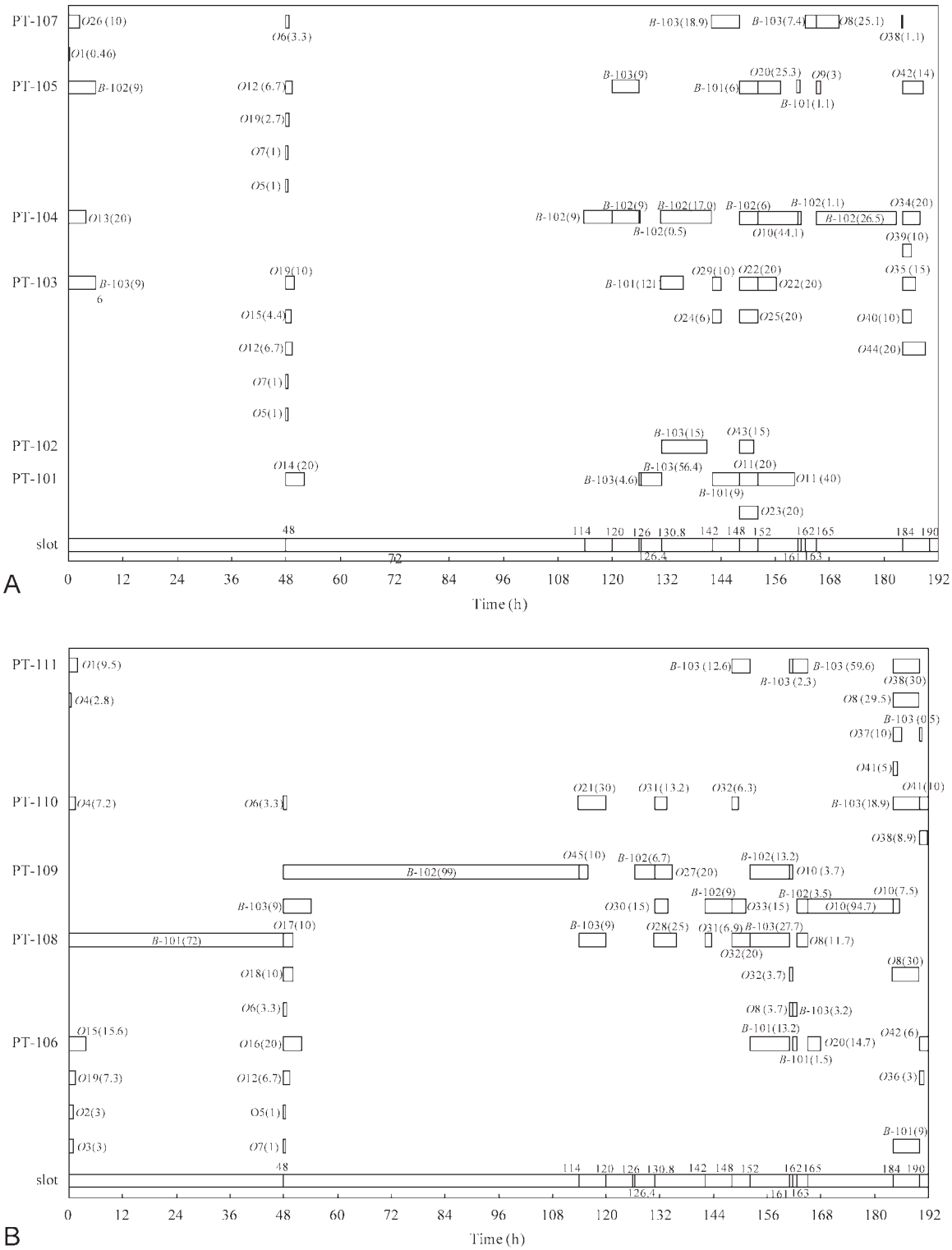

Figure 10. (a) Order delivery schedule for PT-101 to PT-105, and PT-107 for Example 14 (45 orders) from RSPM. (b) Order delivery schedule for PT-106 and PT-108 to PT-111 for Example 14 (45 orders) from RSPM. (c) Blending schedule for Example 14 (45 orders) from RSPM.

3) Some tanks deliver multiple orders simultaneously in Figures 9 and 10. For instance, PT-105 delivers O2, O3, and O15 during slot 1 in Figure 9, and PT-106 delivers O2, O3, O15, and O19 during slot 1 in Figure 10b.

4) Multiple product tanks deliver a single order simultaneously in Figures 9 and 10. For instance, PT-108 and PT-110 deliver O8 during slot 5 in Figure 9, and PT-103 and PT-105 deliver O19 during slot 2 in Figure 10a.

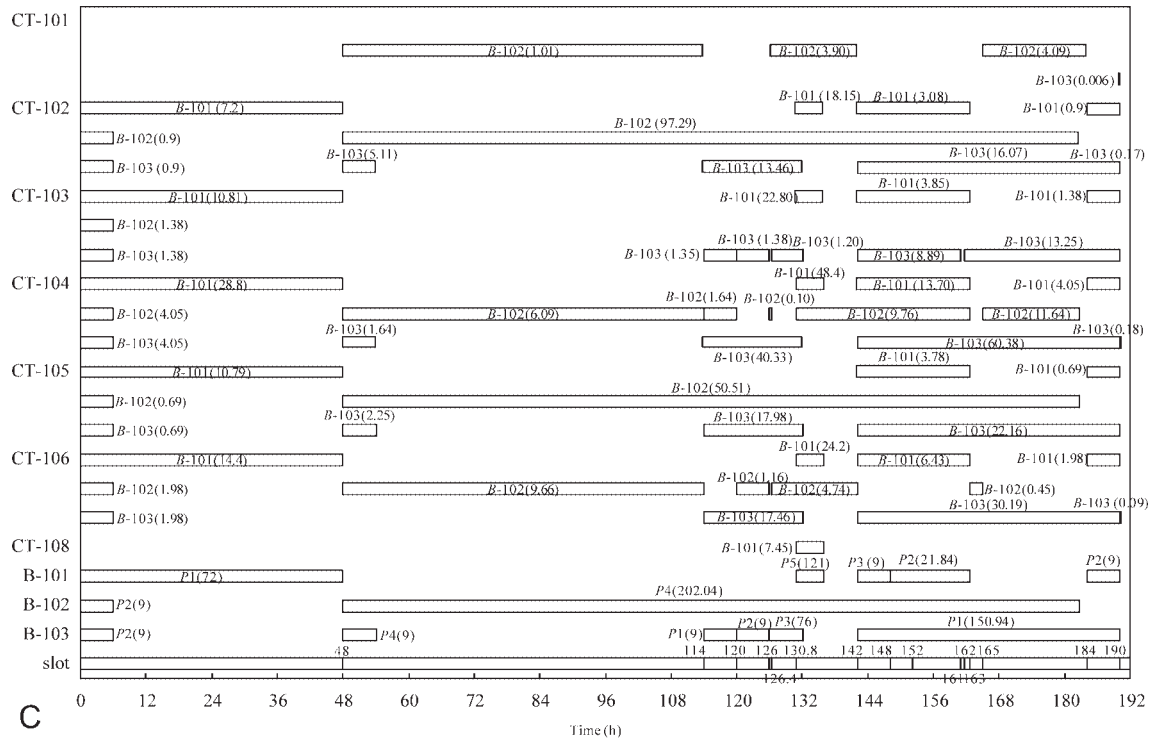

Figure 10. (Continued)

5) Multiple blenders (B-102 and B-103) process the same product (P2) simultaneously during slot 1 in Figure 10c. They again process P4 during slot 2.

MINLP Formulation

Recall that forcing the blending rate to be constant during a run makes the formulation nonlinear and nonconvex. Our adjustment procedure obviated the need to solve MINLPs. To show the effectiveness of our procedure, we used DICOPT/GAMS[17] and BARON/GAMS[17] to solve the nonlinear, nonconvex formulations derived as follows.

We define variable F_{bk} ($F_b^{\mathrm{L}} \le F_{bk} \le F_b^{\mathrm{U}}$) as the blending rate of blender b during slot k and impose the following constraints to maintain a single blending rate for each run

$$Q_{bk} \le F_{bk}\mathrm{SL}_k \qquad 0 < k \le K \tag{55a}$$

$$Q_{bk} \ge F_{bk}SL_k - M_b(v_{b0k} + xe_{bk}) \qquad 0 < k \le K \tag{55b}$$

$$F_{b(k+1)} \ge F_{bk} - F_b^U xe_{bk} \qquad 0 < k \le K \tag{56a}$$

$$F_{b(k+1)} \le F_{bk} + F_b^U xe_{bk} \qquad 0 < k \le K \tag{56b}$$

We replace Eq. 17 by Eqs. 55–56 in SPM and MPM to get MINLP-SPM and MINLP-MPM. For a fair comparison, we allow the same CPU time for DICOPT/GAMS[17] and BARON/GAMS,[17] as we did for our algorithm. For Examples 6–14, we had set the time limits for SPM/MPM and RSPM/RMPM. We also set the same limits for the MIPs of DICOPT/GAMS[17] as shown in Tables 12 and 13.

Tables 12 and 13 show the solution statistics for Examples 1–14. For Example 1, our procedure needs only 0.83 CPU s for the optimal solution of 5149.73 k$, but DICOPT/GAMS[17] needs 4.8 CPU s. BARON/GAMS[17] obtains a suboptimal solution of 5169.73 k$ after 14,400 CPU s. For Example 2, our procedure obtains the optimal solution of 3658.11 k$ within 48.6 CPU s, but BARON/GAMS[17] needs 14,400 CPU s. DICOPT/GAMS[17] gets a worse solution of 4169.81 k$ in 151 CPU s. For Examples 3–5, DICOPT/GAMS[17] does get the optimal solutions, but requires an order of magnitude longer solution times compared with our algorithm. For instance, it takes 3636 s for Example 4 vs. only 645 CPU s for our algorithm. Interestingly, BARON/GAMS[17] also reaches the optimal solution for Example 3, but needs 14,400 s. For the remaining examples (Examples 6–14), our approach always obtains better solutions than both DICOPT/GAMS[17] and BARON/GAMS[17] within the allocated CPU times. For instance, our approach finds a solution of 8100.35 k$ for Example 8, whereas DICOPT/GAMS[17] obtains 13,447.99 k$, and BARON/GAMS[17] gets 9941.99 k$. Similarly, our approach obtains a solution of 13,476.48 k$ for Example 11, whereas BARON/GAMS[17] and DICOPT/GAMS[17] cannot obtain a feasible solution.

Conclusion

We developed a slot-based continuous-time model for integrated scheduling of gasoline blending operations in a refinery. The model incorporates many real-life operating features and policies such as multiple parallel nonidentical blenders, piecewise constant component input flows and

Table 12. Solution Statistics of Various Algorithm/Code for SPM for Example 1–14

Example	Order	Algorithm	Discrete Variables	Continuous Variables	Constraints	Total CPU Time (s)	Cost (k$)
1	5	DICOPT	100	274	1105	4.8	5149.73
		BARON	100	274	1105	14,400*	5169.73
		Ours	100	269	1098	0.83	5149.73
2	10	DICOPT	252	603	2854	151	4169.81
		BARON	252	603	2854	14,400*	3658.11
		Ours	252	598	2847	48.6	3658.11
3	12	DICOPT	315	638	3206	1020	3159.12
		BARON	315	638	3206	14,400*	3159.12
		Ours	315	633	3199	178	3159.12
4	15	DICOPT	372	728	3862	3636	4556.67
		BARON	372	728	3862	14,400*	6182.57
		Ours	372	723	3855	645	4556.67
5	15	DICOPT	372	728	3874	927	4556.67
		BARON	372	728	3874	14,400*	4821.55
		Ours	372	723	3867	216	4556.67
6	18	DICOPT	576	956	5487	14,400*	6364.90
		BARON	576	956	5487	14,400*	6380.21
		Ours	576	950	5478	10,801	5213.88
7	20	DICOPT	775	1204	7186	14,400*	8100.35
		BARON	775	1204	7186	14,400*	9648.97
		Ours	775	1197	7175	10,803	8100.35
8	20	DICOPT	682	1076	7682	14,400*	13,447.99
		BARON	682	1076	7682	14,400*	9941.99
		Ours	682	1164	7664	10,805	8100.35
9	23	DICOPT	1251	1908	13,952	14,400*	N/A
		BARON	1251	1908	13,952	14,400*	N/A
		Ours	1251	1890	13,922	14,400*	10,750.49
10	25	DICOPT	1503	2229	16,526	14,400*	15,614.97
		BARON	1503	2229	16,526	14,400*	N/A
		Ours	1503	2209	16,492	14,400*	11,629.09
11	30	DICOPT	2328	3242	24,801	46,800*	N/A
		BARON	2328	3242	24,801	46,800*	N/A
		Ours	2328	3216	24,755	40,065	13,476.48
12	35	DICOPT	3846	4997	39,582	118,800*	N/A
		BARON	3846	4997	39,582	118,800*	N/A
		Ours	3846	4963	39,520	118,800*	15,016.66
13	40	DICOPT	4414	6020	51,874	118,800*	27,662.61
		BARON	4414	6020	51,874	118,800*	N/A
		Ours	4414	5966	51,775	118,800*	16,46.52
14	45	DICOPT	4532	6128	52,878	118,800*	24,601.27
		BARON	4532	6128	52,878	118,800*	N/A
		Ours	4532	6077	52,785	118,800*	18,233.61

CPU time limit for MIP of DICOPT is set at 10800 s for Example 1–10, 36000 s for Example 11, and 108000 s for Examples 12–14.
CPU time limit for SPM of ours is set at 10800 s for Examples 1–10, 36000 s for Example 11, and 108000 s for Examples 12–14.
CPU time limit for RSPM of ours is set at 3600 s for Examples 1–10, and 10800 s for Examples 11–14.
Total CPU time limit of DICOPT, BARON, and ours is set at 14,400 s for Examples 1–10, 46800 s for Example 11, and 118800 s for Examples 12–14.
*Reached total CUP time limit.

qualities, multi-product orders with multiple delivery dates, multi-purpose product tanks, minimum lengths for blending runs, constant blending rate in a run, common transfer policies, blending and storage transitions, etc. Although the problem is inherently nonconvex and nonlinear, we proposed an ingenious schedule adjustment procedure that requires only MILP solutions. On 14 test problems of varying sizes and features, our proposed procedure obtained the same or better solutions than commercial solvers such as DICOPT/GAMS[17] and BARON/GAMS.[17] Furthermore, it needed an order of magnitude shorter solution times than DICOPT/GAMS[17] and BARON/GAMS.[17] In addition, our model can easily be simplified for assumptions such as identical blenders, one blender per product at a time, etc, which are common in existing work. Much further work is needed, as our model still cannot optimally solve truly large problems involving more than 30 orders within reasonable time.

Acknowledgments

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

Notation

Sets

BJ = set of pairs (blender b, product tank j) such that blender b can feed product tank j
BP = set of (b, p) pairs such that blender b can process product p
JO = set of pair (product tank j, order o) such that tank j can deliver order o
OP = set of pair (order o, product p) such that order o is for product p
PJ = set of pairs (product p, product tank j) such that tank j can hold product p
TK = set of pair (slot k, period t) such that slot k is in period t

Table 13. Solution Statistics of Various Algorithms/codes for MPM for Examples 1–14

Example	Order	Algorithm	Discrete Variables	Continuous Variables	Constraints	Total CPU Time (s)	Cost (k$)
1	5	DICOPT	130	334	1393	78.7	5149.73
		BARON	130	334	1393	14400*	5149.73
		Ours	130	328	1384	12.6	5149.73
2	10	DICOPT	329	735	3609	798	3658.11
		BARON	329	735	3609	14400*	3678.11
		Ours	329	729	3600	137	3658.11
3	12	DICOPT	411	775	4057	531	3179.12
		BARON	411	775	4057	14400*	3179.12
		Ours	411	769	4048	82.7	3179.12
4	15	DICOPT	486	883	4889	3173	4556.67
		BARON	486	883	4889	14400*	4881.28
		Ours	486	877	4880	495	4556.67
5	15	DICOPT	486	883	4901	571	4567.19
		BARON	486	883	4901	14400*	N/A
		Ours	486	877	4892	65.2	4556.67
6	18	DICOPT	712	1121	6642	14400*	5704.93
		BARON	712	1121	6642	14400*	5516.98
		Ours	712	1114	6631	10801	5213.88
7	20	DICOPT	1219	1735	10939	14400*	8191.39
		BARON	1219	1735	11155	14400*	N/A
		Ours	1219	1725	10922	10801	8100.35
8	20	DICOPT	1159	1797	12494	14400*	11446.89
		BARON	1159	1797	12494	14400*	10156.62
		Ours	1159	1779	12464	10834	8114.85
9	23	DICOPT	1423	2127	15742	14400*	N/A
		BARON	1423	2127	15742	14400*	N/A
		Ours	1423	2107	15708	10811	10613.65
10	25	DICOPT	1867	2687	20286	14400*	N/A
		BARON	1867	2687	20286	14400*	13473.86
		Ours	1867	2663	20244	14400*	11488.35
11	30	DICOPT	2328	3242	25521	46800*	N/A
		BARON	2328	3242	25521	46800*	N/A
		Ours	2328	3216	24755	46800*	13622.18
12	35	DICOPT	4099	5294	42092	118800*	16618.59
		BARON	4099	5294	42092	118800*	N/A
		Ours	4099	5258	42026	108597	15321.74
13	40	DICOPT	4685	6359	54962	118800*	N/A
		BARON	4685	6359	54962	118800*	N/A
		Ours	4686	6302	54857	118800*	16653.07
14	45	DICOPT	4830	7493	56226	118800*	24634.40
		BARON	4830	7493	56226	118800*	N/A
		Ours	4830	6439	56127	108256	18781.78

CPU time limit for MIP of DICOPT is set at 10,800 s for Examples 1–10, 36,000 s for Example 11, and 108,000 s for Examples 12–14.
CPU time limit for MPM of ours is set at 10,800 s for Examples 1–10, 36,000 s for Example 11, and 108,000 s for Examples 12–14.
CPU time limit for RMPM of ours is set at 3600 s for Examples 1–0 and 10,800 s for Examples 11–14.
Total CPU time limit of DICOPT, BARON, and ours is set at 14,400 s for Examples 1–10, 46,800 s for Example 11 and 118,800 s for Examples 12–14.
*Reached total CPU time limit.

Subscripts

b = blender
i = blend component and its dedicated tank
j = product tank
k = slot
o = order
p = product
s = gasoline property specification
t = period

Superscripts

L = lower limit
U = upper limit

Parameters

c_i = price ($ per unit volume) of component i
CB_b = cost ($ per occurrence) of a transition on blender b
CCQ_{bk} = corrected volume processed by blender b during slot k
CRL_{bk} = corrected run length of blender b at the end of slot k
CT_j = cost ($ per occurrence) of a transition in product tank j
DD_o^L = earliest delivery time of order o
DD_o^U = due date of order o
DM_o = demurrage ($ per unit time) for order o
DR_{jo} = delivery rate of product tank j to order o
DR_j^U = maximum cumulative delivery rate of product tank j
F_b^{LU} = limits on the processing rate of blender b
F_i = constant feed rate of component i into its tank
F_{it} = constant feed rate of component i to its tank during period t
H = scheduling horizon
M_b = most volume that blender b can process during a slot
N = the number of products that are needed to process in blenders
N_b = the number of blenders
r_{pi}^{LU} = limits on the fraction of component i in product p
R_{bk} = rate of blender b in slot k
RL_b = maximum of the minimum blend run lengths for blender b
RL_{bp}^L = minimum run length of blender b for product p
T_0 = time zero or the time at which slot 1 starts
TCQ_{bk} = total volume processed by blender b during the current run at the end of slot k
TQ_o = amount of order o

V_i^{LIU} = limits on the holdup in component tank i
$\mathrm{VP}_j^{\mathrm{U}}$ = capacity of product tank j
θ_{ps}^{U} = limits on the index for property s of product p
θ_{is} = blend index of property s of component i
θ_{ist} = blend index for a property s of component i during period t
$\rho_{\max}$ = maximum density among all products
ρ_i = density of component i
$\rho_i^{\max}$ = maximum possible density among all products during period t
ρ_{it} = density of component i during period t

Binary variables

u_{jpk} = 1, if product p is stored in product tank p during slot k
v_{bjk} = 1, if blender b feeds product tank j ($0 < j \leq \boldsymbol{J}$) during slot k
z_{jok} = 1, if product tank j is delivering order o during slot k

0–1 Continuous variables

ue_{jk} = 1 If product tank j switches products at the end of slot k
v_{b0k} = 1 If blender b is idle during slot k
x_{bpk} = 1 If blender b produces product p during slot k
xe_{bk} = 1 If blender b ends its current run for a product during slot k

Continuous variables

CQ_{bk} = volume processed by blender b during the current run, if the run does not end during slot k
d_o = demurrage ($) for order o
DQ_{jok} = volume of order o delivered by product tank j during slot k
F_{bk} = rate of blender b during slot k
G_{bjk} = volume that blender b feeds product tank j during slot k
q_{ibk} = volume of component i used by blender b during slot k
Q_{bk} = volume processed in blender b during slot k
RL_{bk} = length of the current run of blender b at the end of slot k
SL_k = length of slot k
T_k = time at which slot k ends
TC = total operating cost ($)
V_{ik} = inventory in component tank i at the end of slot k
VP_{jk} = inventory in product tank j at the end of slot k

Literature Cited

1. Pinto JM, Joly M, Moro LFL. Planning and scheduling models for refinery operations. *Comput Chem Eng*. 2000;24:2259–2276.
2. Jia ZY, Ierapetritou M. Mixed-integer linear programming model for gasoline blending and distribution scheduling. *Ind Eng Chem Res*. 2003;42:825–835.
3. Reddy PCP, Karimi IA, Srinivasan R. A new continuous-time formulation for scheduling crude oil operations. *Chem Eng Sci*. 2004;59:1325–1341.
4. Reddy PCP, Karimi IA, Srinivasan R. A novel solution approach for optimizing scheduling crude oil operations. *AIChE J*. 2004;50:1177–1197.
5. Li J, Li WK, Karimi IA, Srinivasan R. Improving the robustness and efficiency of crude scheduling algorithms. *AIChE J*. 2007;53:2659–2680.
6. Dewitt CW, Lasdon LS, Waren AD, Brenner DA, Melhem SA. OMEGA: an improved gasoline blending system for Texaco. *Interfaces*. 1989;19:85–101.
7. Rigby B, Lasdon LS, Waren AD. The evolution of Texaco's blending systems: from OMEGA to Starblend. *Interfaces*. 1995;25:64–83.
8. Li WK, Hui CW, Li AX. Integrating CDU, FCC and product blending models into refinery planning. *Comput Chem Eng*. 2005;29:2010–2028.
9. Karimi IA, McDonald CM. Planning and scheduling of parallel semicontinuous processes. Part 2: short-term scheduling. *Ind Eng Chem Res*. 1997;36:2701–2714.
10. Joly M, Pinto JM. Mixed-integer programming techniques for the scheduling of fuel oil and asphalt production. *Trans IChemE A*. 2003;81:427–447.
11. Glismann K, Gruhn G. Short-term scheduling and recipe optimization of blending processes. *Comput Chem Eng*. 2001;25:627–634.
12. Mendez CA, Grossmann IE, Harjunkoski I, Kabore P. A simultaneous optimization approach for off-line blending and scheduling of oil-refinery operations. *Comput Chem Eng*. 2006;30:614–634.
13. Kelly JD, Mann JL. Crude oil blend scheduling optimization: an application with multimillion dollar benefits. *Hydrocarbon Process*. 2003;82:47–51.
14. Kelly JD. The unit-operation-stock superstructure (UOSS) and the quantity-logic-quality paradigm (QLQP) for production scheduling in the process industries. Presented at: *MISTA Conference*, New York, July 18–21, 2005.
15. Kelly JD. Next-generation refinery scheduling technology. Presented at: *NPRA Plant Automation and Decision Support Conference*, San Antonio, TX, September 2003.
16. Li J, Karimi IA, Srinivasan R. Robust and efficient algorithm for optimization crude oil operations. Presented at: *AIChE Annual Meeting*, Cincinnati, OH, October 30–November 4, 2005.
17. Brooke A, Kendrick D, Meeraus A, Raman R. *GAMS: a user's guide*. Washington, DC: GAMS Development Corporation, 1998.
18. Liu Y, Karimi IA. Scheduling multistage, multiproduct batch plants with nonidentical parallel units and unlimited intermediate storage. *Chem Eng Sci*. 2007;62:1549–1566.

Appendix A

The model of Jia and Ierapetritou[2] has two basic problems.

First, the model is infeasible as shown below. Equations 16a, 11a, and 11b (denoted as JI-16a, JI-11a, etc. here) from their model are

$$\sum_{j\in J} \mathrm{Blnd}_{sjn} \geq 6\,\mathrm{Bflow} \quad \forall s \in \mathbf{S}, \quad n \in \mathbf{N} \qquad \text{(JI-16a)}$$

$$sv_{sjn} \leq xv_{sn} \leq \sum_{j\in J_s} sv_{sjn} \quad \forall s \in \mathbf{S}, \quad n \in \mathbf{N} \qquad \text{(JI-11a)}$$

$$\sum_{s} xv_{sn} \leq 1 \qquad \forall s \in \mathbf{S}, \quad n \in \mathbf{N} \qquad \text{(JI-11b)}$$

Because Bflow is a positive blend rate, product s is being produced and must be transferred from the blender to at least one product tank j at event point n. Therefore, there must be at least one j such that

$$sv_{sjn} \geq 1 \quad \forall s \in \mathbf{S}, \quad n \in \mathbf{N} \qquad \text{(A1)}$$

where $sv_{sjn} = 1$, if product s is produced and transferred to tank j at event point n.

From Eqs. A.1 and JI-11a, we get $xv_{sn} = 1$. Hence,

$$\sum_{s} xv_{sn} = N \quad \forall s \in \mathbf{S}, \quad n \in \mathbf{N} \qquad \text{(A2)}$$

where, N is the number of products that are needed to process in blenders in the problem. Since $N > 1$ for most problems, $\sum_s xv_{sn} > 1$, and that contradicts Eq. (JI-11b).

Second, it allows a tank to hold multiple products at a time, as shown below. They used the following (Eq. 2 in their article) to force the inventory of product s in tank j to be zero, if j does not hold s at event point n

$$V_j^{\min} y_{sjn} \leq Pst_{sjn} + \mathrm{Blnd}_{sjn} \leq V_j^{\max} y_{sjn} \quad \forall s \in \mathbf{S}, \quad j \in J_s,\ n \in \mathbf{N} \qquad \text{(JI-2)}$$

where, $y_{sjn} = 1$, if j holds s at event point n. Note that Eq. JI-2 cannot guarantee that j holds at most one product at an event point.

Appendix B

We prove that Eqs. 2, 4, 7, and 9 make x_{bpk} binary. Recall that a blender b is either idle or feeding a product tank at any time.

1. If b is idle during slot k, then $v_{b0k} = 1$ and $x_{bpk} = 0$ for $(b, p) \in \boldsymbol{BP}$ from Eq. 9.

2. If b is not idle during k, then $v_{b0k} = 0$ and $v_{bjk} = 1$ for one j with $(b, j) \in \boldsymbol{BJ}$ and $v_{bj'k} = 0$ for $j' \neq j$ from Eq. 2. If product tank j holds a product p during slot k, then $u_{jpk} = x_{bpk} = 1$ from Eq. 7. Equation 9 then forces $x_{bp'k} = 0$ for $p' \neq p$.

Appendix C

We prove that our adjustment procedure ensures that constant blend rate satisfies the limits on the blending rates and the minimum run length at the same time. From Eqs. 16 and 17, we get $0 \leq Q_{bk} \leq F_b^{\mathrm{U}}\ \mathrm{SL}_k$. Then, using Eqs. 37 and 38, we have $0 \leq \mathrm{CCQ}_{bk} \leq F_b^{\mathrm{U}}\ \mathrm{CRL}_{bk}$ for k with $xe_{bk} = 0$, and $0 \leq \mathrm{CCQ}_{b(k-1)} + Q_{bk} \leq F_b^{\mathrm{U}}\ [\mathrm{CRL}_{b(k-1)} + \mathrm{SL}_k]$ for k with $xe_{bk} = 1$. In other words,

$$0 \leq \frac{\mathrm{CCQ}_{b(k-1)} + Q_{bk}}{\mathrm{CRL}_{b(k-1)} + SL_k} \leq F_b^{\mathrm{U}} \quad \text{for } xe_{bk} = 1 \qquad \text{(C1)}$$

The above along with Eq. 40 ensures $F_b^{\mathrm{L}} \leq R_{bk} \leq F_b^{\mathrm{U}}$ for $xe_{bk} = 1$ and $v_{b0k} = 0$. Thus, the optimal solution from RSPM will satisfy the blend rate limits.

For $xe_{bk} = 1$, we get the following from Eqs. 20 and 39.

$$\mathrm{TCQ}_{bk} \geq F_b^{\mathrm{L}} \sum_{p=1}^{P} \mathrm{RL}_{bp}^{\mathrm{L}} x_{bpk} \quad \text{for } xe_{bk} = 1, \quad (b,p) \in \boldsymbol{BP},\ 0 < k \leq K \qquad \text{(C2)}$$

If $F_b^L \geq \frac{\mathrm{CCQ}_{b(k-1)}+Q_{bk}}{\mathrm{CRL}_{b(k-1)}+\mathrm{SL}_k}$, for k with $xe_{bk} = 1$ and $v_{b0k} = 0$, then $R_{bk} = F_b^{\mathrm{L}}$ from Eq. 40, and

$$\frac{\mathrm{TCQ}_{bk}}{R_{bk}} = \frac{\mathrm{TCQ}_{bk}}{F_b^{\mathrm{L}}} \geq \sum_{p=1}^{P} \mathrm{RL}_{bp}^{\mathrm{L}} x_{bpk} \quad \text{for } xe_{bk} = 1, (b,p) \in \boldsymbol{BP},\ 0 < k \leq K \qquad \text{(C3)}$$

which means that the run length exceeds the minimum.

If $F_b^{\mathrm{L}} \leq \frac{\mathrm{CCQ}_{b(k-1)}+Q_{bk}}{\mathrm{CRL}_{b(k-1)}+\mathrm{SL}_l}$ for k with $xe_{bk} = 1$ and $v_{b0k} = 0$, then $R_{bk} = \frac{\mathrm{CCQ}_{b(k-1)}+Q_{bk}}{\mathrm{CRL}_{b(k-1)}+\mathrm{SL}_k}$, and

$$\frac{\mathrm{TCQ}_{bk}}{R_{bk}} = \frac{\mathrm{CCQ}_{b(k-1)} + Q_{bk}}{\frac{\mathrm{CCQ}_{b(k-1)}+Q_{bk}}{\mathrm{CRL}_{b(k-1)}+\mathrm{SL}_k}} = \mathrm{CRL}_{b(k-1)} + \mathrm{SL}_k \quad \text{for } xe_{bk} = 1, (b,p) \in \boldsymbol{BP},\ 0 < k \leq K \qquad \text{(C4)}$$

From Eqs. 12–14, 37, and 38, we know that CRL_{bk} satisfies Eq. 14, and hence,

$$\mathrm{CRL}_{b(k-1)} + \mathrm{SL}_k \geq \sum_{p=1}^{P} RL_{bp}^{\mathrm{L}} x_{bpk} \quad \text{for } xe_{bk} = 1, (b,p) \in \boldsymbol{BP},\ 0 < k \leq K \qquad \text{(C5)}$$

Equations C4 and C5 show that the run length exceeds the minimum for this case too.

Manuscript received Oct. 12, 2008, and revision received Apr. 8, 2009.