Inventory Pinch Based, Multiscale Models for Integrated Planning and Scheduling-Part II: Gasoline Blend Scheduling

Pedro A. Castillo Castillo and Vladimir Mahalec
Dept. of Chemical Engineering, McMaster University, Hamilton, ON, Canada L8S 4L8

DOI 10.1002/aic.14444
Published online April 1, 2014 in Wiley Online Library (wileyonlinelibrary.com)

Integration of planning and scheduling optimizes simultaneous decisions at both levels, thereby leading to more efficient operation. A three-level discrete-time algorithm which uses nonlinear models and integrates planning and detailed scheduling is introduced: first level optimizes nonlinear blend models via multiperiod nonlinear programming (NLP), where period boundaries are initially determined by the inventory pinch points; second level uses fixed recipes (from the first level) in a multiperiod mixed-integer linear program to determine first an optimal production plan and then to optimize an approximate schedule which minimizes the total number of switches in blenders and swing tanks; third level computes detailed schedules that adhere to inventory constraints computed in the approximate schedule. If inventory infeasibilities appear at the second or the third level, the first-level periods are subdivided and blend recipes are reoptimized. Algorithm finds the same or better solutions and is substantially faster than previously published full-space continuous-time model. © 2014 American Institute of Chemical Engineers *AIChE J*, 60: 2475–2497, 2014

Keywords: gasoline blend planning, scheduling, inventory pinch, recipe optimization, minimum number of recipes, multiscale models

Introduction

In an oil refinery, gasoline can account for 60–70% of the total profit[1–3]; therefore, minimization of blending operation costs represents a huge opportunity to increase profit margins. Oil refineries have to deal with unsteady product demands and crude prices, as well as continuously stricter environmental regulations.[4] In such a context, computational supply chain optimization tools based on mathematical models have become very important for companies to plan and schedule their operations in the best way possible with the objective to reduce costs and maximize revenues.[5] Market opportunities in the short term should be exploited without compromising the objectives in the long term.[6] Optimization of blending operations (as the production of gasoline in an oil refinery) involves the fulfillment of product quality specifications and demand requirements at the minimum cost, subject to raw materials availability and production and storage capacity limits. Kelly[7] pointed out that quality and quantity details are not the only elements comprising the optimization problem of blending systems, and he described several logistical details that, if incorporated into the blend scheduling problem, will provide a more accurate production schedule. Some of the logic constraints listed by Kelly[7] are the following: (1) a flow (i.e., the total volume pumped/processed or a flow/production rate) must be between its lower and upper bounds, (2) a flow must be equal across contiguous time periods if a specific task has not finished, (3) processing units can only execute one task at a time, (4) a tank can only store a new material when its holdup is less than or equal to the quantity specified (this type of tanks are called swing tanks as they can switch to a different material service), (5) minimum and maximum bounds on the running time (up-time) and idle time (down-time) of a unit, task, or operation mode, may be specified, (6) the use of sequence-dependent and -independent changeover down-times may be considered, and (7) an order (internal or external) cannot be fulfilled outside its specified time window, but material flow may be specified to be constant or allowed to be intermittent within the window.

Many authors have worked on the integration of production planning and scheduling optimization problems, either developing easier-to-converge models or designing more efficient algorithms to solve them. Maravelias and Sung[8] reviewed the opportunities and challenges of integrating the planning and scheduling levels, pointing at the importance of developing more computationally effective models for complex process systems, improving decomposition and iterative algorithms, and the possibility to solve the scheduling problem more efficiently by developing hybrid methods using different solution techniques. A broad classification of scheduling formulation approaches is also found in Maravelias,[9] Maravelias and Sung,[8] and Mendez et al.[10] Gasoline blending falls into the category of network-based formulations due to the continuous nature of the process. With respect to the time representation used, the formulated problems can be classified as discrete-, continuous-, or mixed-time models. In the discrete-time models, the scheduling horizon is divided in a given number of time periods whose duration is known *a priori*, whereas in the continuous-time models the length of these periods is not known in advance (for this reason, the word “time slot” is preferred when referring to this type

Corresponding concerning this article should be addressed to V. Mahalec at mahalec@mcmaster.ca.

of time intervals). An in-depth review of advantages and disadvantages of discrete- and continuous-time formulations can be found in Floudas and Lin[11] and Sundaramoorthy and Maravelias.[12] Joly and Pinto[13] developed a discrete-time mixed-integer linear programming (MILP) model for the scheduling of fuel oil and asphalt production, and they pointed out that, although continuous-time formulations may decrease significantly the combinatorial feature of a model, discrete-time models may still be a good option because (1) the resource constraints are easier to handle (e.g., products between flow rates and time intervals are linear) and (2) discrete-time models provide tight formulations in general. In mixed-time formulations, the time grid is fixed but the durations of the tasks are variable.

In principle, to solve an integrated planning and scheduling problem it is enough to write a discrete- or continuous-time model, that is, the full-space model, and solve it; however, for real-life, large-scale problems, this will lead to intractable mixed-integer nonlinear programming (MINLP) or MILP models. Given their large-scale combinatorial nature, scheduling problems are at least nondeterministic polynomial time (NP) complete.[6] In the last decade, researchers have been working on improving full-space model formulations in order to avoid prohibited execution times.

Jia and Ierapetritou[1] solved simultaneously the gasoline blend scheduling and distribution problem. They presented a continuous-time event-based MILP model for the scheduling problem. The model includes multipurpose product tanks (tank switching), delivery of the same order from multiple product tanks, and one product tank delivering multiple orders. A set of preferred blend recipes is given (i.e., blend recipes are not optimized). Their largest problem (one blender, four products, 11 product tanks, nine blend components, 45 orders, and a scheduling horizon of 8 days) was solved to proven optimality in 5 CPU hours.

Mendez et al.[3] introduced an iterative algorithm to optimize blend recipes and schedule blending operations. Nonlinear quality constraints are modeled as linear constraints by using correction factors. At the end of each iteration, these correction factors for the product properties are calculated according to the blend recipes computed. The algorithm stops when the correction factors converge and the products properties fulfill the specifications. Minimum blend run constraints and multipurpose tanks are features not included in the model. Due to the assumption made in their case studies that each blender produces only one particular gasoline grade (i.e., the sequencing problem is avoided), their computational times are very small (less than 2 s).

Li et al.[2] presented a continuous-time slot-based MILP model that uses process slots. This model includes the blend recipe optimization, inventory constraints, blender capacity constraints, and delivery scheduling for the demand orders. Blend indices are used instead of the actual quality properties in order to avoid nonlinear constraints. Their model also includes parallel nonidentical blenders, multipurpose tanks, and other attributes and constraints found in industrial practice. After the model is solved, a schedule adjustment step is required to ensure that each blend run has a constant blending rate. Although their formulation incorporates many details of the industrial systems, computational times of more than 20 h were required to solve examples for a blending system of significant size (e.g., 3 blenders, 9 and 11 component and product storage tanks, respectively) and a scheduling horizon of 8 days; nevertheless, their solutions were better than those provided by DICOPT and BARON solving the corresponding MINLP model in the same time.

Li and Karimi[4] replaced process slots with unit slots and expanded the model by Li et al.[2] to include blender setup times, limited inventory of components, and simultaneous receipt/delivery by the product tanks. Due to the reduction in the number of discrete variables when using unit slots instead of process slots, computational times improved significantly for small- and medium-size problems; however, large-scale problems (e.g., 2–3 blenders, 5 products, 9 components, 9 properties, 11 product tanks, 35–45 orders, and a planning horizon of 8 days) used all the allocated CPU time (46,800–118,800 s, depending on the problem) as the solution did not meet the stopping criteria.

Decomposition techniques have been used to solve more efficiently the integrated planning and scheduling problem, as well as the scheduling problem itself. There is usually a trade off between shorter execution times and the quality of the solution obtained depending of the decomposition method used.

Bassett et al.[14] reviewed several time-based decomposition approaches to solve the scheduling problem; these decompositions are based on subdividing the scheduling horizon in smaller intervals, solving the corresponding subproblems in a specific sequence, and applying heuristics methods to combine the solutions.

Elkamel et al.[15] presented a spatial and a temporal decomposition to schedule batch processes in a general chemical plant. The spatial decomposition is based on grouping units which perform similar tasks and their corresponding orders. In the temporal decomposition, the product orders are grouped according to their due dates, and the last due date of each group delineates the time where the scheduling horizon is subdivided. Global optimality is only guaranteed if all the subproblems are independent.

Munawar and Gudi[16] proposed a three-level hierarchical approach to integrate the planning and scheduling decisions in the multistage hybrid flowshop problem for a single facility. The first level consists of the midterm planning model and its solution provides the production targets for each time period. Slopping loses are assumed at this level and the objective function maximizes production in the initial periods in order to have production capacity available in later periods to handle unexpected events (e.g., demand variations, machine breakdowns etc). At the second level, the schedule is computed using a continuous-time model (which is solved sequentially for each of the planning periods) that maximizes profit and penalizes product changeover and inventory costs. Actual slopping loses are calculated at this level, and inventory upper bounds are overestimated based on heuristics or previous process knowledge. The third level determines the detailed product-to-tank assignments using a heuristic algorithm to find the minimum number of tanks required to manage the inventory levels provided by the second level. This algorithm is based on slicing the inventory profiles along the scheduling horizon at the given capacity of the individual tanks, and generating subprofiles to determine the points in time that a tank is free to be reused.

Li and Ierapetritou[17] developed a bilevel decomposition algorithm to solve separately the production planning and scheduling levels. The planning level is modeled using discrete-time representation while the scheduling subproblems (one per planning period) are formulated as continuous-time models. The algorithm aims to close the difference between the objective function of both levels. At the planning level, an underestimation term of the production costs associated with the scheduling level is included and computed through Lagrangian relaxation. At the end of each iteration, the algorithm contracts the bounds of the planning decision variables.

Mouret et al.[18] introduced a new algorithm to solve a MINLP model for refinery planning and scheduling of crude-oil operations using Lagrangian decomposition, as well as a new hybrid algorithm to solve the associated dual problem which combines subgradient and cutting plane methods at different steps. The authors pointed out that, in this case, the planning and scheduling problems are only linked by the crude distillation unit (CDU) feedstock quantities; that is, the planning model is not an aggregated formulation of the scheduling problem. For that reason, a spatial Lagrangian decomposition is a better option to solve the problem than a hierarchical approach.

For gasoline blend planning and scheduling, Glismann and Gruhn[19] used a two-level approach: at the top level, a discrete-time NLP model computes blend recipes and, at the lower level, a discrete-time MILP model solves the short-term scheduling problem. In this approach, the time periods of the NLP model are defined by product liftings and other specific planning priorities while the scheduling MILP model time periods are defined to be 2-hours long. If a feasible solution cannot be found at the lower level, or if deviations from the goals determined at the top level cannot be accepted, blend recipes are recomputed by NLP model but this time including the information from MILP solution through the addition of constraints regarding the blend components consumption (this step is not clearly described in their article). The new blend recipes can be chosen as alternatives to the previous ones at the MILP scheduling model which contains constraints to enforce a minimum running time for a single recipe on a blender.

The work presented in this article introduces a new method to solve the gasoline blend planning and scheduling problem, using the inventory pinch concept to reduce the number of different blend recipes. It is based on the gasoline blend planning decomposition shown in Part I of this article. We use a three-level decomposition:

1. At the first level, blend recipes are optimized by solving a discrete-time multiperiod NLP model. The boundaries of the time periods in this NLP model are initially given by the inventory pinch points and break points (if applicable) in the components' qualities or unit costs.

2. At the second level, a blend plan using the recipes from the first level is computed. The blend plan defines the swing tanks allocation to each product and the volumes to produce in each blender in each second-level time period along the horizon. A discrete-time multiperiod MILP model is solved for the entire horizon in two phases:

a. Blend planning. The objective function minimizes the blend cost and inventory infeasibilities. Hence, this phase uses the blend recipes from the first level to compute an optimal blend plan.

b. Approximate scheduling. The objective function minimizes the number of blend runs and the number of product transitions in the blenders and swing tanks. It is assumed that the length of the time periods at the second level is such that a swing tank can only be used for one product during any second-level time period. As the blend recipes and inventory levels from the first level have been proven feasible in the previous phase, they are fixed and the blend cost is not included in the objective function.

3. At the third level, scheduling of the blending operations is carried out based on the decisions from the second level. Production and delivery rates, as well as start and end times of the tasks are computed. Product-tank allocation is fixed as computed at the second level. At the third level, the scheduling horizon is divided in several time intervals, and each one is solved using a discrete-time multiperiod MILP. These small subintervals are solved in two sequences: a forward sequence that computes an initial solution considering the initial conditions of the system, and a reverse sequence that merges blend runs, if possible, to obtain a final production schedule with smaller number of switches. Each sequence is solved in two phases:

a. Feasibility phase. The objective function minimizes the blend cost and inventory infeasibilities. During the forward sequence, this phase determines if the blend recipes from the first level and the blend plan from the second level can provide a feasible schedule; while during the reverse sequence, this phase determines the minimum number of blend runs that can be achieved.

b. Optimization phase. The objective function minimizes the number of blend runs, penalizes long blend runs, variations in the delivery rates, late deliveries, and variations in the destination tank for the product of a blend run.

.If the solutions at the second level or at the third level forward sequence contain slack variables with nonzero values, the current set of blend recipes from the first level leads to infeasible solutions. In such a case, we subdivide the corresponding period at the first-level NLP model and resolve all levels. Hence, we increase the number of time periods (and the number of corresponding recipes) only when such a change is required to ensure the optimality and feasibility of the solution.

The contents of this article are organized as follows. We start with the problem description, and then, the proposed solution approach is explained. The mathematical models and solution algorithm are presented next, followed by the numerical case studies and by the comparison of our solutions with those from the literature. Finally, we conclude with the discussion of the results, summary of the algorithm performance, and outline of the future work.

Problem Statement

The integrated gasoline blend planning and scheduling problem addressed in this work is stated as follows:

Given

1. A scheduling horizon $[0, H]$;
2. A set of blend components and profiles of their quality levels along the horizon;
3. A set of tanks to store the blend components, their initial inventories, limits on their holdups, and the flow profiles of feeds into the tanks;
4. A set of products and their quality and composition specification limits;
5. A set of blenders, the products that each blender can process, minimum running times of these blenders for a specific product, and limits on their blending rates;
6. A set of tanks to store the products, the products that each tank can store, limits on their holdups, the products and inventories at time zero, and their maximum delivery rates;
7. A set of orders, their constituent products, amounts, and delivery time windows; and
8. Unit cost of the blend components.

Determine

1. The blend recipes (i.e., the volume fractions of the blend components that compound one unit of each product);
2. The blenders that each component tank should feed over time, and their feed rates;
3. The products that each blender should produce over time, and their production rates;
4. The products that each product tank should receive over time, from which blender, and at what flow rates;
5. The orders that each product tank should deliver over time, their amounts, and delivery rates; and
6. The inventory profiles of component and product tanks.

Minimizing

The operating cost that includes the blended materials cost and the cost associated with the number of blend runs and product transitions in the blenders and storage tanks.

Subject to the following constraints

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 (industrial practice).

Assuming

1. Flow rate profile of each component from the upstream process is piecewise constant;
2. Component quality profile is also piecewise constant;
3. There is only one tank for a given blend component;
4. Mixing in each blender is perfect;
5. Changeover times between products are negligible for product tanks;
6. Changeover times between product runs on blenders are product-dependent but sequence-independent;
7. Each order involves only one product (one original order involving different products can be broken into orders of each specific product); and
8. Each order is completed during the scheduling horizon.

Allowing

1. A component tank may receive and feed components at the same time;
2. A component tank may feed some or all blenders simultaneously;
3. Multiple component tanks may feed a blender at the same time;
4. A product tank may receive and feed products at the same time;
5. A product tank may deliver multiple orders at the same time; and
6. Multiple product tanks may deliver an order at the same time.

Solution Approach

A discrete-time formulation is adopted in our models as it leads to simple time-related structure of the equations when compared to the continuous-time representation. As previously stated, one of the main advantages of the discrete-time formulation is the linearity of the terms involving flow rates and time intervals. Although a discrete-time model represents an approximation of the real-life problem,[11] our goal is to obtain close-to-optimal solutions for large-scale problems with short execution times. For comparison purposes, some continuous-time model examples from the literature were solved using our discrete-time approach.

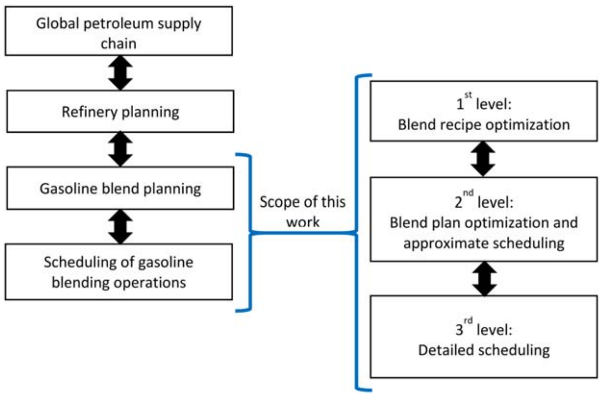
Global petroleum supply chain
Refinery planning
Gasoline blend planning
Scheduling of gasoline blending operations
Scope of this work
1st level: Blend recipe optimization
2nd level: Blend plan optimization and approximate scheduling
3rd level: Detailed scheduling

Figure 1. Proposed decomposition of the gasoline blend planning and scheduling problem.

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

We use the hierarchical framework shown in Figure 1 and our decomposition approach is described in Figure 2. The first level optimizes the blend recipes; the second level uses the blend recipes from the first level to determine how much, when and in which blender a product should be processed, and the allocation of swing tanks to specific products along the horizon; finally, the third level computes the delivery and production rates, and the start and end times of all tasks that minimize the number of blend runs, using the decisions from the second level. We assume that the production rates of the blend components are given by the solution of the refinery planning level; therefore, the inventory cost does not need to be included at any of these three levels as the refinery will carry the inventories as either blend components or as finished gasoline grades. The first and second levels are a decomposition of the gasoline blend planning problem. By solving the recipe optimization separately, nonlinear models can be solved more efficiently since the problem is modeled as a NLP instead of a MINLP, as illustrated in Part I of this work.

To arrive at short execution times, a temporal decomposition technique can be applied to solve the third level (i.e., the detailed scheduling model). As the second level is solved for the entire scheduling horizon, we know that the constraints imposed by the second-level solution constrain the third-level scheduling problem close to the optimal solution (if the constraints of the second level contain a feasible schedule). Hence, we have decided to use a forward and reverse rolling window techniques to compute the schedules at different steps of the algorithm. The scheduling horizon is divided in small subintervals, denoted as *L*-intervals, which represent the width of the rolling window. The forward rolling window sequence is used to generate an initial solution that takes into consideration the conditions at the beginning of the horizon. Once the initial solution is computed, the reverse rolling window sequence is used to determine if the number of blend runs can be reduced.

Inventory pinch concept

The inventory pinch concept used in this work is defined in detail in Castillo et al.,[20] and a brief review can be found

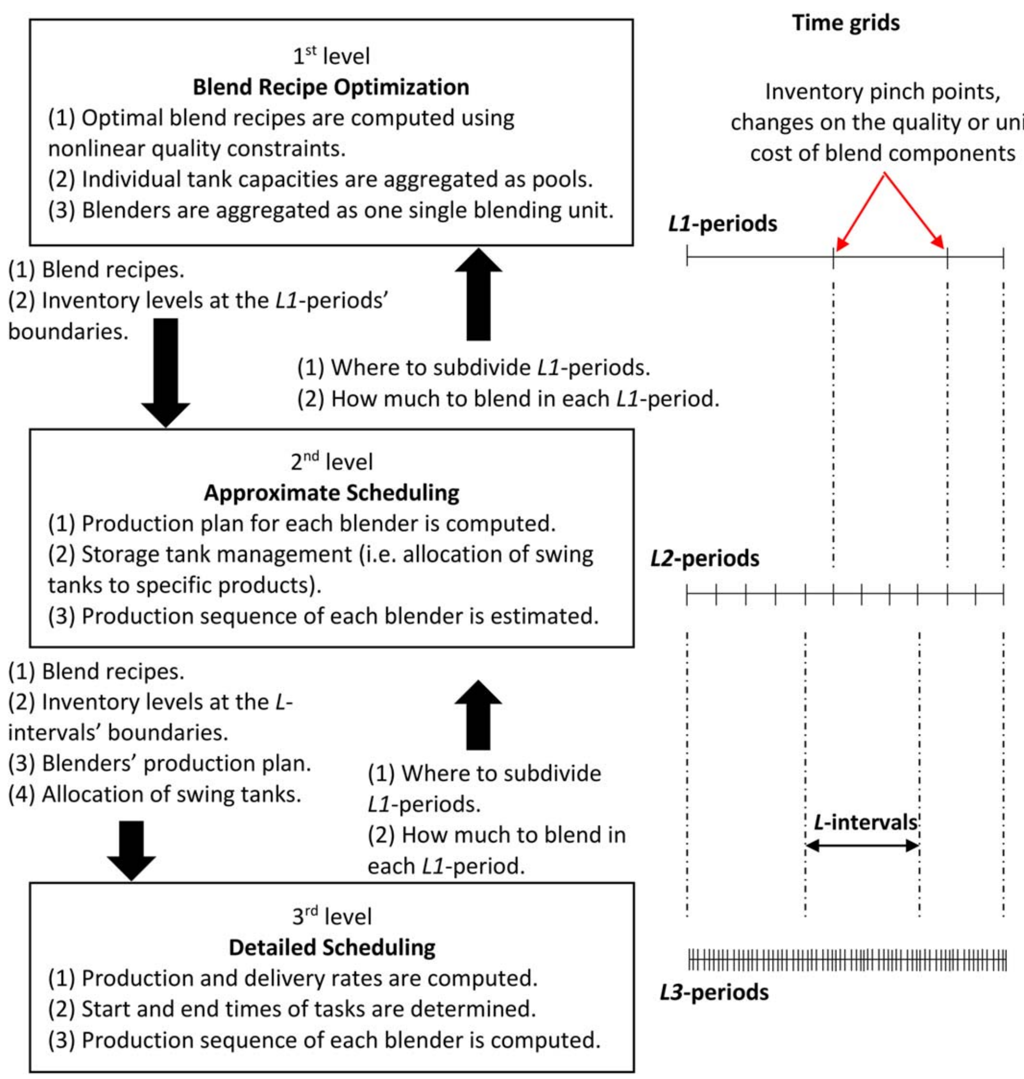
Time grids
1st level
Blend Recipe Optimization
(1) Optimal blend recipes are computed using nonlinear quality constraints.
(2) Individual tank capacities are aggregated as pools.
(3) Blenders are aggregated as one single blending unit.
(1) Blend recipes.
(2) Inventory levels at the L1-periods' boundaries.
(1) Where to subdivide L1-periods.
(2) How much to blend in each L1-period.
2nd level
Approximate Scheduling
(1) Production plan for each blender is computed.
(2) Storage tank management (i.e. allocation of swing tanks to specific products).
(3) Production sequence of each blender is estimated.
(1) Blend recipes.
(2) Inventory levels at the L-intervals' boundaries.
(3) Blenders' production plan.
(4) Allocation of swing tanks.
(1) Where to subdivide L1-periods.
(2) How much to blend in each L1-period.
3rd level
Detailed Scheduling
(1) Production and delivery rates are computed.
(2) Start and end times of tasks are determined.
(3) Production sequence of each blender is computed.
Inventory pinch points, changes on the quality or unit cost of blend components
L1-periods
L2-periods
L-intervals
L3-periods

Figure 2. Inventory pinch-based algorithm for gasoline blend scheduling.

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

in Part I of this article. The inventory pinch points are the times where the cumulative average total production (CATP) curve changes its slope, in order to remain above the cumulative total demand (CTD) curve, the minimum possible number of times, and at the closest distance to the CTD. These cumulative curves are also known as composite curves.[21] $V(0)$ represents the initial inventory available. The concept can be applied directly to multiple products (i.e., the demand of all products is aggregated in such case), and minimum inventory limits and target inventories can be incorporated easily. The inventory pinch points can be seen as well in the grand composite curve (i.e., CATP–CTD values) as those points in time where the inventory goes to zero. The cumulative curves provide the minimum production quantities required to meet the demand in each interval where the CATP has a constant slope; however, the grand composite curve given by CATP–CTD does not represent the actual total inventory profile nor the slopes of the CATP curve are the actual production rates; those values will be calculated through our Multiperiod Inventory Pinch (MPIP) algorithm.

The inventory pinch points define the times within the planning/scheduling horizon where the product inventories are at the minimum allowed limits, and they are used to define the time grids of each level.

Time grids construction of each level

At the first level, the boundaries of the time periods are initially delimited by the inventory pinch points, the times when the quality of blend components changes, and the times when the unit cost of blend components or products vary.

At the second level, the boundaries of the time periods are defined by the planner taking into consideration the following:

• The boundaries of the first level. All boundaries of the first level must exist as well at the second level.

• The minimum time a storage tank will be holding a specific product. The smaller the time periods at the second level, the better the time resolution of assigning the swing tanks to specific products. In our case studies, we use one-day periods at the second level, but it is possible to use 1/2 or 1/4 of the day as the duration of these time periods as dictated by operational considerations.

• The minimum time that a blending unit will require to produce the minimum threshold amount. In this case, larger

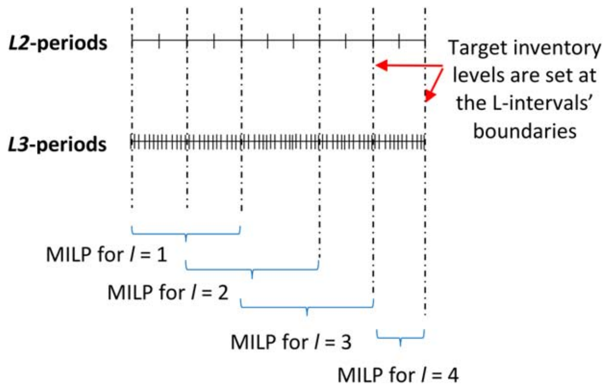

Figure 3. *L*-intervals at the third level.

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

periods are preferred to avoid inventory infeasibilities (i.e., slack variables with nonzero values) due to the blending units not being able to fulfill the minimum production threshold constraint.

- The delivery windows. The higher the number of the period boundaries that correspond to the start and end times of the delivery windows, the fewer iterations the algorithm will require. The second-level periods are still aggregated periods; all tasks are assumed to be completed at the end boundary. Therefore, inventory infeasibilities may appear if a delivery window ends before the conclusion of the second-level period. As a heuristic rule, an original delivery window is narrowed when it spans less than the half of a second-level period, if and only if the order is possible to be met at maximum order delivery rate within the reduced window; otherwise, another second-level period should be considered.

At the third level, the time periods are small enough that only one task in a given unit can take place (e.g., a blender can only blend one specific gasoline grade). They are usually 1- or 2-hours long.

The length of the time periods at any level does not need to be uniform, and their boundaries are not required to coincide with the start of a calendar day, month and so forth. For example, it is possible to use a few smaller periods at the second level if it is known that no blender will operate at some intervals of the horizon (e.g., after the last delivery window if it ends before the conclusion of the horizon). For sake of exposition, the time periods of the first, second, and third level, are denoted as L1-periods, L2-periods, and L3-periods, respectively. One L1-period contains one or more L2-periods, and one L2-period contains several L3-periods; therefore, the product demand, blend component supply, and blend capacity at any level are the aggregated values of the corresponding periods of the next lower level.

Mathematical Models

The mathematical models are presented next. The following sets are used:

A =	$\{(\alpha)$ | set of different supply flow rates of blend components$\}$
Bl =	$\{(\text{bl})$ | set of blenders$\}$
E =	$\{(e)$ | set of quality properties$\}$
G =	$\{(g)$ | set of time slots for blend run allocation$\}$
I =	$\{(i)$ | set of blend components$\}$
J =	$\{(j)$ | set of product storage tanks$\}$
K =	$\{(k)$ | set of time periods at the first level or L1-periods$\}$
L =	$\{(l)$ | set of time intervals in which the scheduling horizon is solved$\}$
M =	$\{(m)$ | set of time periods at the second level or L2-periods$\}$
N =	$\{(n)$ | set of time periods at the third level or L3-periods$\}$
O =	$\{(o)$ | set of orders$\}$
P =	$\{(p)$ | set of products$\}$
BJ =	$\{(\text{bl}, j)$ | blender bl can feed tank $j\}$
BP =	$\{(\text{bl}, p)$ | blender bl can process product $p\}$
JP =	$\{(j, p)$ | tank j can hold product $p\}$
MK =	$\{(m, k)$ | L2-periods contained in each L1-period$\}$
NA =	$\{(n, \alpha)$ | blend component supply profile a occurs within L3-period $n\}$
OP =	$\{(o, p)$ | order o consists of product $p\}$
JO =	$\{(j, o)$ | tank j can deliver order $o\}$
GM =	$\{(g, m)$ | slots g for blend allocation is contained in period $m\}$
ML =	$\{(m, l)$ | L2-periods contained in interval $l\}$
MLE =	$\{(n, m)$ | last L2-period contained in interval $l\}$
NM =	$\{(n, m)$ | L3-periods contained in each L2-period$\}$
NMF =	$\{(n, m)$ | all L3-periods, except the first one, contained in each L2-period$\}$
NL =	$\{(n, l)$ | L3-periods contained in interval $l\}$
NLE =	$\{(n, l)$ | last L3-period contained in interval $l\}$
NLF =	$\{(n, l)$ | all L3-periods, except the first one, contained in interval $l\}$
NLO =	$\{(n, l, o)$ | L3-periods contained in interval l when order o can be delivered$\}$
JON =	$\{(j, o, n)$ | tank j may deliver order o during L3-period $n\}$
JPN =	$\{(j, p, n)$ | L3-periods when tank j can store product $p\}$
NLOA =	$\{(n, l, o)$ | L3-period contained in interval l when order o can start to be delivered$\}$

In principle, the models presented here can be adapted to any scheduling problem with similar characteristics. Numbering of the equations continues from that of Part I of this article.

First level—Blend recipe optimization

The first-level objective is to minimize the blend cost by determining the optimum volume fractions (i.e., blend recipes) to mix the blend components available into final products that meet quality specifications and demand requirements. At this level, the values of the product demand, blend component supply, and blend capacity are aggregated values for each of the L1-periods, and product storage tanks are aggregated into product pools. The mathematical model of the first level presented in Part I of this article is used without modifications. The objective function defined by Eq. 1 minimizes the blend cost (Eq. 2) and the inventory infeasibilities. The penalty coefficients for the product slacks variables are much greater than the unit cost coefficients of the blend components. Slack variables will be zero at the optimal solution (i.e., penalty coefficients will not affect the final blend cost); if they have nonzero values, it means that the problem is infeasible because there is not enough blend components to blend products as required by the quality specifications or demand requirements. Inventory cost is not included in the objective function as inventory levels will be at the minimum allowed at the end of each L1-period (because the CATP curve touches the CTD curve).

In the first iteration of the algorithm, the volumes to be blended in each L1-period are the minimum amount required to fulfill the demand; thus, the solution of the first level model is a lower bound of the global blend cost. Discrete variables are not required at the first level because compliance of minimum blend size thresholds and maximum blenders' capacity constraints are handled by adjusting the volumes to be blended in each L1-period by moving the minimum possible amount of volume to the previous L1-periods.

Equations 3–15 only appear in Part I of this article

$$\min Z_{\mathrm{L1}} = \mathrm{BlendCost}_{\mathrm{L1}} + \sum_{k\in K} \left\{ \begin{array}{l} \sum_{i\in I} \mathrm{Penalty}_{\mathrm{bc,L1}} \cdot \left(S^{+}_{\mathrm{bc,L1}}(i,k) + S^{-}_{\mathrm{bc,L1}}(i,k) \right) \\ + \sum_{p\in P} \mathrm{Penalty}_{\mathrm{pool,L1}} \cdot \left(S^{+}_{\mathrm{pool,L1}}(p,k) + S^{-}_{\mathrm{pool,L1}}(p,k) \right) \end{array} \right\} \tag{1}$$

$$\mathrm{BlendCost}_{\mathrm{L1}} = \sum_{k\in K} \left(\sum_{i\in I, p\in P} \mathrm{Cost}_{\mathrm{bc}}(i) \cdot V_{\mathrm{comp,L1}}(i,p,k) \right) \tag{2}$$

Second level—Blend plan optimization and approximate scheduling

The blend plan consists on determining (1) how much to blend of each product and in which blender in each L2-period; (2) allocation of swing tanks to specific products in each L2-period; and (3) the inventory profiles of all storage tanks along the planning horizon.

The second level computes the optimal blend plan using the blend recipes from the first level. The blend recipes of each L1-period are fixed in the corresponding L2-periods. The second level is basically a disaggregation step of the first- level decisions. A MILP model is used to deal with constraints as the minimum blend size threshold, and others that require discrete variables. The second level is solved in two phases: blend planning optimization followed by approximate scheduling. In this way, inventory infeasibilities are detected in less time.

Blend planning optimization at the second level minimizes the blend cost and the inventory slack variables (see Eq. 16). If a feasible operation can be obtained using the blend recipes from the first level, inventory slack variables will be zero at the solution of the second level; otherwise, the inventory slacks will show which specific products, by how much, and in which L2-periods they cannot be produced in the amounts required. To ensure that the nonzero slacks will appear on the product tanks, the product inventories at the L1-period boundaries are fixed instead of setting the production targets from the first level, and:

1. Penalty coefficients for the products' inventory slack variables are smaller in comparison with the penalty for the components' inventory slacks (i.e., $\mathrm{Penalty}_{\mathrm{bc,L2}} >> \mathrm{Penalty}_{\mathrm{pr,L2}}(m)\ \forall\ m$; $\mathrm{Penalty}_{\mathrm{pool,L2}}(m) = \mathrm{Penalty}_{\mathrm{pr,L2}}(m)\ \forall\ m$).

2. The penalty coefficients for the product inventory slacks decrease with time (i.e., $\mathrm{Penalty}_{\mathrm{pr,L2}}(m) > \mathrm{Penalty}_{\mathrm{pr,L2}}(m+1)\ \ \forall\ \ m$) in order to move the inventory infeasibilities as late as possible in the planning horizon In this way, the use of a given blend recipe is maximized. The penalty coefficients must decrease as fast as possible and after each L1-period boundary a significant change must take place.

When the solution of this phase has inventory infeasibilities (i.e., component supply or blender constraints are such that the recipes computed at the first level are not feasible within a L1-period), the algorithm will subdivide such L1-periods and reoptimize the blend recipes.

Given the assumption that the flow rates of blend components are given by the solution of the refinery planning level, inventory cost is not included at this level as the refinery will carry the total inventories as either blend components or finished gasoline grades

$$\min Z^{\mathrm{feas}}_{\mathrm{L2}} = \mathrm{BlendCost}_{\mathrm{L2}} + \sum_{m\in M} \left\{ \begin{array}{l} \sum_{i\in I} \mathrm{Penalty}_{\mathrm{bc,L2}}(m) \cdot \left(S^{+}_{\mathrm{bc,L2}}(i,m) + S^{-}_{\mathrm{bc,L2}}(i,m) \right) \\ + \sum_{p\in P} \mathrm{Penalty}_{\mathrm{pool,L2}}(m) \cdot \left(S^{+}_{\mathrm{pool,L2}}(p,m) + S^{-}_{\mathrm{pool,L2}}(p,m) \right) \\ + \sum_{j\in J} \mathrm{Penalty}_{\mathrm{pr,L2}}(m) \cdot \left(S^{+}_{\mathrm{pr,L2}}(j,m) + S^{-}_{\mathrm{pr,L2}}(j,m) \right) \end{array} \right\} \tag{16}$$

Equation 17 computes the blend cost

$$\mathrm{BlendCost}_{\mathrm{L2}} = \sum_{m\in M} \left(\sum_{(\mathrm{bl},p)\in BP} \sum_{i\in I} \mathrm{Cost}_{\mathrm{bc}}(i) \cdot V_{\mathrm{comp,L2}}(i,p,\mathrm{bl},m) \right) \tag{17}$$

Approximate scheduling at the second level minimizes the number of product transitions in the blenders and in the product storage tanks, as well as the number of blend runs (see Eq. 18). Binary variable $x_{\mathrm{L2}}(p,\mathrm{bl},m)$ defines a blend run; that is, it determines if product p is going to be produced in blender bl during period m if its value is 1. Blend runs are penalized because a solution at the second level can suggest to blend the same product in the same blender for several adjacent L2-periods, thus, not incurring in a penalty for product changeover in the blender; however, they are not likely to have the same blending rate, which is a constraint at the third level to define a blend run. Therefore, it is better to have the minimum number of blend runs, and then reduce the expected number of product changeovers (i.e., $\mathrm{PenaltyBR}_{\mathrm{L2}}(\mathrm{bl}) >> \mathrm{PenaltyBS}_{\mathrm{L2}}$). If the inventory slacks are zero at the blend planning solution, we know that the blend recipes and inventory targets from the first-level solution can yield a feasible blend plan; then, if those are fixed at the second level, the blend cost does not need to be included in Eq. 18.

Equations 19–66 only appear in Part I of this article

$$\min Z^{\mathrm{opt}}_{\mathrm{L2}} = \sum_{m\in M} \left\{ \begin{array}{l} \sum_{(\mathrm{bl},p)\in \mathrm{BP}} \mathrm{PenaltyBR}_{\mathrm{L2}}(\mathrm{bl}) \cdot x_{\mathrm{L2}}(p,\mathrm{bl},m) \\ + \sum_{j\in J} \mathrm{PenaltyTS}_{\mathrm{L2}}(j) \cdot ue_{\mathrm{L2}}(j,m) \end{array} \right\} + \sum_{g\in G} \left(\sum_{\mathrm{bl}\in \mathrm{Bl}} \mathrm{PenaltyBS}_{\mathrm{L2}} \cdot xe_{\mathrm{L2}}(\mathrm{bl},g) \right) \tag{18}$$

Third level—Detailed scheduling

The decisions of the second level are now disaggregated at the third level. The goals of the third level are to determine:

1. The blending rates for each blend run;
2. The delivery rates from all tanks to each order along the scheduling horizon;
3. The production sequence in each blender; and
4. Start and end times of all tasks.

We use the blend recipes from the first level, and the inventory levels, the swing tanks allocation, the production of each blender, and the delivery plan from the second level to reduce the search space and model size at the third level.

The scheduling horizon is divided in various subintervals, denoted as *L*-intervals (see Figure 3), and a MILP model is solved for each one of them. The boundaries of the *L*-intervals must be synchronized with the boundaries of some L2-periods to enable the inventory levels computed at the second level to be fixed at the start and end boundaries of the *L*-intervals. The *L*-intervals are solved first in a forward sequence, then the length of these *L*-intervals is increased and they are solved in a reverse sequence. During the forward pass, *L*-intervals do not need to overlap as the only goal is to check if the second- level solution is feasible or not. During the reverse pass, *L*-intervals may overlap to reduce the number of blend runs while solving relatively small size models.

The reverse pass decreases the number of blend runs by merging the same-product blends that are adjacent to the boundaries of the *L*-intervals used during the forward pass. Due to our model formulation that uses the start of the blend runs to count them, as well as the fulfillment of some constraints (e.g., minimum running time and minimum production volume of a blend run), the feasibility of the *L*-interval being solved with respect to the rest of the scheduling horizon is easier to attain using a reverse pass than a second forward pass.

The third level (at each forward and reverse pass) is solved in two phases: a feasibility phase followed by an optimization phase.

Objective Function of the Third-Level Feasibility Phase. The objective function is given by Eq. 67; it minimizes the blend cost, the inventory infeasibilities, and the delivery infeasibilities. As the third level uses the inventory levels computed by the second level, the inventory cost does not need to be included at this level. The blend cost is computed by Eq. 68. Inventory slack variables will be zero at the solution of the third level if a feasible operation can be obtained using the blend recipes computed at the first level and the blend plan from the second level. If this is not the case, the inventory slacks will show which specific products, by how much, and in which L3-periods they cannot be produced in the amounts required to meet the demand. Once again, to have the nonzero slacks on the product tanks, inventory levels from the second level are fixed instead of the production targets, and:

1. Penalty coefficients for the products' inventory slack variables are smaller in comparison with the penalty for the components' inventory slacks (i.e., $\text{Penalty}_{\text{bc,L3}}(n) >> \text{Penalty}_{\text{pr,L3}}(n)\ \forall\ n$), and they decrease with time (i.e., $\text{Penalty}_{\text{bc,L3}}(n) >> \text{Penalty}_{\text{bc,L3}}(n+1)\ \forall\ n$). Component slacks appear when component tanks overflow.
2. The penalty coefficients for the product inventory slacks decrease with time (i.e., $\text{Penalty}_{\text{pr,L3}}(n) > \text{Penalty}_{\text{pr,L3}}(n+1)\ \forall\ n$) to move the inventory infeasibilities as late as possible in the planning horizon.
3. Penalty coefficients for the delivery slacks are higher than those of the component and product inventory slacks (i.e., $\text{PenaltyD}_{\text{pr,L3}} >> \text{Penalty}_{\text{bc,L3}}(n) >> \text{Penalty}_{\text{pr,L3}}(n)\ \forall\ n$).

If the solution of the MILP has inventory infeasibilities, it indicates that component supply or blender constraints are such that the recipes computed at the first level and/or the blend plan from the second level are not feasible within a L2-period. The algorithm will subdivide the corresponding L1-period and reoptimize the blend recipes

$$\min Z_{\text{L3}}^{\text{feas}} = \text{BlendCost}_{\text{L3}} + \sum_{o\in O} \text{PenaltyD}_{\text{pr,L3}} \cdot \left(S_{\text{order,L3}}^{+}(o) + S_{\text{order,L3}}^{-}(o)\right) + \sum_{n\in N} \left\{ \begin{array}{l} \sum_{i\in I} \text{Penalty}_{\text{bc,L3}}(n) \cdot \left(S_{\text{bc,L3}}^{+}(i,n) + S_{\text{bc,L3}}^{-}(i,n)\right) \\ + \sum_{j\in J} \text{Penalty}_{\text{pr,L3}}(n) \cdot \left(S_{\text{pr,L3}}^{+}(j,n) + S_{\text{pr,L3}}^{-}(j,n)\right) \end{array} \right\} \tag{67}$$

$$\text{BlendCost}_{\text{L3}} = \sum_{n\in N} \sum_{\text{bl}\in \text{Bl}} \sum_{p\in P} \sum_{i\in I} \text{Cost}_{\text{bc}}(i) \cdot V_{\text{comp,L3}}(i,p,\text{bl},n) \tag{68}$$

Trying to solve the entire scheduling horizon at once will result in a third-level model with a large number of equations and discrete variables which may be computationally inefficient to solve. However, the scheduling horizon can be divided in smaller time intervals denoted as *L*-intervals and Eqs. 67 and 68 are replaced by Eqs. 69 and 70, respectively

$$\min Z_{\text{L3}}^{\text{feas}}(l) = \text{BlendCost}_{\text{L3}}(l) + \sum_{o\in O} \text{PenaltyD}_{\text{pr,L3}} \cdot \left(S_{\text{order,L3}}^{+}(o,l) + S_{\text{order,L3}}^{-}(o,l)\right) + \sum_{n\in \text{NL}} \left\{ \begin{array}{l} \sum_{i\in I} \text{Penalty}_{\text{bc,L3}}(n) \cdot \left(S_{\text{bc,L3}}^{+}(i,n) + S_{\text{bc,L3}}^{-}(i,n)\right) \\ + \sum_{j\in J} \text{Penalty}_{\text{pr,L3}}(n) \cdot \left(S_{\text{pr,L3}}^{+}(j,n) + S_{\text{pr,L3}}^{-}(j,n)\right) \end{array} \right\} \tag{69}$$

$$\text{BlendCost}_{\text{L3}}(l) = \sum_{n\in \text{NL}} \sum_{\text{bl}\in \text{Bl}} \sum_{p\in P} \sum_{i\in I} \text{Cost}_{\text{bc}}(i) \cdot V_{\text{comp,L3}}(i,p,\text{bl},n) \tag{70}$$

Notice that, if no inventory infeasibilities appear, we have

$$\min Z_{\text{L3}}^{\text{feas}} = \sum_{l} \min Z_{\text{L3}}^{\text{feas}}(l)$$

And $\text{BlendCost}_{\text{L3}} = \sum_{l} \text{BlendCost}_{\text{L3}}(l)$.

Objective Function of the Third-Level Optimization Phase. After a feasible schedule is generated (i.e., all slack variables have a value of zero at the solution of the feasibility phase of the third level), the third level minimizes the number of blend runs, penalizes variations in the delivery rates, punish late deliveries, and reduces the number of tanks receiving product from the same blend run (reduction along the duration of the blend run as a blender only feeds one tank at a time). Equation 71 is the objective function for the optimization phase where the first term penalizes the number of blend runs, the second term penalizes irregular delivery rates, the third term penalizes late deliveries (i.e., $\text{PenaltyDpm}_{\text{L3}}(n) > \text{PenaltyDpm}_{\text{L3}}(n-1)\ \forall\ n$), the fourth term represents the penalty associated with long blend runs, and the last term is the penalty for sending product from one blender to different storage tanks during the same blend run but at different times (the blenders can only send product to only one tank at a time). In this work, the higher penalty corresponds to the number of blend runs. Equation 71 does not minimizes the blend cost because the blend recipes and the inventory levels at the boundaries of the L-interval are fixed (i.e., blend cost will be the same as that computed during the feasibility phase)

$$\min Z_{\text{L3}}^{\text{opt}}(l) = \sum_{n\in\text{NL}} \left\{ \begin{array}{l} \sum_{\text{bl}\in\text{Bl}} \text{PenaltyBsw}_{\text{L3}} \cdot sw_{\text{L3}}(\text{bl},n) \\ + \sum_{(j,o)\in\text{JON}} \text{PenaltyDsw}_{\text{L3}} \cdot \text{Dsw}_{\text{L3}}(j,o,n) \\ + \sum_{(j,o)\in\text{JON}} \text{PenaltyDpm}_{\text{L3}}(n) \cdot \text{Del}_{\text{pr,L3}}(j,o,n) \\ + \sum_{\text{bl}\in\text{Bl}} \text{PenaltyBL}_{\text{L3}} \cdot t_{\text{blend,L3}}(bl,n) \\ + \sum_{j\in J} \text{PenaltyTsw}_{\text{L3}} \cdot \text{ve}_{\text{L3}}(j,n) \end{array} \right\} \tag{71}$$

Material Balance on Blend Component Tanks. The volumetric balance on the blend components ensures that the initial inventory plus the supply are equal to the final inventory plus the amount transferred to the blenders, during a L3-period. At this level, the L3-periods are small enough that the supply rate of blend components is constant within such periods. Equation 72a is used during the feasibility phase to include the slack variables. If the blend recipes lead to a feasible solution, then the slack variables have a value of zero at the feasibility phase solution and they can be omitted during the optimization phase (Eq. 72b)

$$\begin{gathered} F_{\text{bc}}(i,\alpha) \cdot t_{\text{L3}}(n) + V_{\text{bc,L3}}(i,n-1) - V_{\text{bc,L3}}(i,n) \\ - \sum_{(\text{bl},p)\in\text{BP}} V_{\text{comp,L3}}(i,p,\text{bl},n) \\ + S_{\text{bc,L3}}^{+}(i,n) - S_{\text{bc,L3}}^{-}(i,n) = 0 \\ \forall\ i,n \in \text{NL}, a \in \text{NA} \end{gathered} \tag{72a}$$

$$\begin{gathered} F_{\text{bc}}(i,\alpha) \cdot t_{\text{L3}}(n) + V_{\text{bc,L3}}(i,n-1) \\ - V_{\text{bc,L3}}(i,n) - \sum_{(\text{bl},p)\in\text{BP}} V_{\text{comp,L3}}(i,p,\text{bl},n) = 0 \\ \forall i,n \in \text{NL}, a \in \text{NA} \end{gathered} \tag{72b}$$

Fixed Blend Recipe. Equation 73 fixes the blend recipe $r(i,p,k)$ from the first level in its corresponding L3-periods. Note that $r(i,p,k)$ is a parameter and not a variable at the third level. $F_{\text{blend,L3}}(p,\text{bl},n)$ is the production rate of product p in blender bl during period n

$$\begin{gathered} V_{\text{comp,L3}}(i,p,\text{bl},n) = r(i,p,k) \cdot t_{\text{L3}}(n) \cdot F_{\text{blend,L3}}(p,\text{bl},n) \\ \forall\ i,(p,\ \text{bl}) \in \text{BP}, (n,m) \in \text{NM}, n\ \in \text{NL} \end{gathered} \tag{73}$$

Blender Constraints. Equation 74 establishes that a blender may only blend a product according to the second level blend plan. Equation 75 constrains the blender to process only one product during a L3-period. $x_{\text{L3}}(p,\text{bl},n)$ is a 0–1 continuous variable which value is 1 if product p is blended in bl during period n, and 0 otherwise. Note that $x_{\text{L2}}(p,\text{bl},m)$ is a parameter and not a variable at the third level. $w_{\text{blend,L3}}(\text{bl},n)$ is a 0–1 continuous variable which can only take the value equal to 1 if the blender is idle, or equal to 0 if it is running. All 0–1 continuous variables are set to be less than or equal to 1 (equations omitted here)

$$x_{\text{L3}}(p,\text{bl},n) \le \sum_{m\in l} x_{\text{L2}}(p,\text{bl},m) \quad \forall\ (\text{bl},p) \in \text{BP}, (n,m) \in \text{NM}, n \in \text{NL} \tag{74}$$

$$w_{\text{blend,L3}}(\text{bl},n) + \sum_{p\in\text{BP}} x_{\text{L3}}(p,\text{bl},n) = 1 \quad \forall \text{bl}, n \in \text{NL} \tag{75}$$

Equations 76 and 77 observe that the production rate must be equal to or less than the maximum blending rate, and equal to or greater than the minimum blending rate, respectively

$$F_{\text{blend,L3}}(p,\text{bl},n) \le F_{\text{blend}}^{\max}(\text{bl}) \cdot x_{\text{L3}}(p,\text{bl},n) \quad \forall\ (\text{bl},p) \in \text{BP}, n \in \text{NL} \tag{76}$$

$$F_{\text{blend,L3}}(p,\text{bl},n) \ge F_{\text{blend}}^{\min}(\text{bl}) \cdot x_{\text{L3}}(p,\text{bl},n) \quad \forall\ (\text{bl},p) \in \text{BP}, n \in \text{NL} \tag{77}$$

Volume Transferred from Blenders to Product Tanks. The set JPN is constructed according to the solution of the second-level model; this enables the product allocation of swing tanks determined at the second level to be fixed at the third level. Equations 78–81 force a blender to feed at most only one product tank during a L3-period. Binary variable $v_{\text{L3}}(j,\text{bl},n)$ specifies that blender bl is feeding product tank j during period n if its value is equal to 1

$$\begin{gathered} V_{\text{trans,L3}}(j,\text{bl},n) \le F_{\text{blend,L3}}(p,\text{bl},n) \cdot t_{\text{L3}}(n) + F_{\text{blend}}^{\max}(\text{bl}) \\ \cdot t_{\text{L3}}(n) \cdot (1 - v_{\text{L3}}(j,\text{bl},n)) \\ \forall(\text{bl},p) \in \text{BP},\ (\text{bl},j) \in \text{BJ},\ (j,p) \in \text{JPN}, n \in \text{NL} \end{gathered} \tag{78}$$

$$\begin{gathered} V_{\text{trans,L3}}(j,\text{bl},n) \ge F_{\text{blend,L3}}(p,\text{bl},n) \cdot t_{\text{L3}}(n) - F_{\text{blend}}^{\max}(\text{bl}) \\ \cdot t_{\text{L3}}(n) \cdot (1 - v_{\text{L3}}(j,\text{bl},n)) \\ \forall(\text{bl},p) \in \text{BP},\ (\text{bl},j) \in \text{BJ},\ (j,p) \in \text{JPN}, n \in \text{NL} \end{gathered} \tag{79}$$

$$V_{\text{trans,L3}}(j,\text{bl},n) \le F_{\text{blend}}^{\max}(\text{bl}) \cdot t_{\text{L3}}(n) \cdot v_{\text{L3}}(j,\text{bl},n) \quad \forall\ (\text{bl},j) \in \text{BJ}, n \in \text{NL} \tag{80}$$

$$w_{\text{blender,L3}}(\text{bl},n) + \sum_{j\in\text{BJ}} v_{\text{L3}}(j,\text{bl},n) = 1 \quad \forall \text{bl}, n \in \text{NL} \tag{81}$$

Equation 81 forces continuous variable $w_{\text{blender,L3}}(\text{bl},n)$ to be only 0 or 1. Equation 82 observes that the blender bl is feeding product p into a tank j that contains such product. Let us note that the allocation of swing tanks computed at the second level is fixed at the third level. This equation forces continuous variable $x_{\text{L3}}(\text{bl},n)$ to be only 0 or 1

$$\sum_{j\in\text{JP}} v_{\text{L3}}(j,\text{bl},n) = x_{\text{L3}}(p,\text{bl},n) \quad \forall\ \text{bl}, n \in \text{NL} \tag{82}$$

Equations 83a and 83b define if a change in the destination tank for the product in the blender has occurred at period n (when the blender is already running). However, let us note that this change is constrained by Eqs. 81 and 82 to be made only to another tank assigned to hold the same product at period n. 0–1 continuous variable $xe_{\text{L3}}(\text{bl},n)$ represents the start or end of a blend run if its value is 1; therefore, $ve_{\text{L3}}(\text{bl},n)$ is a 0–1 continuous variable due to being penalized in Eq. 71

$$\begin{gathered} ve_{\text{L3}}(\text{bl},n) \ge v_{\text{L3}}(j,\text{bl},n) - v_{\text{L3}}(j,\text{bl},n-1) \\ - xe_{\text{L3}}(\text{bl},n) \quad \forall(\text{bl},j) \in \text{BJ}, n \in \text{NL} \end{gathered} \tag{83a}$$

$$\begin{gathered} ve_{\text{L3}}(\text{bl},n) \ge v_{\text{L3}}(j,\text{bl},n-1) - v_{\text{L3}}(j,\text{bl},n) \\ - xe_{\text{L3}}(\text{bl},n) \quad \forall(\text{bl},j) \in \text{BJ}, n \in \text{NL} \end{gathered} \tag{83b}$$

It is important to notice that the only binary variable at the third level is $v_{L3}(j,bl,n)$, which indicates that the number of discrete variables only increases with the system structure (i.e., number of blenders and product tanks) and the discretization of the scheduling horizon (i.e., the number of L3-periods).

Material Balance on Product Tanks. The volumetric balance on the product tanks specifies that the initial inventory plus the volume supplied by the blenders are equal to the final inventory plus the amount delivered, during a L3-period. Equation 84a is used during the feasibility phase to include the slack variables, and Eq. 84b is utilized during the optimization phase

$$\sum_{bl\in Bl} V_{trans,L3}(j,bl,n)+V_{pr,L3}(j,n-1)-V_{pr,L3}(j,n)-t_{L3}(n) \cdot \sum_{o\in JON} D_{pr,L3}(j,o,n)+S^{+}_{pr,L3}(j,n)-S^{+}_{pr,L3}(j,n)=0 \quad \forall j, n\in NL \tag{84a}$$

$$\sum_{bl\in Bl} V_{trans,L3}(j,bl,n)+V_{pr,L3}(j,n-1)-V_{pr,L3}(j,n) -t_{L3}(n)\cdot \sum_{o\in JON} D_{pr,L3}(j,o,n)=0 \forall j, n\in NL \tag{84b}$$

Inventory Limits. Inventory constraints are only forced on individual storage tanks by Eqs. 85 and 86

$$V^{min}_{bc}(i) \le V_{bc,L3}(i,n) \le V^{max}_{bc}(i) \quad \forall i, n\in NL \tag{85}$$

$$V^{min}_{pr}(j) \le V_{pr,L3}(j,n) \le V^{max}_{pr}(j) \quad \forall j, n\in NL \tag{86}$$

Initial Inventory. Equations 87 and 88 set the initial state of the blend component and product tanks, respectively. Note that these equations are required only for the first *L*-interval

$$V_{bc,L3}(i,n=0)=V^{start}_{bc}(i) \quad \forall i \tag{87}$$

$$V_{pr,L3}(j,n=0)=V^{start}_{pr}(j) \quad \forall j \tag{88}$$

Blend Runs. At this level, 0–1 continuous variable $xe_{L3}(bl,n)$ represents a state transition in blender bl at the beginning of period *n*; in other words, a transition from being running to being idle, or vice versa. 0–1 continuous variable $sw_{L3}(bl,n)$ represents the start of a blend run in blender bl at the beginning of period *n*. Equation 89 defines which product the blender is processing if it is running at the beginning of the horizon, and Eq. 90 determines if the blender is idle at time zero. Equations 91 and 92 identify a state transition in the blender; they force continuous variable $xe_{L3}(bl,n)$ to be 1 when a product starts or finishes a blend run. Equations 93a and 93b force $xe_{L3}(bl,n)$ to be 0 when blender is running or idle during two consecutive L3-periods, respectively. Equations 94 and 95 define if a blend run has started in the blenders

$$x_{L3}(p,bl,n=0)=x^{start}(p,bl) \quad \forall (bl,p)\in BP \tag{89}$$

$$w_{blend,L3}(bl,n=0)=w^{start}_{blend}(bl) \quad \forall bl \tag{90}$$

$$xe_{L3}(bl,n) \ge x_{L3}(p,bl,n)-x_{L3}(p,bl,n-1) \quad \forall (bl,p)\in BP, n \in NL \tag{91}$$

$$xe_{L3}(bl,n) \ge x_{L3}(p,bl,n-1)-x_{L3}(p,bl,n) \quad \forall (bl,p)\in BP, \ n \in NL \tag{92}$$

$$xe_{L3}(bl,n) \le w_{blend,L3}(bl,n)+w_{blend,L3}(bl,n-1) \quad \forall bl, \ n\in NL \tag{93a}$$

$$xe_{L3}(bl,n) \le 2-w_{blend,L3}(bl,n)-w_{blend,L3}(bl,n-1) \quad \forall bl, \ n\in NL \tag{93b}$$

$$sw_{L3}(bl,n) \ge xe_{L3}(bl,n)+w_{blend,L3}(bl,n-1)-1 \quad \forall bl, \ n\in NL \tag{94}$$

$$sw_{L3}(bl,n) \le \frac{xe_{L3}(bl,n)+w_{blend,L3}(bl,n-1)}{2} \quad \forall bl, \ n\in NL \tag{95}$$

Constant Blending Rate. Equation 96 ensures that the blending rate is constant during a blend run. Equation 97 specifies the blending rate of the blenders at the beginning of the scheduling horizon

$$-F^{max}_{blend}(bl)\cdot xe_{L3}(bl,n) \le F_{blend,L3}(p,bl,n) -F_{blend,L3}(p,bl,n-1) \le F^{max}_{blend}(bl)\cdot xe_{L3}(bl,n) \quad \forall (bl,p)\in BP, \ n\in NL \tag{96}$$

$$F_{blend,L3}(p,bl,n=0)=F^{start}_{blend}(p,bl) \quad \forall (bl,p)\in BP \tag{97}$$

Constraints on the Minimum Running Times of the Blenders. To know when a blender has surpassed its minimum allowed running time when processing product *p*, the cumulative running time of a blender at the end of period *n* is computed [i.e., $t_{blend,L3}(bl,n)$]. Equation 98 sets the running time at the beginning of the scheduling horizon. Equations 99 and 100 compute the cumulative running time at period *n* as the cumulative running time in period *n* − 1 plus the duration of period *n*, if the blender is not idle. Equation 101 restarts to zero the cumulative running time when the blender goes idle. Parameter *H* is the length of the scheduling horizon

$$t_{blend,L3}(bl,n=0)=t^{start}_{blend}(bl) \quad \forall bl \tag{98}$$

$$t_{blend,L3}(bl,n) \le t_{blend,L3}(bl,n-1)+t_{L3}(n) +H\cdot w_{blend,L3}(bl,n) \quad \forall bl, n\in NL \tag{99}$$

$$t_{blend,L3}(bl,n) \ge t_{blend,L3}(bl,n-1)+t_{L3}(n)-H \cdot w_{blend,L3}(bl,n) \quad \forall bl, \ n\in NL \tag{100}$$

$$t_{blend,L3}(bl,n) \le H\cdot \left(1-w_{blend,L3}(bl,n)\right) \quad \forall bl, \ n\in NL \tag{101}$$

Equation 102 observes that state transitions in the blender can only occur after the minimum running time has been achieved or surpassed

$$xe_{L3}(bl,n) \le \frac{t_{blend,L3}(bl,n-1)}{t^{min}_{blend}(p,bl)}+H \cdot (1-x_{L3}(p,bl,n-1)) \quad \forall (bl,p)\in BP, \ n\in NL \tag{102}$$

Equation 102 does not avoid solutions with blend runs completed at the end of the horizon (or *L*-interval) with less than the minimum running time; therefore, Eq. 103 is necessary to ensure that any blend run within the scheduling horizon has a run time at least equal to the minimum

$$t_{blend,L3}(bl,n) \ge t^{min}_{blend}(p,bl)\cdot x_{L3}(p,bl,n) \quad \forall (bl, \ p)\in BP, \ n \in NLE \tag{103}$$

Constraints on the Minimum Idle Times of the Blenders. We consider product-dependent changeover times in the blenders. The cumulative idle time of blender bl at the end of period *n* is denoted as $it_{blend,L3}(bl,n)$. Equation 104 sets the idle time at the beginning of the scheduling horizon. Equations 105 and 106 compute the cumulative idle time at period n as the cumulative idle time at period $n-1$ plus the duration of period n, if the blender is not running. Equation 107 restarts to zero the idle time when the blender starts a blend run

$$it_{\text{blend,L3}}(\text{bl}, n=0) = \text{it}^{\text{start}}_{\text{blend}}(\text{bl}) \quad \forall \text{bl} \tag{104}$$

$$\text{it}_{\text{blend,L3}}(\text{bl}, n) \le \text{it}_{\text{blend,L3}}(\text{bl}, n-1) + t_{\text{L3}}(n) + H \cdot \left(1 - w_{\text{blend,L3}}(\text{bl}, n)\right) \quad \forall \text{bl}, \ n \in \text{NL} \tag{105}$$

$$\text{it}_{\text{blend,L3}}(\text{bl}, n) \ge \text{it}_{\text{blend,L3}}(\text{bl}, n-1) + t_{\text{L3}}(n) - H \cdot \left(1 - w_{\text{blend,L3}}(\text{bl}, n)\right) \quad \forall \text{bl}, \ n \in \text{NL} \tag{106}$$

$$\text{it}_{\text{blend,L3}}(\text{bl}, n) \le H \cdot w_{\text{blend,L3}}(\text{bl}, n) \quad \forall \text{bl}, \ n \in \text{NL} \tag{107}$$

Equation 108 ensures that a blender cannot start to process product p in period n unless the cumulative idle time at period $n-1$ is greater than the minimum required

$$\text{it}_{\text{blend,L3}}(\text{bl}, n-1) \ge \text{it}^{\min}_{\text{blend}}(p, \text{bl}) \cdot x_{\text{L3}}(p, \text{bl}, n) - H \cdot \left(1 - w_{\text{blend,L3}}(\text{bl}, n-1)\right) \quad \forall (\text{bl}, \ p) \in \text{BP}, \ n \in \text{NL} \tag{108}$$

Constraints on the Minimum Production Volume of the Blend Runs. We consider that a blend run should produce at least a minimum amount of product. The cumulative volume produced by blender bl as the start of the blend run up to the end of period n (if the blend run has not been completed) is denoted as $vc_{\text{blend,L3}}(\text{bl},n)$. Equation 109 sets the cumulative volume at the beginning of the scheduling horizon. Equations 110 and 111 compute the cumulative volume produced up to period n as the cumulative volume at period $n-1$ plus the volume blended in period n, if the blender is not idle. Equation 112 restarts to zero the cumulative volume when the blender ends a blend run and while it remains idle

$$vc_{\text{blend,L3}}(\text{bl}, n=0) = vc^{\text{start}}_{\text{blend}}(\text{bl}) \quad \forall \text{bl} \tag{109}$$

$$vc_{\text{blend,L3}}(\text{bl}, n) \le vc_{\text{blend,L3}}(\text{bl}, n-1) + \sum_{p \in \text{BP}} V_{\text{blend,L3}}(p, \text{bl}, n) + F^{\max}_{\text{blend}}(\text{bl}) \cdot H \cdot w_{\text{blend,L3}}(\text{bl}, n) \quad \forall \text{bl}, \ n \in \text{NL} \tag{110}$$

$$vc_{\text{blend,L3}}(\text{bl}, n) \ge vc_{\text{blend,L3}}(\text{bl}, n-1) + \sum_{p \in \text{BP}} V_{\text{blend,L3}}(p, \text{bl}, n) - F^{\max}_{\text{blend}}(\text{bl}) \cdot H \cdot w_{\text{blend,L3}}(\text{bl}, n) \quad \forall \text{bl}, \ n \in \text{NL} \tag{111}$$

$$vc_{\text{blend,L3}}(\text{bl}, n) \le F^{\max}_{\text{blend}}(\text{bl}) \cdot H \cdot \left(1 - w_{\text{blend,L3}}(\text{bl}, n)\right) \quad \forall \text{bl}, \ n \in \text{NL} \tag{112}$$

Equations 113 and 114 constraint a blender to end a blend run until the minimum volume has been produced

$$xe_{\text{L3}}(\text{bl}, n) \le \frac{vc_{\text{blend,L3}}(\text{bl}, n-1)}{\text{VMIN}^{\min}_{\text{blend}}(p, \text{bl})} + F^{\max}_{\text{blend}}(\text{bl}) \cdot H \cdot (1 - x_{\text{L3}}(p, \text{bl}, n-1)) \quad \forall (\text{bl}, \ p) \in \text{BP}, \ n \in \text{NL} \tag{113}$$

$$vc_{\text{blend,L3}}(\text{bl}, n) \ge \text{VMIN}^{\min}_{\text{blend}}(p, \text{bl}) \cdot x_{\text{L3}}(p, \text{bl}, n) \quad \forall (\text{bl}, p) \in \text{BP}, \ n \in \text{NLE} \tag{114}$$

Equations 109–114 are not used if these minimum blend thresholds are not considered; in that case, the minimum volume that may be produced by a blend run is given as the minimum blending rate multiplied by the minimum running time. From Eqs. 99 to 113, parameter H can be substituted by the smallest possible number to tighten the model.

Order Delivery. Equation 115 computes the volume to deliver of order o during L-interval l, denoted as $\text{Demand}_{\text{order,L3}}(o,l)$, according to the second-level solution. As stated before, in this work only one L2-period is contained by one L-interval during the feasibility phase. Note that $of_{\text{L2}}(o,m)$ is a parameter and not a variable at the third level

$$\text{Demand}_{\text{order,L3}}(o, l) = \text{Demand}(o) \cdot \sum_{m \in \text{ML}} of_{\text{L2}}(o, m) \quad \forall o \tag{115}$$

Equations 116a, 116b, 117a, and 117b represent the material balance around the lifting/shipping ports. Together, Eqs. 116a and 117a force the delivery to occur within the corresponding window during the feasibility phase (and Eqs. 116b and 117b during the optimization phase). Equations 116a and 116b constraint the amount shipped within the delivery window [the delivery window is given by set NLO (n,l,o)] to be equal to the demand, while Eqs. 117a and 117b observe that the amount shipped within the whole L-interval is equal to the demand. Therefore, no delivery occurs outside the delivery window

$$t_{\text{L3}}(n) \cdot \left[\sum_{n \in \text{NLO}} \sum_{(j,o) \in \text{JON}} D_{\text{pr,L3}}(j, o, n)\right] = \text{Demand}_{\text{order,L3}}(o, l) + S^{+}_{\text{order,L3}}(o, l) - S^{-}_{\text{order,L3}}(o, l) \quad \forall o \tag{116a}$$

$$t_{\text{L3}}(n) \cdot \left[\sum_{n \in \text{NLO}} \sum_{(j,o) \in \text{JON}} D_{\text{pr,L3}}(j, o, n)\right] = \text{Demand}_{\text{order,L3}}(o, l) \quad \forall o \tag{116b}$$

$$t_{\text{L3}}(n) \cdot \left[\sum_{n \in \text{NL}} \sum_{(j,o) \in \text{JON}} D_{\text{pr,L3}}(j, o, n)\right] = \text{Demand}_{\text{order,L3}}(o, l) + S^{+}_{\text{order,L3}}(o, l) - S^{-}_{\text{order,L3}}(o, l) \quad \forall o \tag{117a}$$

$$t_{\text{L3}}(n) \cdot \left[\sum_{n \in NL} \sum_{(j,o) \in \text{JON}} D_{\text{pr,L3}}(j, o, n)\right] = \text{Demand}_{\text{order,L3}}(o, l) \quad \forall o \tag{117b}$$

Delivery rate of order o must be equal to or less than its specified maximum (Eq. 118), and the total delivery rate of tank j must be equal to or less than its maximum delivery capacity (Eq. 119)

$$\sum_{j \in \text{JON}} D_{\text{pr,L3}}(j, o, n) \le D^{\max}_{\text{order}}(o) \quad \forall o, n \in \text{NLO} \tag{118}$$

$$\sum_{o \in \text{JON}} D_{\text{pr,L3}}(j, o, n) \le D^{\max}_{\text{pr}}(j) \quad \forall j, n \in \text{NLO} \tag{119}$$

To keep the delivery rate as constant as possible, Eqs. 120 and 121 compute the difference between delivery rates, which is penalized in Eq. 71

$$\text{Dsw}_{\text{L3}}(j, o, n) \ge D_{\text{pr,L3}}(j, o, n) - D_{\text{pr,L3}}(j, o, n-1) \quad \forall (j, o) \in \text{JON}, n \in \text{NLO} \tag{120}$$

$$\text{Dsw}_{\text{L3}}(j, o, n) \ge D_{\text{pr,L3}}(j, o, n-1) - D_{\text{pr,L3}}(j, o, n) \quad \forall (j, o) \in \text{JON}, \ n \in \text{NLO} \tag{121}$$

Target Inventories. During the forward rolling window pass, Eqs. 122 and 123 set the inventory targets for the blend component and the product tanks at the end boundary of the L-interval l according to the second-level solution. Starting inventories are set according to the solution from the previous L-interval $(l-1)$.

During the reverse rolling window pass, the inventory targets for the blend component and the product tanks at the start boundary of the L-interval l are set by the values computed during the forward pass. Inventories at the end boundary of the L-interval are set according to the solution from the L-interval $(l+1)$

$$V_{\mathrm{bc,L3}}(i,n)=V_{\mathrm{bc,L2}}(i,m) \quad \forall i, n \in \mathrm{NLE}, m \in \mathrm{MLE} \qquad (122)$$

$$V_{\mathrm{pr,L3}}(j,n)=V_{\mathrm{pr,L2}}(j,m) \quad \forall j, n \in \mathrm{NLE}, m \in \mathrm{MLE} \qquad (123)$$

MPIP Scheduling Algorithm

The algorithm presented here is based on the gasoline blend planning and scheduling problem; however, it can be applied to any system with similar characteristics.

Step 1: Modify the original order delivery windows if necessary. This is required to transform the problem from a continuous-time to a discrete-time domain.

Step 2: Construct the CTD and the CATP curves. Determine the pinch point(s) location.

Step 3: Set iteration counters $\mathrm{iter_S}=1$ and $\mathrm{iter_P}=1$. Divide the scheduling horizon (at the first level) in the number of L1-periods indicated by the pinch points, unit cost, and quality breakpoints of blend components.

Step 4: Solve the first-level model to compute the optimal blend recipes (Eq. 1–15).

- In the first "planning" iteration ($\mathrm{iter_P}=1$), the volumes to produce of each product in each L1-period are the minimum amounts required to meet the aggregated demand in each L1-period.
- In following "planning" iterations ($\mathrm{iter_P}>1$), the volumes to produce are defined according to the solution of the second level (see Step 7) or the third level (see Step 12).
- If the quantity to be produced [i.e., $V_{\mathrm{blend,L1}}(p,k)$] violates the maximum blend capacity or the minimum blend size threshold constraints, volumes are adjusted by moving the least amount possible of volume to previous L1-periods (i.e., preblending).
- If any inventory slack variable has a nonzero value at the solution, the problem is infeasible as the availability or quality of blend components is not enough to meet the product quality specifications or to deliver the products within the delivery windows. Stop.

Step 5: Solve the second level blend planning model (Eqs. 16, 17, and 19–55).

Step 6: If all the inventory slack variables from Step 5 are zero, a feasible blend plan, based on optimal recipes computed at the first level, has been found; go to Step 9. Otherwise, continue to Step 7: Subdivide the L1-period at the end boundary of the L2-period with the first infeasibility.

- The volumes to be blended in each new L1-period are given by the solution of Step 5 plus the positive slacks minus the negative slacks. If $V_{\mathrm{blend,L1}}(p,k)$, for any p or k, violates the maximum blend capacity or the minimum blend size threshold constraints, volumes are adjusted by moving the least amount possible of volume to previous L1-periods.

Step 8: Set $\mathrm{iter_P}=\mathrm{iter_P}+1$. Go back to Step 4.

Step 9: Solve the second level approximate scheduling model (Eqs. 17–61).

- The solution provides an initial guess (upper bound) for the number of blend runs and a lower bound for the number of product transitions in the blenders.
- The solution provides the product-tank allocation that yields the minimum product transitions in the storage tanks.

Step 10: Define the L-intervals for the third level forward rolling window pass.

- The length of these L-intervals can be different but their boundaries must match L2-period boundaries.
- L-intervals are not required to overlap.

Step 11: Solve the third-level feasibility phase model (Eqs. 69, 70, and 72–123) for each L-interval from Step 10.

- Use the forward rolling window approach, that is, start with $l=1$ and move on to the end of the scheduling horizon. In this way, the initial conditions of each L-interval are computed considering the conditions at the beginning of the horizon.

Step 12: If the solution from Step 11 has all slack variables, for all L-intervals, with values equal to zero, continue to Step 14. Otherwise:

- Subdivide the scheduling horizon at the first level at the start boundary of the L2-period containing the largest infeasibility. Small infeasibilities may disappear with new recipes.
- In the next iteration, if nonzero slacks continue to appear in the same L2-period, subdivide the scheduling horizon at the first level at the end boundary of that L2-period. If such period is the last L2-period, or if nonzero slacks continue appearing in the same period in following iterations; subdivide the horizon at the first level in such a way that the blend run with nonzero slacks is delimited (i.e., a blend recipe is computed for that particular blend run). A good guess to delimit the blend run is given by $t_{\mathrm{blend,L2}}(p,\mathrm{bl},m)+\mathrm{it}_{\mathrm{blend}}^{\mathrm{min}}(p,\mathrm{bl})$.
- Necessity of moving volumes to previous time periods as in Step 7 is less common at this level.
- Inventory targets computed at the first level are not fixed at the corresponding boundaries at the second level if the inventory targets at other second level boundaries already define how much product is required to be blended with the same blend recipe; for example, when a single blend run is delimited at the first level.

Step 13: Set $\mathrm{iter_S}=\mathrm{iter_S}+1$, and go back to Step 4.

Step 14: Solve the third level optimization phase model (Eqs. 70–123) for each L-interval from Step 10.

- Use the forward rolling window approach.
- If inventory infeasibilities appear or the model for an L-interval is infeasible, this is due to a different boundary condition (i.e., the initial blender conditions are different from those used at Step 11). Resolve previous L-interval and fix the initial conditions computed in Step 11.

Step 15: Determine new L-intervals for the reverse rolling window pass:

- If the same product is being blended in adjacent L2-periods, in the same blender and using the same blend recipe, those L2-periods can be grouped into one L-interval.
- L-intervals can be overlapped if necessary.
- Increasing the length of the L-intervals increases the model size as well as the execution times required to solve it.

Step 16: Solve the third level feasibility phase model (Eqs. 69, 70, and 72–121) for each L-interval from Step 15, including the following constraint

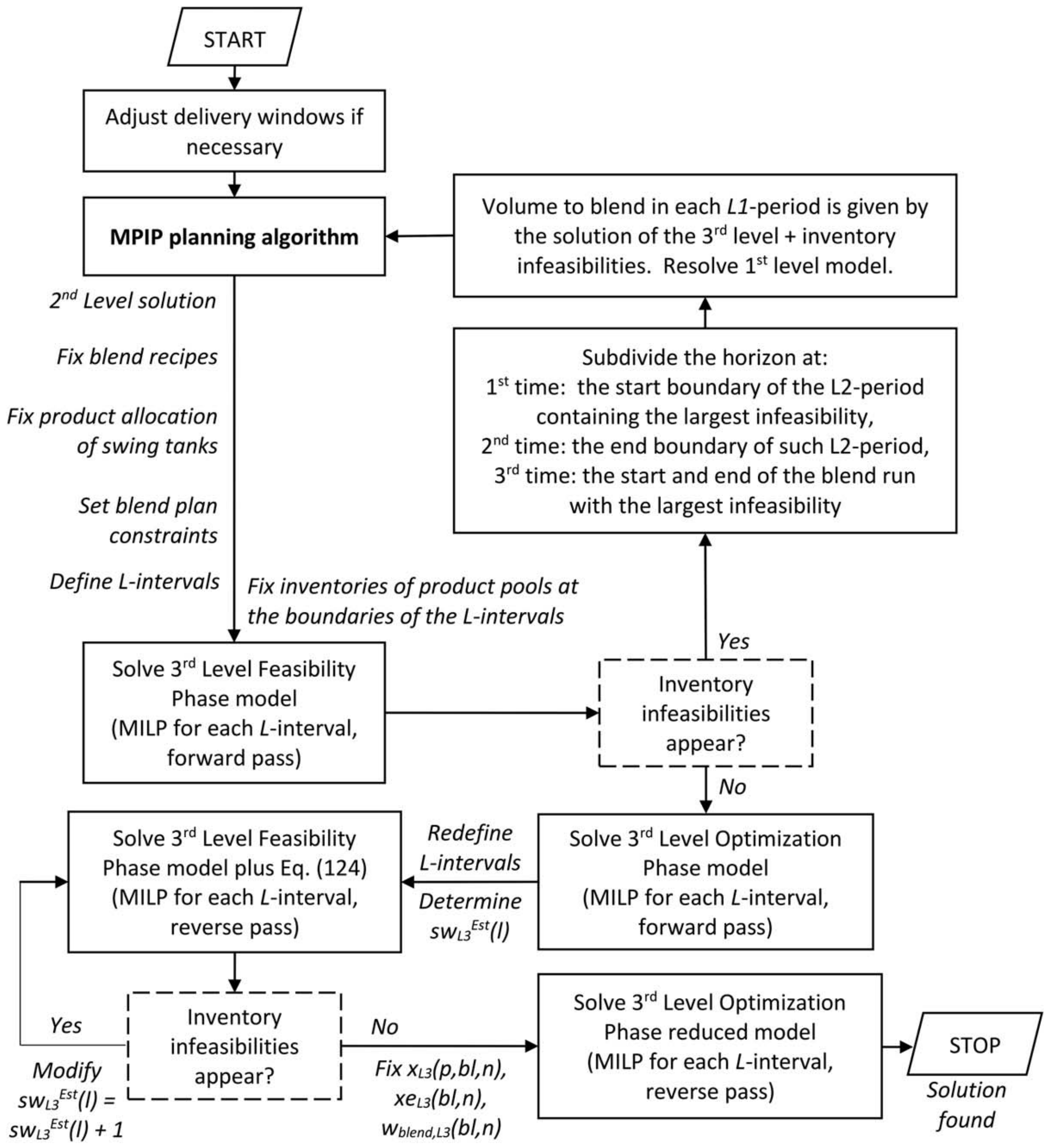
START
Adjust delivery windows if necessary
MPIP planning algorithm
Volume to blend in each L1-period is given by the solution of the 3rd level + inventory infeasibilities. Resolve 1st level model.
2nd Level solution
Fix blend recipes
Fix product allocation of swing tanks
Set blend plan constraints
Define L-intervals
Subdivide the horizon at: 1st time: the start boundary of the L2-period containing the largest infeasibility, 2nd time: the end boundary of such L2-period, 3rd time: the start and end of the blend run with the largest infeasibility
Fix inventories of product pools at the boundaries of the L-intervals
Yes
Solve 3rd Level Feasibility Phase model (MILP for each L-interval, forward pass)
Inventory infeasibilities appear?
No
Solve 3rd Level Feasibility Phase model plus Eq. (124) (MILP for each L-interval, reverse pass)
Redefine L-intervals
Determine sw_L3^Est(l)
Solve 3rd Level Optimization Phase model (MILP for each L-interval, forward pass)
Yes
Modify sw_L3^Est(l) = sw_L3^Est(l) + 1
Inventory infeasibilities appear?
No
Fix x_L3(p,bl,n), xe_L3(bl,n), w_blend,L3(bl,n)
Solve 3rd Level Optimization Phase reduced model (MILP for each L-interval, reverse pass)
STOP
Solution found

Figure 4. Flowchart for the MPIP scheduling algorithm.

$$\sum_{\mathrm{bl}\in\mathrm{Bl}} sw_{\mathrm{L3}}(\mathrm{bl},n)=sw_{\mathrm{L3}}^{\mathrm{Est}}(l) \quad \forall n \in \mathrm{NL} \tag{124}$$

- $sw_{\mathrm{L3}}^{\mathrm{Est}}$ is the estimated lower bound for the total number of blend runs according to solution from Step 14 (i.e., one blend run per product being blended in the L-interval l, minus the blend runs that have started in L-interval $l-1$)
- Use a reverse rolling window approach, that is, start with L-interval $l=L$ and continue until the beginning of the horizon. In this way, changes at the boundary conditions will not affect the feasibility of previous L-intervals.
- If the solution for an L-interval contains inventory slacks greater than zero, set $sw_{\mathrm{L3}}^{\mathrm{Est}}(l)=sw_{\mathrm{L3}}^{\mathrm{Est}}(l)+1$ and resolve the L-interval. Repeat until the solution does not contain inventory infeasibilities (feasibility is guaranteed by Step 14).
- This step aims to join the most possible blend runs computed at Step 14.
- Inventory levels are only fixed at the boundaries of the L-interval being solved. In other words, if the L-intervals overlap (i.e., end boundary of L-interval $l-1$ is within L-interval l), then the new inventory levels computed at l should be used for $l-1$.
- Variables $x_{\mathrm{L3}}(p,\mathrm{bl},n)$ and $w_{\mathrm{blend,L3}}(\mathrm{bl},n)$ are fixed in the last L3-period n of L-interval l if in L3-period $n+1$ the blender is starting or continuing a blend run. In addition, parameters $t_{\mathrm{blend}}^{\min}(p,\mathrm{bl})$ and $\mathrm{VMIN}_{\mathrm{blend}}(p,\mathrm{bl})$ are adjusted for Eqs. 103 and 114, respectively, to account for blend runs that are completed in L-interval $l+1$.

Step 17: Fix variables $F_{\mathrm{blend,L3}}(p,\mathrm{bl},n)$, $x_{\mathrm{L3}}(p,\mathrm{bl},n)$, $xe_{\mathrm{L3}}(\mathrm{bl},n)$, and $w_{\mathrm{blend,L3}}(\mathrm{bl},n)$, and solve the third-level

Table 1. Test Set #1, Case Study 13—Demand Profile #5

Delivery Window		Product U87		Product U91		Product U93	
Start Time (h)	End Time (h)	Order	Amount ($\times 10^3$ bbl)	Order	Amount ($\times 10^3$ bbl)	Order	Amount ($\times 10^3$ bbl)
0	24	O1	60	O14	50	O26	30
24	48	O2	50	O15	80	O27	30
48	72	O3	50	O16	70	O28	45
72	96	O4	70	O17	50	–	0
96	120	O5	90	O18	50	O29	30
120	144	O6	80	O19	30	O30	40
144	168	O7	130	O20	30	O31	30
168	192	O8	50	O21	30	O32	30
192	216	–	0	O22	30	O33	30
216	240	O9	30	O23	30	O34	30
240	264	O10	50	–	0	O35	30
264	288	O11	50	O24	40	–	0
288	312	O12	50	O25	30	O36	30
312	336	O13	80	–	0	O37	40

optimization phase reduced model (Eqs. 70, 71, 80–86, 88, and 115–121) for each L-interval from Step 15. Stop.

- This step reduces the variations in the final delivery rates and the variations in the transfer sequence from the blenders to the storage tanks.

Steps 2–9 are the same steps of the MPIP planning algorithm for gasoline blend planning (see Part I of this article); therefore, the flowchart of the MPIP scheduling algorithm can be visualized as Figure 4. Hence, we refer to the iterations at Step 8 as planning iterations (iter$_P$), and those at Step 13 as scheduling iterations (iter$_S$).

Numerical Results and Discussion

We present two sets of case studies: Test Set #1 contains the examples discussed in Part I of this article (i.e., case studies with our own data), whereas Test Set #2 are examples taken from the literature[4] which incorporate all of the characteristics given in the Problem Statement section of this article. Test Set #1 is used to produce detailed schedules from the blend plans computed at the second level of our case studies presented in Part I of this article, and Test Set #2 is used to compare our MPIP scheduling algorithm with a full-space continuous-time model. Test Set #1 contains problems where the operation range of the blenders is small, the demand requirements are high, and there is one nonlinear blending property; whereas in Test Set #2 there is a large difference between the maximum and minimum blending rates, demand requirements are low, and all blending properties are assumed linear. For all examples in each Test Set, the length of the L3-periods is 1 h.

Test Set #1: Integration of planning and detailed scheduling

The gasoline blending system and all data required for these case studies appear in Part I of this article.

The penalty coefficients at the third level are:

- For all blenders, the penalty for starting a blend run is PenaltyBsw$_{L3} = 1 \times 10^4$ \$.
- The penalty for variations in the delivery rate is PenaltyDsw$_{L3}$ = 100\$.
- The penalty profile for late delivery of orders [i.e., PenaltyDpm$_{L3}(n)$] increases from 0\$ at the beginning of an L-interval up to 120\$ at the end of the L-interval.
- The penalty for changing the destination tank for the product from the blender is PenaltyTsw$_{L3}$ = 500\$.
- The penalty for a blender being running for one L3-period is PenaltyBL$_{L3}$ = 1\$.

We consider the highest penalty for a blend run to obtain the production sequence with the minimum possible number of blend runs; then, the next higher penalties are for variations in the delivery rates and late deliveries, and the smallest penalty corresponds to long blend runs. The demand orders in our case studies involve a single product and their delivery time windows are assumed to be one day. As we are using the L2-periods as 1 day, there is no need to adjust the time delivery windows.

All case studies have been computed on a DELL PowerEdge T310 (Intel® Xeon® CPU, 2.40 GHz, and 12 GB RAM) running Windows Server 2008 R2 OS. GAMS IDE 23.7.3 was used to solve each one of the case studies. The first-level NLP model was solved using IPOPT, and the second- and third-level models were solved using CPLEX 12.3.

Illustrative Example #1: *Test Set #1, Case Study 13—Two Nonidentical Blenders, Irregular Supply Profile of Blend Components.* Case study 13 was used to illustrate the steps of the MPIP planning algorithm in Part I of this article; now, we will continue with the steps of the MPIP scheduling algorithm. This case study has demand profile #5 (shown in Table 1), 2 blenders (A and B), and the supply flow rate of components is irregular along the planning horizon. We continue the calculations from the solution of the second level approximate scheduling solution presented in Part I of this article. The next step is to solve the third-level feasibility model. Each L3-period is 1-hour long.

First Scheduling Iteration. Let us define 14 L-intervals (Step 10), each one for each L2-period (i.e., each L-interval has 24 L3-periods). The inventory levels corresponding to the L2-period boundaries are fixed at the respective boundaries of the L-intervals. In this example, solution of the third-level feasibility phase (Step 11) presents two inventory slacks with nonzero values in the last L-interval on the light naphtha (LNP) component tank, $S_{bc,L3}{}^{+}$(LNP,336) = 0.0999 $\times$ 10^3 bbl, and $S_{bc,L3}{}^{-}$(LNP,330) = 0.0999 $\times$ 10^3 bbl. This means that LNP tank overflows at period $n = 330$. The positive slack at period $n = 336$ only appears because the target inventory at the end of L-interval $l = 14$ is not met as the overflow volume is not considered available anymore for future periods. Therefore, Step 12 of the MPIP scheduling algorithm indicates that we need to subdivide the last L1-period $k = 4$ at the point in time corresponding to the start boundary of L2-period $m = 14$. This example illustrates that

Table 2.Test Set. 1, Case Study 13—Blend Plan

L2-Periods			Production Volumes ($\times 10^3$ bbl)				
			Blender A			Blender B	
Period ID	Start Time (h)	End Time (h)	U87	U91	U93	U91	U93
$m = 1$	0	24	–	–	–	–	59
$m = 2$	24	48	100	–	–	–	30
$m = 3$	48	72	–	70	–	–	–
$m = 4$	72	96	70	–	–	64	–
$m = 5$	96	120	103.5	–	–	36	30
$m = 6$	120	144	103.5	–	–	30	36
$m = 7$	144	168	93	–	–	30	30
$m = 8$	168	192	50	30	–	–	30
$m = 9$	192	216	–	31.5	–	–	30
$m = 10$	216	240	31.5	33.3	–	–	30
$m = 11$	240	264	48.5	–	–	–	30
$m = 12$	264	288	54.3	–	–	65.2	–
$m = 13$	288	312	45.7	–	–	–	30
$m = 14$	312	336	80	–	–	–	40

typically at the third level the inventory infeasibilities are relatively small.

Second Scheduling Iteration. The new blend recipes computed at Step 4 have a cost equal to BlendCost$_{L1}$ = 37,784.52 $\times$ 10^3 \$, and the approximate schedule computed by the second level (Step 9) is presented in Table 2.Test Set. This approximate schedule has a total cost equal to Z_{L2} = 38,502.52 $\times$ 10^3 \$. No product switchover is required in the storage tanks. The third-level feasibility phase (Step 11) did not find any inventory infeasibilities. The third-level optimization phase (Step 14) was solved initially with 14 *L*-intervals (one for each L2-period), and its solution is shown in Figure 5a. Then, two contiguous L2-periods where the same product is being blended in the same blender were merged into a new *L*-interval (Step 15). L2-periods with different blend recipes were not merged because it was considered that a single blend run should have only one blend recipe. The *L*-intervals from Step 15 were solved (Step 16) using the previous solution as starting point. Figure 5b shows the production schedule computed at Step 16. The final delivery schedule and the inventory profiles of blend components and product pools (which are computed at Step 17 and that correspond to the production schedule shown in Figure 5b) are presented in Figures 6 and 7, respectively.

Test Set #1 Results. Table 3 shows the difference between the second- and third-level solutions (Step 9 and Step 17 of the MPIP scheduling algorithm, respectively). The blend cost is the same at both levels due to fixed blend recipes and inventory levels. The number of blend runs is smaller at the third level as some of the blend runs at the second level can be combined into single blend runs. The minimum number of product transitions in the blenders is achieved in some cases, and if it is not, the difference is very small. No product transitions are required in the storage tanks. Table 3 also presents the total number of scheduling iterations required and the cumulative execution times required at each level (i.e., the total time at each level considering all scheduling iterations). For our case studies, only three scheduling iterations were required at most.

The forward pass at the third level is carried out using 14 *L*-intervals (one for each L2-period), each one of them with 24 L3-periods. The sum of the cumulative execution times at

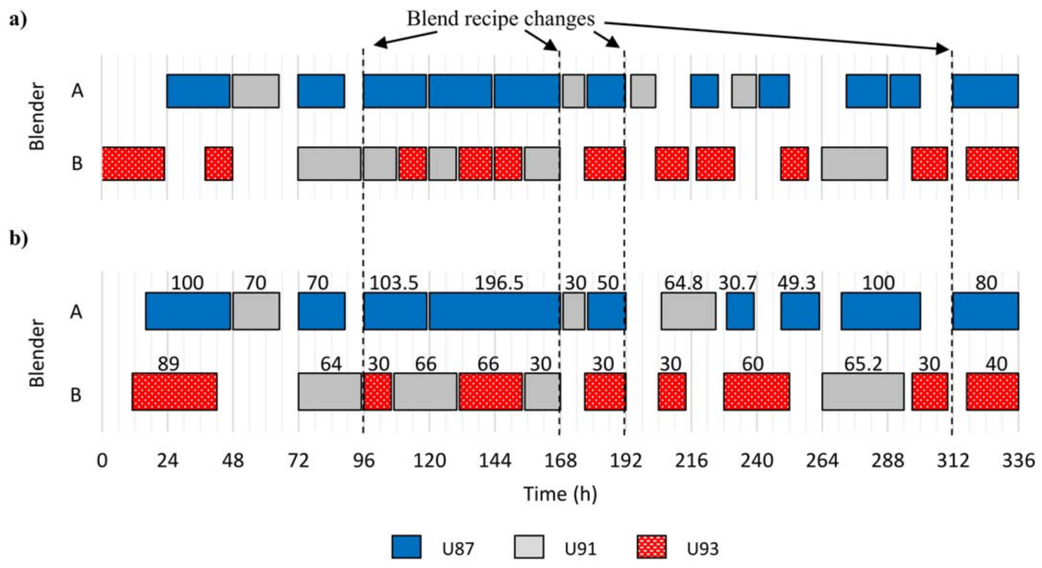

Figure 5. Test Set #1, Case study 13—Production schedule computed using (a) *L*-intervals with 24 L3-periods and (b) using *L*-intervals with 48 L3-periods.

Units in kbbl. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

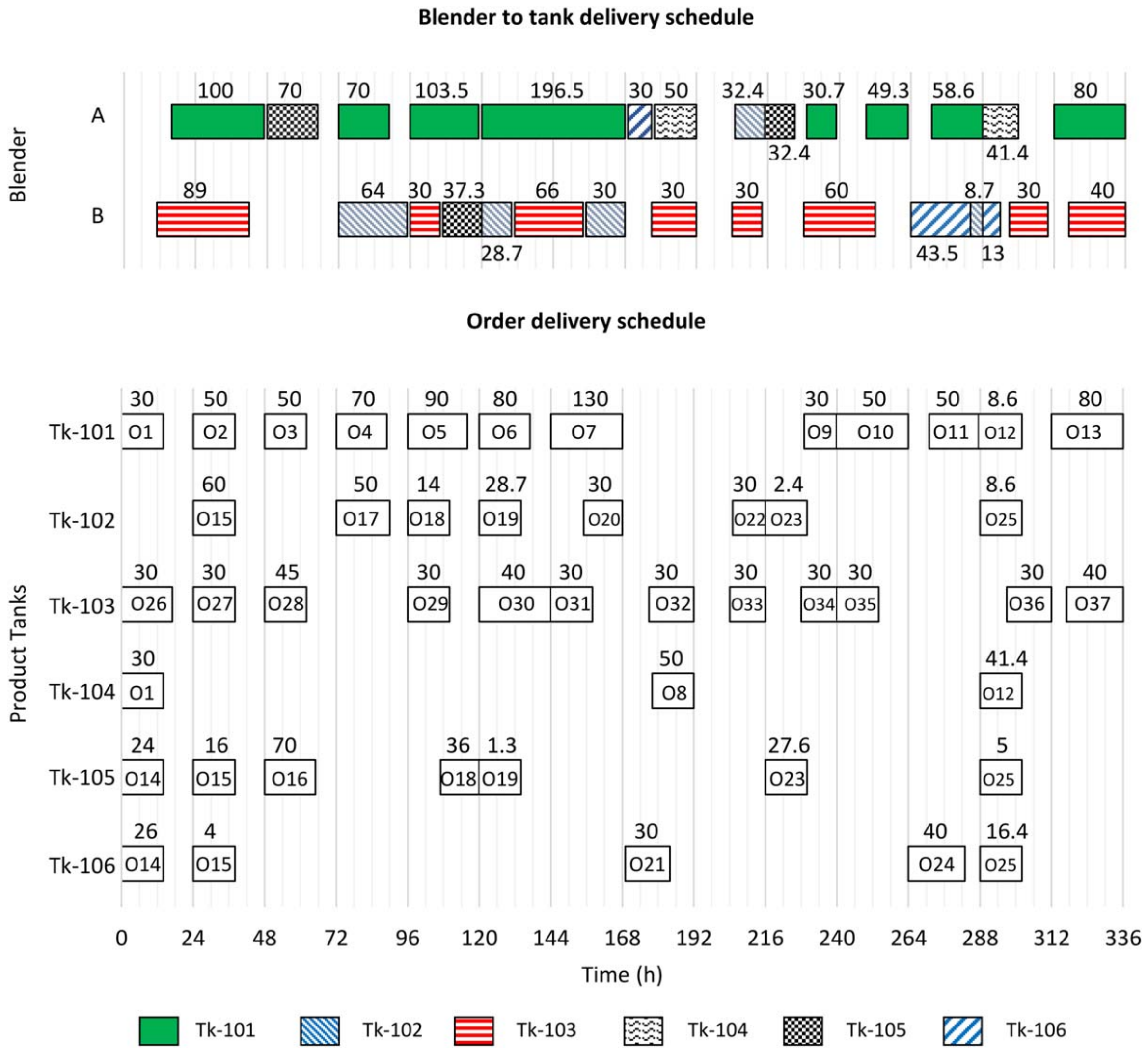

Figure 6. Test Set #1, Case study 13—Delivery schedule.

Units in kbbl. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

the first level, the second level, and the third level forward pass is the time required to construct an initial solution for the detailed schedule.

The reverse pass at the third level is carried out using *L*-intervals with 48 L3-periods. The number of these *L*-intervals varies in each case study as it depends on the number of L2-periods that are adequate to merge (see Step 15). The execution times required to solve all of these 48-hour *L*-intervals are shown in the last column from Table 3.

Regarding the model size, the number of discrete variables is greater at the second level (which is solved for the entire horizon) than at the third level with *L*-intervals of 14 or 48 h; however, the third-level model with 48-hour *L*-intervals requires more computational effort to be solved. The number of discrete variables at the third level increases only with the addition of blenders or product storage tanks to the system.

Test Set #2: Comparison of the MPIP scheduling algorithm with full-space continuous-time model using unit slots

Examples 3, 4, 7, 8, 9, 12, and 14 from Li and Karimi[4] were solved using the MPIP scheduling algorithm for comparison purposes. All examples were computed on a DELL PowerEdge T310 (Intel® Xeon® CPU, 2.40 GHz, and 12 GB RAM) running Windows Server 2008 R2 OS. GAMS IDE 23.7.3 was used to solve each one of the case studies. This computer is approximately 25% slower and has 4GB RAM less than the machine used by Li and Karimi[4].

Time delivery windows for orders O8, O10, O13, O19, and O33 were reduced from [118, 190], [150.5, 185.5], [0, 56], [0, 50], and [0, 76], to [120, 190], [150, 185], [0, 48], [0, 48], and [0, 72], respectively. Scheduling horizon at the second level is divided in nine L2-periods (period $m = 1$ to $m = 7$ are 24-hours long, period $m = 8$ has 22 h, while $m = 9$ is a 2-hour period) in example 3, 4, 7, 8, 9, and 12; and the horizon in example 14 is divided in eight L2-periods, one per day. The cost coefficients of the blend components and the penalties associated with the number of blend runs and product transitions in the tanks are the same as those presented by Li and Karimi.[4] The penalty coefficients for the third level are the same as those used in Test Set #1.

Li and Karimi[4] use blend indices which blend linearly on a volumetric basis instead of the actual quality properties; therefore, the first level model is a linear program and all the three levels were solved using CPLEX 12.3.

Table 3. Test Set 1—Comparison between the Solutions Computed at the Second and Third Level

Case Study	Demand Profile	# Pinch Points	# Blenders	Comp. Supply	Blend Cost ($\times 10^3$ \$)	Second Level Solution: # Blend runs	Second Level Solution: # Switches[a]	Third Level Solution: # Blend Runs	Third Level Solution: # Switches[a]	# Scheduling Iterations	Cumulative CPU Time (s): First Level	Cumulative CPU Time (s): Second Level[b]	Cumulative CPU Time (s): Third Level Forward Pass	Cumulative CPU Time (s): Third Level Reverse Pass
1	1	0	1	C	37,542.5	18	14	16	15	1	0.6	300.0	75.7	905
2	2	1	1	C	38,309.8	24	15	20	17	2	2.1	121.1	95.7	1923
3	3	1	1	C	37,991.1	22	16	21	18	3	1.9	261.4	108.9	1926
4	3	1	2	C	37,991.1	24	10	22	12	1	0.6	300.0	108.0	2696
5	4	2	2	C	37,681.1	26	8	22	9	1	0.9	93.0	153.2	3343
6	5	2	2	C	37,324.3	28	8	24	12	1	2.2	37.3	320.9	652
7	6	3	3	C	37,377.5	33	10	27	10	1	0.9	300.0	104.9	2573
8	1	0	1	I	37,943.4	18	14	17	16	3	1.9	505.2	165.5	981
9	2	1	1	I	38,754.2	26	15	21	19	2	1.2	22.4	106.4	3477
10	3	1	1	I	38,405.2	25	16	21	18	3	2.2	83.6	179.8	2430
11	3	1	2	I	38,405.2	26	11	20	13	1	0.7	197.0	186.1	5106
12	4	2	2	I	38,073.4	26	10	23	10	1	0.9	15.8	114.2	3666
13	5	2	2	I	37,784.5	31	10	25	16	2	2.4	25.7	102.3	1923
14	6	3	3	I	37,796.4	33	10	26	10	1	1.1	300.0	138.6	2,455

[a]Product switches in the blenders.
[b]Maximum allocated time in each iteration = 300 s.

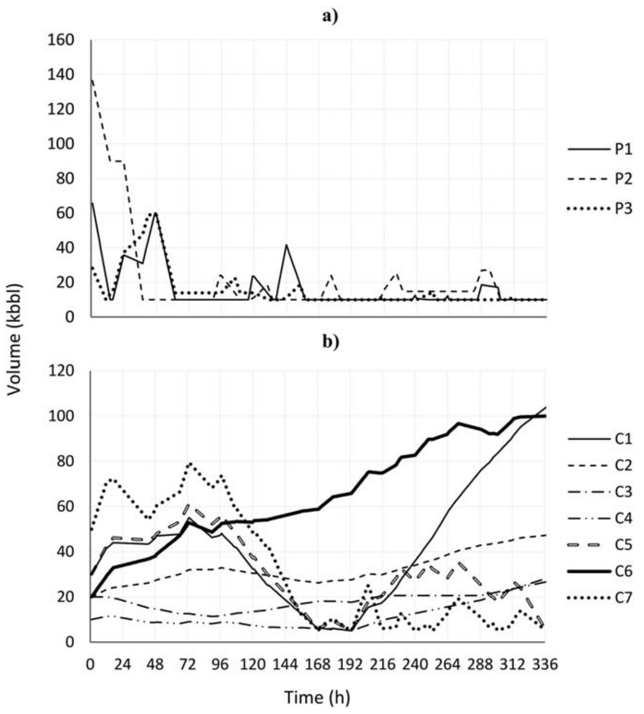

Figure 7. Test Set #1, Case study 13—Inventory profiles of (a) product pools and (b) blend components.

Illustrative Example #2: Example 12—Two Blenders, Constant Supply Profile of Blend Components. In this example, the gasoline blending system has 2 blenders, 9 blend components with dedicated tanks, 5 products, and 11 swing tanks; there are 35 orders to deliver within a scheduling horizon of 8 days (192 h). Nine product qualities are required to be within specification and blend indices are utilized to compute these (i.e., linear quality constraints are used). The supply flow rate of components is constant along the scheduling horizon. The first step is to adjust the start and end times of the delivery windows. The cumulative curves for this example (after the adjustment of the delivery windows) are shown in Figure 8, where it can be seen that there is only one inventory pinch point at time $t = 190$ h. As there are no orders to deliver after the pinch point, there is no production in the last 2 h of the horizon.

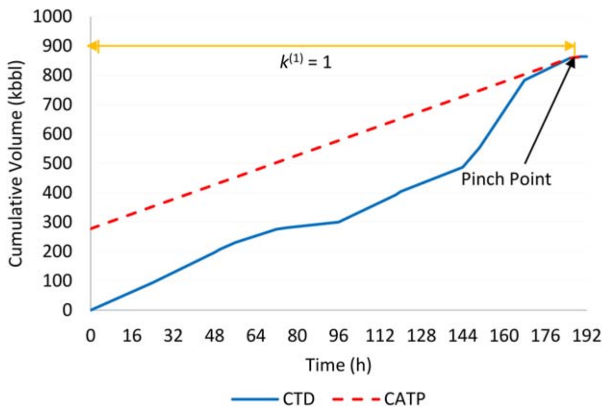

Figure 8. Test Set #2, Example 12—Cumulative curves, inventory pinch points, and L1-periods.

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

Table 4. Test Set #2, Example 12—Blend Recipes for Period $k = 1$ (i.e., time interval [0, 190 h])

	Product				
Component	P1	P2	P3	P4	P5
C1	0.0127	–	0.1000	0.2400	–
C2	0.3873	0.3198	0.2900	0.1000	0.3076
C4	0.4000	0.4500	0.4300	0.3260	0.4000
C5	–	–	–	0.1899	–
C6	0.2000	0.2200	0.1800	0.1441	0.2000
C8	–	0.0102	–	–	0.0924

First Scheduling Iteration. The aggregate demand for L1-period $k = 1$ is 185 kbbl of P1, 190 kbbl of P2, 195 kbbl of P3, 178 kbbl of P4, and 116 kbbl of P5. Production targets for L1-period $k = 1$ are 86.49 kbbl of P1, 89.84 kbbl of P2, 165 kbbl of P3, 129.04 kbbl of P4, and 116 kbbl of P5. As aggregated demand for L1-period $k = 2$ is zero, production targets for that L1-period are zero as well. Production targets are computed as the minimum amount to fulfill aggregated demand and target inventories (see Eq. 65 from Part I of this article). Solving the first-level model (Step 4), which is a LP in this example, provides the blend recipes shown in Table 4. Only one set of blend recipes is required for the entire horizon as there is no production in period $k = 2$. The blend cost is $\text{BlendCost}_{L1} = 14{,}692.13 \times 10^3$ \$.

The second-level model has nine L2-periods (i.e., $m = 1$ to $m = 7$ are the first 7 days, $m = 8$ contains the first 22 h of day 8, and $m = 9$ contains the last 2 h of day 8) and is solved using the blend recipes from Table 4. The blend planning solution does not contain inventory infeasibilities (i.e., all slack variables have a value equal to zero), and the solution from the approximate scheduling (Step 9) is shown in Table 5. Blend cost at the second level is $\text{BlendCost}_{L2} = 14{,}692.13 \times 10^3$ \$, the same as the one from the first level. The second level determines that tank PT-102 must hold product P5 for the entire horizon and is the only product transition required in the storage tanks (tank PT-102 is initially empty but assigned to hold product P3).

Note that product P3 is being processed in different blenders (see Table 5), but it may be blended in the same unit; there is no penalty for this situation as both blenders are allowed to produce such product and in both blenders it represents a new blend run and a product transition (i.e., it incurs in the same penalty). Because there is only one product being blended in each L2-period (i.e., only one task being executed within the L2-periods), feasibility of the blend recipes and the blend plan is guaranteed at the third level; however, we still require to determine if the orders can be delivered on time.

The duration of the L3-periods is 1 h; thus, the complete scheduling horizon has 192 L3-periods. The *L*-intervals at Step 10 are defined as each single day; that is, there are eight *L*-intervals, each one with 24 L3-periods. The third-level solution at Step 11 shows that all slack variables have a value equal to zero. The production and delivery schedules computed at Step 14 are shown in Figure 9. The blend cost at the third level is $\text{BlendCost}_{L3} = 14{,}692.13 \times 10^3$ \$, which is identical to the second-level blend cost. Step 15–17 of the MPIP algorithm can be omitted because there is no product being produced in the same blender in adjacent L2-periods.

It could be argued that, if both blend runs of product P3 are being processed in blender A, a single *L*-interval spanning [48 h, 168 h] should be able to merge those blend runs into a single one. However, the second level already has determined that this is not possible (at least with the blend recipes being used) as the second level could not find a solution where P3 can be blended in adjacent L2-periods, even when there is blend capacity available.

Test Set #2 Results. Table 6 summarizes the results for the Test Set #2. Although we implemented the full-space continuous-time model[4] in GAMS and solved all the examples shown in Table 6 in our machine, we chose to present the reported final solution from Li and Karimi[4] because their execution times are shorter than those required by our machine; however, we show the results that we obtained in our machine for the first integer solution. For examples 3, 4, 7, and 8 from Li and Karimi,[4] the MPIP scheduling algorithm computed solutions with the same blend cost and the same total number of product transitions (i.e., the same total cost). For Test Set #2, we use the definition for product transitions by Li and Karimi[4] as the number of blend runs plus the product transitions in the swing tanks. The same or a smaller number of product transitions was found for examples 9, 12, and 14. Please note that in examples 9, 12, and 14 from Li and Karimi,[4] there seems to be an inconsistency between the data describing the examples and their reported solution because their reported value of the objective function is smaller than the lower bound which is computed as the optimal solution of the aggregate blend problem. As the solution of the aggregate problem is based on the relaxed problem that only minimizes the cost of materials subject to their availability, and inventory and production capacity constraints (i.e., the aggregate problem does not consider the cost of switchovers and their associated constraints), this solution is a lower bound on the optimum. Any results with

Table 5. Test Set #2, Example 12—Blend Plan at the Second Level

L2-period				Blender	
ID	Start Time (h)	End Time (h)	Product	A	B
$m = 1$	0	24	P2	–	89.84
$m = 2$	24	48	P4	–	129.04
$m = 3$	48	72	P3	–	78.94
$m = 5$	96	120	P5	116.00	–
$m = 7$	144	168	P3	86.06	–
$m = 8$	168	190	P1	86.49	–

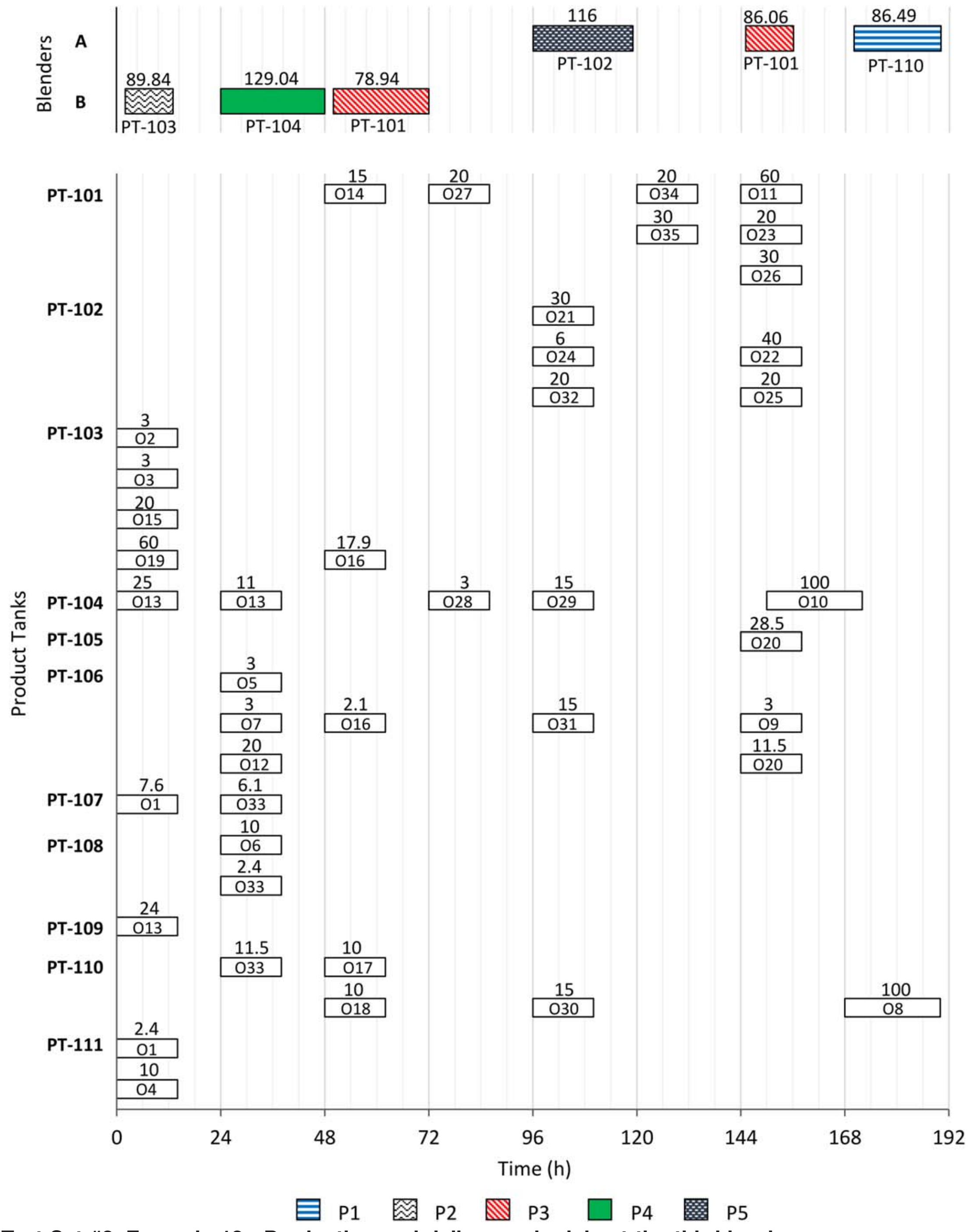

Figure 9. Test Set #2, Example 12—Production and delivery schedule at the third level.

Units in kbbl. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

even lower value of the objective function correspond to infeasible solutions. Following this reasoning, it appears that Li and Karimi[4] solutions for example 9, 12, and 14 are infeasible; therefore, it is not meaningful to compare the quality of the reported solutions for these examples with those obtained by the MPIP algorithm.

For large problems, the execution times required by the MPIP scheduling algorithm are considerably smaller than the times required by the continuous-time model. As expected, full-space continuous-time model is faster for small size problems when compared to MPIP algorithm; however, for large size problems the continuous-time model requires a significant amount of CPU time to find an initial integer solution.

It is important to note that only one scheduling iteration of the MPIP algorithm is required to solve all examples from Test Set #2; moreover, it is not necessary to overlap *L*-intervals and solve Steps 15–17 to obtain the optimal solution due to few blend runs required to meet the demand. Table 7 shows the model size at each level. It can be seen that the number of discrete variables per *L*-interval with 24 L3-periods is greater than the number required by the continuous-time full-space model in Example 3, but it is smaller for all the remaining examples.

Table 6. Test Set #2—Comparison between the MPIP scheduling Algorithm and a Full-Space Continuous-Time Model

					Full-Space Continuous-Time Model[4] (MILP: CPLEX)					MPIP scheduling Algorithm (LP, MILP: CPLEX)			
					Reported Final Solution[a]			First Integer Solution[b]		Final Solution			
Example ID	# Pinch Points	# Blenders	# Orders	Comp. Supply	Obj. Func. (×10^3 $)	# Trans.[c]	CPU Time (s)	Obj. Func. (×10^3 $)	CPU Time (s)	Obj. Func.[d] (×10^3 $)	Blend Cost (×103 $)	# Trans.[c]	CPU Time (s)
3	1	1	12	C	3,159.1	1	2.6	3,392.3	0.67	3,159.1	3,139.1	1	14.7
4	1	1	15	C	4,556.7	1	4.2	4,820.2	0.33	4,556.7	4,536.7	1	15.3
7	1	1	20	C	8,100.4	3	10,802	11,557.1	108.8	8,100.4	8,040.4	3	16.2
8	1	2	20	C	8,080.4	2	10,819	8,815.8	17.6	8,080.4	8,040.4	2	19.1
9	1	2	23	C	10,573.7	4	10,814	14,134.0	120.2	10,777.2	10,702.7	4	35
12	1	2	35	C	14,764.9	5	46,800	16,191.1	42,619	14,786.6	14,692.1	5	44.5
14	0	3	45	C	17,737.4	9	118,800	23,265.8	117,928	20,381.0	20,286.5	5	372.2
3	1	1	12	I	3,159.1	1	1.2	3,368.6	0.78	3,159.1	3,139.1	1	14.2
4	1	1	15	I	4,556.7	1	2.0	5,988.2	0.94	4,556.7	4,536.7	1	16.2
7	1	1	20	I	8,100.4	3	10,814	9,671.7	67.8	8,100.4	8,040.4	3	17
8	1	2	20	I	8,082.9	2	14,170	8,657.6	614.6	8,080.4	8,040.4	2	17.3
9	1	2	23	I	10,573.7	4	10,817	11,483.3	1,138	10,778.8	10,704.3	4	33.2
12	1	2	35	I	14,809.4	7	46,800	15,363.5	12,671	15,241.7	15,147.2	5	128.7
14	0	3	45	I	17,737.4	9	118,800	23,856.2	116,354	21,121.4	21,046.9	4	333.2

[a]Reported values by Li and Karimi[4] where problems were solved using CPLEX 12.1 on a Dell Precision PWS690 computer, Intel Xeon CPU, 3GHz, and 16GB RAM.
[b]Values computed by implementing Li and Karimi[4] model in our machine.
[c]Total number of product transitions in the blenders and storage tanks.
[d]Values computed considering the objective function and associated coefficients from Li and Karimi[4]. C = Constant, I = Irregular.

Table 7. Test Set #2—Model Size at Each Level

	First Level		Second-Level Optimization Phase			Third-Level Optimization Phase			Li and Karimi[4]
						For Each L-Interval with 24 L3-Periods			
Example ID	# Equations	# Continuous Variables	# Equations	# Continuous Variables	# Discrete Variables	# Equations	# Continuous Variables	# Discrete Variables	# Discrete Variables
3	423	237	3632	3087	752	6485	10,711	264	246
4	487	237	3695	3117	752	5659	12,399	264	285
7	487	237	3800	3167	752	6872	15,583	264	672
8	487	237	5348	4184	1174	10,212	17,298	528	595
9	588	280	6570	5045	1460	11,421	19,390	528	635
12	588	280	7008	5295	1516	11,719	26,398	528	2237
14	316	148	8078	5828	1794	17,232	34,867	792	3664
3	423	237	3632	3087	752	6485	10,711	264	246
4	487	237	3695	3117	752	7506	10,354	264	285
7	487	237	3800	3167	752	6872	15,583	264	986
8	487	237	5348	4184	1174	10,620	14,748	528	949
9	588	280	6570	5045	1460	11,421	19,398	528	1015
12	588	280	7008	5295	1516	11,837	26,326	528	3050
14	316	148	8078	5828	1794	17,232	34,867	792	3989

Conclusions

This works introduces an inventory pinch-based, multi-scale algorithm (MPIP scheduling algorithm) to solve integrated gasoline blend planning and scheduling problems. Integrated planning and scheduling is decomposed via three levels, each of them representing a different view of the problem: the first level determines the best blend recipes by considering only the aggregated availability of resources and demand requirements across the planning and scheduling horizon. The second level first computes an optimal blend plan (how much to produce and when) using finer grid time periods; if there is no feasible blend plan based on the blend recipes computed at the first level, an interval at the first level is subdivided at the point of earliest infeasibility and the blend recipes are reoptimized. Once the existence of an optimal blend plan is confirmed, an approximate schedule is optimized. Volumetric constraints and resource allocation from the approximate schedule are constraints on the detailed schedule which is computed at the third level.

The MPIP scheduling algorithm has been tested on two sets of case studies. Test Set #1 includes problems with limited storage and blending capacity, and high product demands. These case studies are tightly constrained and hence are expected to be computationally demanding. Test Set #2 includes examples taken from the literature[2,4] which are not so tightly constrained (e.g., the blending rates have a wider range between minimum and maximum, there is a large initial inventory of products) and a simple demand profile. For Test Set #2, the MPIP scheduling algorithm computed schedules which have the same or better blend cost and same or better number of product transitions as those provided by a full-space continuous-time model, and in significantly shorter execution times when solving large examples. In several large examples from Test Set #2, the results reported in the literature are lower than the aggregate lower bound, indicating an inconsistency between the problem descriptions and the results reported in the literature.

The scheduling model (i.e., third-level model) presented here only requires one binary variable: the decision to feed product storage tank *j* from blender bl in period *n*. Therefore, the model size only increases with the system structure (i.e., with the number of product tanks or blenders) and the number of time periods in which the horizon is discretized.Using only one binary variable at the third level, delivery rates are not forced to be greater than some minimum value. However, low delivery rates are easy to convert into higher rates as the feasibility of the solution is not affected, although nonintermittent deliveries are not guaranteed.

If the scheduling horizon is 2–4 weeks long, the model at the third level becomes a very large MILP model. As second level imposes constraints via approximate schedule, we have chosen to implement forward and reverse rolling window method as a quick heuristic strategy to obtain a good solution. Results presented in this work demonstrate that MPIP scheduling algorithm solves large scale gasoline blending problems to the same or better solution points and much faster than previously published methods.

Future work will be to solve the refinery planning and scheduling problem using our decomposition framework. Decomposition approaches (e.g., Lagrangian decomposition) will also be examined and integrated into our algorithm, especially at the third level.

Acknowledgment

Support by Ontario Research Foundation and McMaster Advanced Control Consortium is gratefully acknowledged.

Notation

Subscripts

bc =	refers to a variable or parameter related to the blend component tanks
blend =	refers to a variable or parameter related to the blenders
comp =	refers to a variable or parameter related to the transfer of volume between component tanks and blenders
L1 =	refers to a variable or parameter of the first level
L2 =	refers to a variable or parameter of the second level
L3 =	refers to a variable or parameter of the third level
order =	refers to a variable or parameter related to the orders
pool =	refers to a variable or parameter related to the product pools
pr =	refers to a variable or parameter related to the individual product tanks
trans =	refers to a variable or parameter related to the transfer of volume between blenders and product tanks or pools

Superscripts

max =	refers to a maximum value that a variable may have
min =	refers to a minimum value that a variable may have if different from zero
start =	refers to the initial value at the beginning of the planning horizon that a variable may have
target =	refers to a target value for a variable

Parameters

$\mathrm{Cost}_{\mathrm{bc}}(i)$ =	cost of blend component *i*
$D^{\max}_{\mathrm{order}}(o)$ =	maximum delivery rate of order *o*
$D^{\max}_{\mathrm{pr}}(j)$ =	maximum delivery rate of tank *j*
$\mathrm{Demand}(o)$ =	demand of order *o* for the complete scheduling horizon
$\mathrm{Demand}_{\mathrm{order,L3}}(o,l)$ =	demand of order *o* in *L*-interval *l*
$F_{\mathrm{bc}}(i,\alpha)$ =	supply flow rate of blend component *i* for supply profile α
$F^{\max}_{\mathrm{blend}}(\mathrm{bl})$ =	maximum blending rate of blender bl
$F^{\min}_{\mathrm{blend}}(\mathrm{bl})$ =	minimum blending rate of blender bl
H =	length of the planning horizon
$\mathrm{it}^{\min}_{\mathrm{blend}}(p,\mathrm{bl})$ =	minimum idle time required by blender bl before processing product *p*
$\mathrm{Penalty}_{\mathrm{bc,L1}}$ =	penalty for the inventory slack variables of blend component tanks at the first level
$\mathrm{Penalty}_{\mathrm{bc,L2}}(m)$ =	penalty for the inventory slack variables of component *i* in L2-period *m*
$\mathrm{Penalty}_{\mathrm{bc,L3}}(n)$ =	penalty for the inventory slack variables of blend component tanks in L3-period *n*
$\mathrm{Penalty}_{\mathrm{pool,L1}}$ =	penalty for the inventory slack variables of product pools at the first level
$\mathrm{Penalty}_{\mathrm{pool,L2}}(m)$ =	penalty for the inventory slack variables of product pool *p* in L2-period *m*
$\mathrm{Penalty}_{\mathrm{pr,L2}}(m)$ =	penalty for the inventory slack variables of product tank *j* in L2-period *m*
$\mathrm{Penalty}_{\mathrm{pr,L3}}(n)$ =	penalty for the inventory slack variables of product tanks in L3-period *n*
$\mathrm{PenaltyBL}_{\mathrm{L3}}$ =	penalty for processing a product in a blender during a L3-period *n*
$\mathrm{PenaltyBR}_{\mathrm{L2}}(\mathrm{bl})$ =	penalty for a blend run processed in blender bl during a L2-period *m*
$\mathrm{PenaltyBS}_{\mathrm{L2}}$ =	penalty for a product transition in a blender at the second level
$\mathrm{PenaltyBsw}_{\mathrm{L3}}$ =	penalty for starting a blend run at the third level
$\mathrm{PenaltyD}_{\mathrm{pr,L3}}$ =	penalty for the delivery slack variables at the third level
$\mathrm{PenaltyDpm}_{\mathrm{L3}}(o,n)$ =	penalty for late delivery of order *o* at the third level
$\mathrm{PenaltyDsw}_{\mathrm{L3}}$ =	penalty for irregular delivery rates at the third level
$\mathrm{PenaltyTS}(j)$ =	penalty for a product transition in a swing tank at the second level

$\mathrm{PenaltyTsw}_{\mathrm{L3}}$	= penalty for changing the destination tank for the product in the blender
$sw_{\mathrm{L3}}^{\mathrm{Est}}(l)$	= parameter that indicates the minimum possible number of blend runs during L-interval l
$t_{\mathrm{blend}}^{\min}(p,\mathrm{bl})$	= minimum running time required by blender bl when processing product p
$t_{\mathrm{L3}}(n)$	= duration of L3-period n
$u^{\mathrm{start}}(j,p)$	= product p stored in tank j at the beginning of the planning horizon
$V_{\mathrm{bc}}^{\max}(i)$	= maximum holdup of tank with blend component i
$V_{\mathrm{bc}}^{\min}(i)$	= minimum holdup of tank with blend component i
$V_{\mathrm{bc}}^{\mathrm{start}}(j,p)$	= volume of blend component i stored at the beginning of the planning horizon
$V_{\mathrm{pr}}^{\max}(j)$	= maximum holdup of tank j
$V_{\mathrm{pr}}^{\min}(j)$	= minimum holdup of tank j
$V_{\mathrm{pr}}^{\mathrm{start}}(j)$	= volume stored in tank j at the beginning of the planning horizon
$\mathrm{VMIN}_{\mathrm{blend}}(p,\mathrm{bl})$	= minimum volume allowed to blend of product p in blender bl during each L2-period

Integer variables

$t_{\mathrm{blend,L2}}(p,\mathrm{bl},m)$	= estimated time to process product p in blender bl in L2-period m
$u_{\mathrm{L2}}(j,p,m)$	= binary variable that indicates if tank j is storing product p in L2-period m
$ue_{\mathrm{L2}}(j,m)$	= binary variable that indicates if there is a product transition in tank j at the beginning of L2-period m
$v_{\mathrm{L3}}(j,p,\mathrm{bl},n)$	= binary variable that indicates if tank j is receiving product p from blender bl in L3-period n
$x_{\mathrm{L2}}(p,\mathrm{bl},m)$	= binary variable that indicates if product p is processed in blender bl in L2-period m

Continuous variables

$\mathrm{BlendCost}_{\mathrm{L1}}$	= total blend cost at the first level
$\mathrm{BlendCost}_{\mathrm{L2}}$	= total blend cost at the second level
$\mathrm{BlendCost}_{\mathrm{L3}}$	= total blend cost at the third level
$D_{\mathrm{pr,L3}}(j,o,n)$	= delivery rate of tank j for order o within L3-period n
$F_{\mathrm{blend,L3}}(p,\mathrm{bl},n)$	= blending rate of blender bl to produce product p during L3-period n
$\mathrm{it}_{\mathrm{blend,L3}}(\mathrm{bl},n)$	= cumulative idle time of blender bl in L3-period n
$of_{\mathrm{L2}}(o,m)$	= fraction of order o to be delivered during L2-period m (it becomes a parameter at the third level model)
$\mathrm{Dsw}_{\mathrm{L3}}(j,o,n)$	= difference between delivery rates to be penalized in objective function of the third level optimization phase
$r(i,p,k)$	= volume of blend component i into product p in L1-period k (it becomes a parameter at the second and third level model)
$S_{\mathrm{bc,L1}}^{+}(i,k)$	= positive inventory slack variable of blend component i at L1-period k
$S_{\mathrm{bc,L1}}^{-}(i,k)$	= negative inventory slack variable of blend component i at L1-period k
$S_{\mathrm{bc,L2}}^{+}(i,m)$	= positive inventory slack variable of blend component i at L2-period m
$S_{\mathrm{bc,L2}}^{-}(i,m)$	= negative inventory slack variable of blend component i at L2-period m
$S_{\mathrm{bc,L3}}^{+}(i,n)$	= positive inventory slack variable of blend component i at L3-period n
$S_{\mathrm{bc,L3}}^{-}(i,n)$	= negative inventory slack variable of blend component i at L3-period n
$S_{\mathrm{order,L3}}^{+}(o,l)$	= positive delivery slack variable of order o at L-interval l
$S_{\mathrm{order,L3}}^{-}(o,l)$	= negative delivery slack variable of order o at L-interval l
$S_{\mathrm{pool,L1}}^{+}(p,k)$	= positive inventory slack variable of product pool p at L1-period k
$S_{\mathrm{pool,L1}}^{-}(p,k)$	= negative inventory slack variable of product pool p at L1-period k
$S_{\mathrm{pool,L2}}^{+}(p,m)$	= positive inventory slack variable of product pool p at L2-period m
$S_{\mathrm{pool,L2}}^{-}(p,m)$	= negative inventory slack variable of product pool p at L2-period m
$S_{\mathrm{pr,L1}}^{+}(i,k)$	= positive inventory slack variable of product tank j at L1-period k
$S_{\mathrm{pr,L1}}^{-}(i,k)$	= negative inventory slack variable of product tank j at L1-period k
$S_{\mathrm{pr,L2}}^{+}(i,m)$	= positive inventory slack variable of product tank j at L2-period m
$S_{\mathrm{pr,L2}}^{-}(i,m)$	= negative inventory slack variable of product tank j at L2-period m
$S_{\mathrm{pr,L3}}^{+}(i,n)$	= positive inventory slack variable of product tank j at L3-period n
$S_{\mathrm{pr,L3}}^{-}(i,n)$	= negative inventory slack variable of product tank j at L3-period n
$sw_{\mathrm{L3}}(\mathrm{bl},n)$	= 0–1 continuous variable that indicates if a blend run has started in blender bl at the beginning of L3-period n
$t_{\mathrm{blend,L3}}(\mathrm{bl},n)$	= cumulative running time of blender bl in L3-period n
$V_{\mathrm{bc,L2}}(i,m)$	= volume stored in component tank i at the end of L2-period m
$V_{\mathrm{bc,L3}}(i,n)$	= volume stored in component tank i at the end of L3-period n
$V_{\mathrm{comp,L1}}(i,p,k)$	= volume of blend component i into product p in L1-period k
$V_{\mathrm{comp,L2}}(i,p,\mathrm{bl},m)$	= volume of blend component i into product p in blender bl in L2-period m
$V_{\mathrm{comp,L3}}(i,p,\mathrm{bl},n)$	= volume of blend component i into product p in blender bl in L3-period n
$V_{\mathrm{pr,L2}}(j,m)$	= volume stored in product tank j at the end of L2-period m
$V_{\mathrm{pr,L3}}(j,n)$	= volume stored in product tank j at the end of L3-period n
$V_{\mathrm{trans,L3}}(j,p,\mathrm{bl},n)$	= volume transferred of product p from blender bl to tank j in L3-period n
$vc_{\mathrm{blend,L3}}(\mathrm{bl},n)$	= cumulative volume produced by blender bl during a blend run up to period n
$ve_{\mathrm{L3}}(\mathrm{bl},n)$	= 0–1 continuous variable which indicates if the product from blender bl is going to be sent to a different storage tank than that of the previous L3-period
$w_{\mathrm{blend,L3}}(\mathrm{bl},n)$	= 0–1 continuous variable which indicates that blender bl is idle in L3-period n if it is equal to 1
$x_{\mathrm{L3}}(p,\mathrm{bl},n)$	= 0–1 continuous variable that indicates if product p is processed in blender bl in L3-period n
$xe_{\mathrm{L2}}(p,\mathrm{bl},m)$	= 0–1 continuous variable that indicates if a state transition has occurred in blender bl at the beginning of L2-period m
$xe_{\mathrm{L3}}(p,\mathrm{bl},n)$	= 0–1 continuous variable that indicates if a state transition has occurred in blender bl at the beginning of L3-period n
Z_{L1}	= objective function value at the first level
Z_{L2}	= objective function value at the second level, total cost
$Z_{\mathrm{L2}}^{\mathrm{opt}}$	= objective function value at the second level, approximate scheduling
$Z_{\mathrm{L2}}^{\mathrm{feas}}$	= objective function value at the second level, blend planning
Z_{L3}	= objective function value at the third level, total cost
$Z_{\mathrm{L3}}^{\mathrm{feas}}$	= objective function value at the third level, optimization phase
$Z_{\mathrm{L3}}^{\mathrm{opt}}$	= objective function value at the third level, feasibility phase

Literature Cited

1. Jia Z, Ierapetritou M. Mixed-integer linear programming model for gasoline blending and distribution scheduling. *Ind Eng Chem Res.* 2003;42:825–835.
2. Li J, Karimi IA, Srinivasan R. Recipe determination and scheduling of gasoline blending operations. *AIChE J.* 2010;56:441–465.
3. Mendez CA, Grossman 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.
4. Li J, Karimi IA. Scheduling gasoline blending operations from recipe determination to shipping using unit slots. *Ind Eng Chem Res.* 2011;50:9156–9174.
5. Sousa RT, Liu S, Papageorgiou LG, Shah N. Global supply chain planning for pharmaceuticals. *Chem Eng Res Des.* 2011;89:2396–2409.

6. Pinto JM, Joly M, Moro LFL. Planning and scheduling models for refinery operations. *Comput Chem Eng*. 2000;24:2259–2276.
7. Kelly JD. Logistics: the missing link in blend scheduling optimization. *Hydrocarbon Process*. 2006;85:45–51.
8. Maravelias CT, Sung C. Integration of production planning and scheduling: overview, challenges and opportunities. *Comput Chem Eng*. 2009;33:1919–1930.
9. Maravelias CT. General framework and modeling approach classification for chemical production scheduling. *AIChE J*. 2012;58(6): 1812–1828.
10. Mendez CA, Cerda J, Grossmann IE, Harjunkoski I, Fahl M. State-of-the-art review of optimization methods for short term scheduling of batch processes. *Comput Chem Eng*. 2006;30:913–946.
11. Floudas CA, Lin X. Continuous-time vs discrete-time approaches for scheduling of chemical processes: a review. *Comput Chem Eng*. 2004;28:2109–2129.
12. Sundaramoorthy A, Maravelias CT. Computational study of network-based mixed-integer programming approaches for chemical production scheduling. *Ind Eng Chem Res*. 2011;50:5023–5040.
13. Joly M, Pinto JM. Mixed-integer programming techniques for the scheduling of fuel oil and asphalt production. *Inst Chem Eng*. 2003; 81:427–447.
14. Bassett MH, Pekny JF, Reklaitis GV. Decomposition techniques for the solution of large-scale scheduling problems. *AIChE J*. 1996; 42(12):3373–3387.
15. Elkamel A, Zentner M, Pekny F, Reklaitis GV. A decomposition heuristic for scheduling the general batch chemical plant. *Eng Optim*. 1997;28(4):299–330.
16. Munawar SA, Gudi RD. A multi-level, control-theoretic framework for integration of planning, scheduling and rescheduling. *Ind Eng Chem Res*. 2005;44:4001–4021.
17. Li Z, Ierapetritou MG. Integrated production planning and scheduling using a decomposition framework. *Chem Eng Sci*. 2009;64:3585–3597.
18. Mouret S, Grossmann IE, Pestiaux P. A new Lagrangian decomposition approach applied to the integration of refinery planning and crude-oil scheduling. *Comput Chem Eng*. 2011;35:2750–2766.
19. Glismann K, Gruhn G. Short-term scheduling and recipe optimization of blending processes. *Comput Chem Eng*. 2001;25:627–634.
20. Castillo PAC, Kelly JD, Mahalec V. Inventory pinch algorithm for gasoline blend planning. *AIChE J*. 2013;59:3748–3766.
21. Singhvi A, Shenoy UV. Aggregate planning in supply chains by pinch analysis. *Trans IChemE A*. 2002;80:597–605.

Manuscript received Oct. 24, 2013, and revision received Feb. 17, 2014.