Article Post-Print

Inventory Pinch Gasoline Blend Scheduling Algorithm Combining Discrete- and Continuous-Time Models

Pedro A. Castillo-Castillo, Vladimir Mahalec*

Department of Chemical Engineering, McMaster University, 1280 Main St. West, Hamilton, ON, L8S 4L8, Canada

* Corresponding author. Tel.: +1 905 525 9140 ext. 26386. E-mail address: mahalec@mcmaster.ca

Keywords: Inventory pinch; nonlinear gasoline blending; discrete-time models; continuous-time models

Abstract

This work introduces multi-period inventory pinch-based algorithm to solve continuous-time scheduling models (MPIP-C algorithm), a three level method which combines discrete-time approximate scheduling with continuous-time detailed scheduling and with inventory pinch-based optimization of operating states. When applied to gasoline blending, the top level computes optimal recipes for aggregated blends over periods initially delineated by inventory pinch points. Discrete-time middle level uses fixed blend recipes to compute an approximate schedule, i.e. what, when, and how much to produce; it also allocates swing storage and associated product shipments with specific storage. Continuous-time model at the third level computes when exactly to start/stop an operation (blend, tank transfer, shipment). MPIP-C algorithm solves linear or nonlinear problems two to three orders of magnitude faster than full-space models.

1. Introduction

Process industries' supply chains are comprised of facilities and activities which procure raw materials to the plants, and store and distribute finished products to the customers. Structure of the supply chains (location, capacity, and type of each plant and storage) is decided upon by optimizing returns over long time horizons, e.g. five or ten years. Once the structure of the supply chain is fixed, one needs to decide the best way to operate it. Since supply chains are large systems, and there are many decisions to be made, it is customary to optimize supply chain operations over different lengths of the time horizon and at different levels of model accuracy. When optimizing operation over long time horizons, e.g. over one year, it is important to

determine how much of each product will be produced from one season to another, while it is not important to decide at what specific time some equipment will change from one operation mode to another. The latter is a detailed decision which can be made e.g. one or two weeks prior to that day.

Scheduling is an activity that determines the operating states and their sequence for each equipment, amount of feed to be processed by each instance of an operating state, amount and time of shipment of each product, allocation of multipurpose storage to a given service, and others. Scheduling models for oil refinery operations are usually mixed-integer linear programs (e.g. Gothe-Lundgren et al., 2002; Jia & Ierapetritou, 2003, 2004; Li & Karimi, 2011) even if the underlying processes are nonlinear by nature, in order to decrease the computational burden. Some MINLP models have been published recently for the detailed scheduling of crude oil operations (Li et al., 2012) and the pooling problem (Kolodziej et al., 2013); they have been solved via the latest generation of specialized MINLP solvers. Currently, most scheduling models are usually formulated using a continuous-time representation (i.e. the horizon is divided into several time slots which duration is a variable to determine) since it requires a smaller number of discrete variables compared with the corresponding discrete-time model. One example of commercial scheduling software is Aspen Petroleum Scheduler (Aspen Technology).

Inclusion of sequencing terms in the model formulation makes scheduling a very challenging computational problem. Due to the inherent discrete decisions involved in the scheduling problem, it is at least NP-complete (Birewar & Grossmann, 1990; Pinto et al., 2000) and frequently NP-hard (Terrazas-Moreno & Grossmann, 2011); this means that there are no polynomial time bounded algorithms to solve this type of problems. Within this current paradigm, it is not possible to guarantee computation of optimal solutions in a reasonable amount of time by a general algorithm when scheduling problems grow beyond a certain model size (which depends on the problem type and instance). Most successful solution strategies are those that employ algorithms tailored to a specific class of scheduling problems and scale up to solve large problems within acceptable computational times. Necessity for such approaches has been presented as “no free lunch theorems for optimization” by Wolpert and Macready (1997).

Decomposition strategies are usually employed to handle large-scale scheduling problems. Bassett et al. (1996) presented various heuristic methods for discrete-time formulations of scheduling problems, most of them using time-based decompositions. Wu and Ierapetritou (2003) reviewed several heuristic approaches for scheduling problems with continuous-time models, including methods using time- and resource-based decompositions. They also included a review of rigorous mathematical approaches such as Lagrangean relaxation and Lagrangean decomposition.

A major part of an oil refinery’s profit comes from gasoline (Mendez et al., 2006; Li & Karimi, 2011). Therefore, determining the best possible way to mix the refinery’s intermediate products (blend components) to produce the different gasoline grades is an important task. For that reason, inclusion of recipe optimization in gasoline blend scheduling models has become the norm in the last ten years. Both discrete- and continuous-time formulations have been developed

to solve the gasoline blend scheduling problem. In order to develop a mixed-integer linear model, quality properties are transformed into blend indices that blend linearly on a volumetric or weight basis (e.g. Li et al., 2010; Li & Karimi, 2011). A different approach is to solve a sequence of MILP models in order to converge to the proper quality values (e.g. Mendez et al., 2006). As well, stochastic methods have been proposed to solve nonlinear recipe optimization problems (e.g. Chen & Wang, 2010; Zhao & Wang, 2011); however, no operational features nor logistic constraints are considered in such cases.

Commercial MILP solvers may require several hours to obtain the optimal solution of detailed scheduling models for medium- and large-scale problems (e.g. Li & Karimi, 2011; Shah & Ierapetritou, 2011). Castillo and Mahalec (2015) presented a detailed continuous-time scheduling model with reduced number of discrete variables, and although such model was able to solve some large-scale examples to optimality in less time than previously published models, execution times still can reach more than 12 hours.

Glismann and Gruhn (2001) proposed a two-level approach to solve the blend scheduling problem: at the top level, a discrete-time NLP model computes blend recipes and production targets, and at the lower level, a discrete-time MILP model solves the short term scheduling problem using the recipes and targets from the top level. An iterative procedure is required to handle possible infeasibilities. The scheduling model is based on a resource-task-network representation. The scheduling model does not consider multipurpose tanks (i.e. swing tanks) nor the delivery scheduling problem (i.e. distribution or shipping problem).

Castillo and Mahalec (2014b) presented a three-level decomposition approach to solve the gasoline blending problem using discrete-time models at each level. They included recipe optimization using linear or nonlinear models, blend size threshold constraints, and most of the operational features described by Li and Karimi (2011). The discrete-time formulation for scheduling horizons of 1 or 2 weeks with 1-hour time periods resulted in a large size scheduling model at the 3rd level that required to be solved in subintervals. These subintervals were solved sequentially, first in a forward direction and then in a reverse direction. Solutions computed by this approach were better and the execution times for large problems were two orders of magnitude shorter than those from previous works (Li et al., 2010; Li & Karimi, 2011). For most of their nonlinear examples, their algorithm computed better solutions than MINLP solvers BARON, ANTIGONE, and GloMIQO, with execution times an order of magnitude shorter. Analogously to Li and Karimi (2011), Castillo and Mahalec (2014b) did not penalize the delivery of the same product order from different tanks.

This work introduces Multi-Period Inventory Pinch-Continuous time algorithm (MPIP-C) for scheduling of processes described by linear and nonlinear models. MPIP-C algorithm decomposes the original scheduling problem into (i) blend recipe optimization, (ii) approximate scheduling, and (iii) detailed scheduling. The 1st level determines the blend recipes (by solving a discrete-time LP or NLP model), and the 2nd level computes an approximate schedule via discrete-time MILP. The time periods at the 1st level are initially delineated by inventory pinch points (Castillo et al., 2013; Castillo & Mahalec, 2014a). Discrete-time model for approximate

scheduling is a modification of a model used in our previous work (Castillo & Mahalec, 2014a). The 3rd level model (i.e. the detailed scheduling problem) is a continuous-time MILP which includes additional constraints arising from the approximate scheduling solution. If there are infeasibilities encountered at the 2nd or the 3rd level, they are resolved by subdividing the corresponding periods at the 1st level. The problem does not have a feasible solution if the 1st level is infeasible.

The rest of this article is structured as follows. Section 2 presents the problem statement. Section 3 presents an overview of the algorithm and of the models used at each level. Section 4 describes the examples used in this work. Section 5 discusses the results obtained using the MPIP-C algorithm. As summarized in Section 6, the computational results show that the proposed MPIP-C algorithm computes optimal or near-optimal solutions with execution times which are much smaller (for both linear and nonlinear models) than those required by the full-space model.

2. Problem Statement

The gasoline blend scheduling problem is summarized as follows:

Given a short-term scheduling horizon, a set of blend components and their properties, a supply profile of blend components, a set of products and their property specifications, a set of delivery orders for each product, a blending system (i.e. storage tanks, blenders, and their interconnections) and its initial conditions, determine a) the blend recipes, b) production and delivery sequences, c) inventory profiles, and d) swing tanks product allocation, while minimizing the cost of the blended materials plus the switching costs (i.e. number of blend runs, number of tanks delivering the same order, and product transitions in the swing tanks) and the demurrage costs.

The blending system is subject to the following constraints:

1. If a blender is to produce a product, it must blend at least a minimum amount.
2. A blender can produce at most one product at any time. Once it begins blending, it must operate for some minimum time before it can switch to another product.
3. A blender requires a minimum setup time during a product changeover
4. A blender can feed at most one product tank at any time (industrial practice).
5. Product tanks can only store one product at any time.

The assumptions made are:

1. Flow rate profile of each component from the upstream process is piecewise constant.
2. Component quality profile is piecewise constant.
3. Perfect mixing occurs in each blender.
4. There is only one tank for a given blend component.
5. Only product tanks defined as swing tanks can change its product service (i.e. change from storing one product to store another).

6. Changeover times between products are negligible for swing tanks.
7. For each blender, changeover times between product blending are product-dependent but sequence-independent.
8. Each order involves only one product (one original order involving different products can be broken into orders of each specific product).
9. Each order is completed during the scheduling horizon.

For illustration, Fig. 1 shows a sample blending system with five dedicated component tanks, two blenders, four product tanks (two dedicated tanks and two swing tanks), and three different finished products.

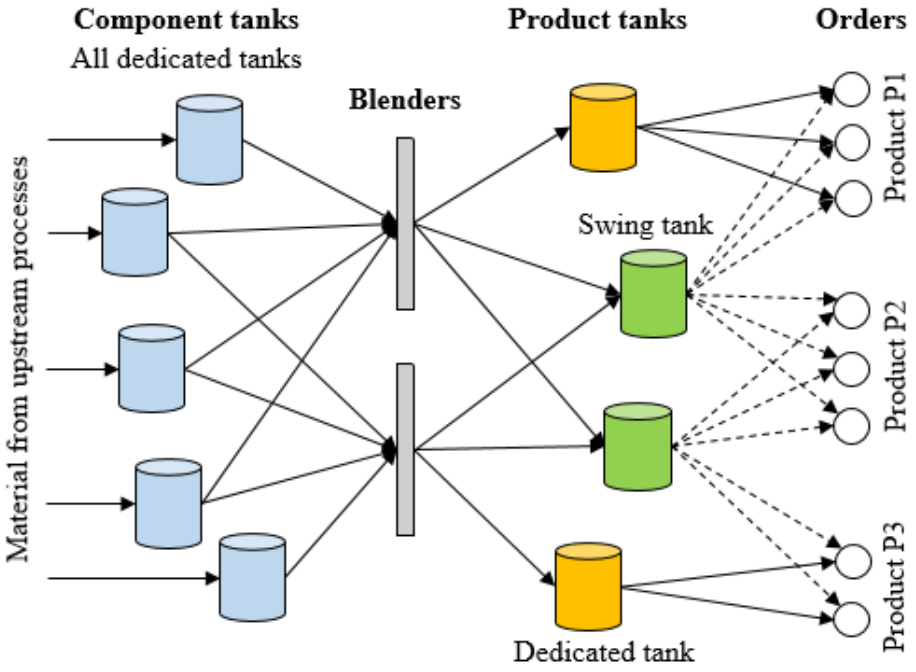

Fig. 1. Sample gasoline blending system.

3. Inventory-pinch based scheduling algorithm combining discrete-time and continuous-time models

This work employs a three-level decomposition of the problem as shown in Fig. 2, which is similar to the one presented by Castillo and Mahalec (2014b) but it includes several important differences. In order to speed-up computations, there is no product sequencing at the 2nd level, and the model at the 3rd level employs a continuous-time formulation instead of a discrete representation of the time domain. Since the proposed algorithm is a continuous-time version of

the multiperiod inventory pinch algorithm, it is denoted as MPIP-C algorithm. In this section we describe our decomposition approach and the equations required at each level. Subscripts *L1*, *L2*, and *L3* identify to which level a variable or parameter corresponds.

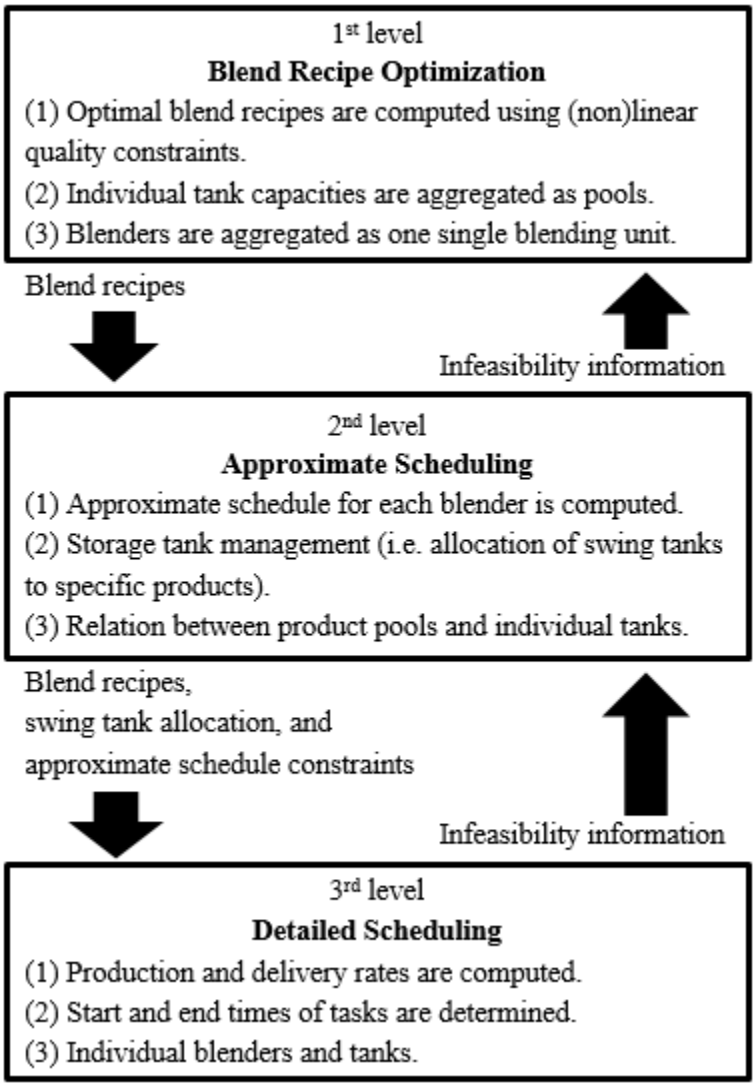

Fig. 2. Decomposition framework for blend scheduling.

3.1. Time grids, time slots, and the inventory pinch concept

The inventory pinch points (see Fig. 3) correspond to the times where the cumulative average total production (CATP) curve changes its slope in order to remain above the cumulative total demand (CTD) curve (Castillo et al., 2013; Castillo & Mahalec, 2014a). Product inventories are at the minimum allowed limits at the inventory pinch points. 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.

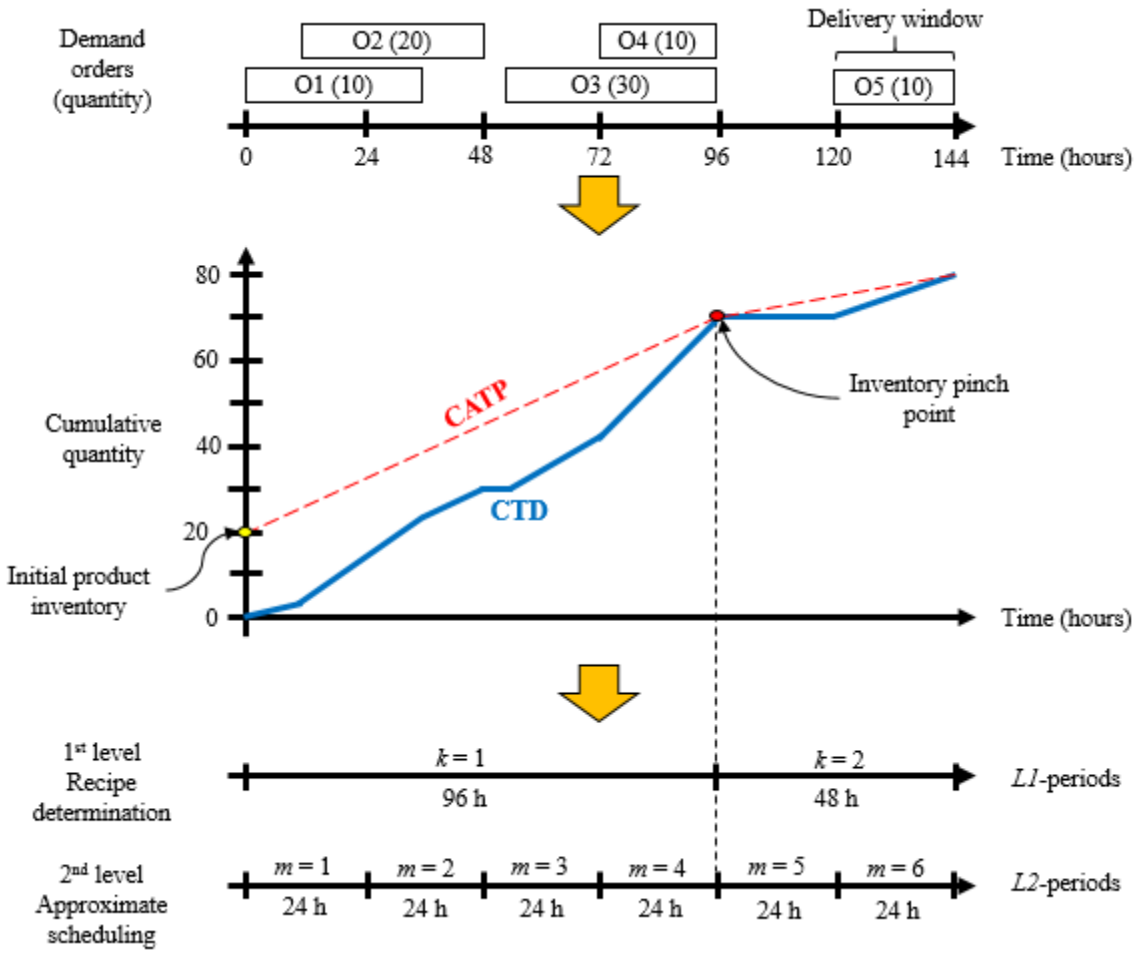

Fig. 3. Example of an inventory pinch point and time grids.

The algorithm starts at the 1st level with determination of the inventory pinch points, which delineate the periods at this level (*L1*-periods). Additional *L1*-periods are delineated by the times when changes in the quality or price of the blend components take place. If there are no inventory constraints, optimal blend recipes between the pinch points remain constant. Once blend recipes are computed at the 1st level, they are used as fixed blend recipes at the 2nd level to compute an approximate schedule via discrete-time MILP model. The set of *L1*-periods is $\mathbf{K} = \{1, \ldots, k, \ldots, K-1, K\}$, and the set of *L2*-periods is $\mathbf{M} = \{1, \ldots, m, \ldots, M-1, M\}$. An integer number of *L2*-periods compose an *L1*-period and this is encoded in set $\mathbf{KM} = \{(k, m)\}$.

L2-periods are chosen based on operational decisions. *L2*-periods must be long enough to complete the smallest allowed blend run and they must be longer than the minimum time that a swing tank may hold a specific product. *L2*-periods are not required to have the same length. Time period boundaries of *L2*-periods are delineated by the period boundaries from the 1st level, points in time when the supply profile of blend components changes, and by clock-time chosen by the scheduler. Additionally, it is important to reduce as much as possible the number of order delivery windows spanning more than one *L2*-period. This reduction of the time delivery

windows will decrease the number of binary variables associated with the delivery of orders (i.e. $z_{L2}(j,o,m)$ and $z_{L3}(j,o,n)$). The following guidelines help to determine if a time delivery window can be reduced:

1. The order can be fulfilled within the adjusted delivery window by a single tank.
2. Adjustments to the time delivery windows should not move existent inventory pinch points nor generate new inventory pinch points (otherwise the minimum blend cost will increase).
3. Based on the previous guideline, delaying the start of a delivery window is preferred instead of shortening the end time of the window.

These guidelines are easy to check without solving an optimization problem. Fig. 4 shows a graphical example where some of the original delivery windows are contracted.

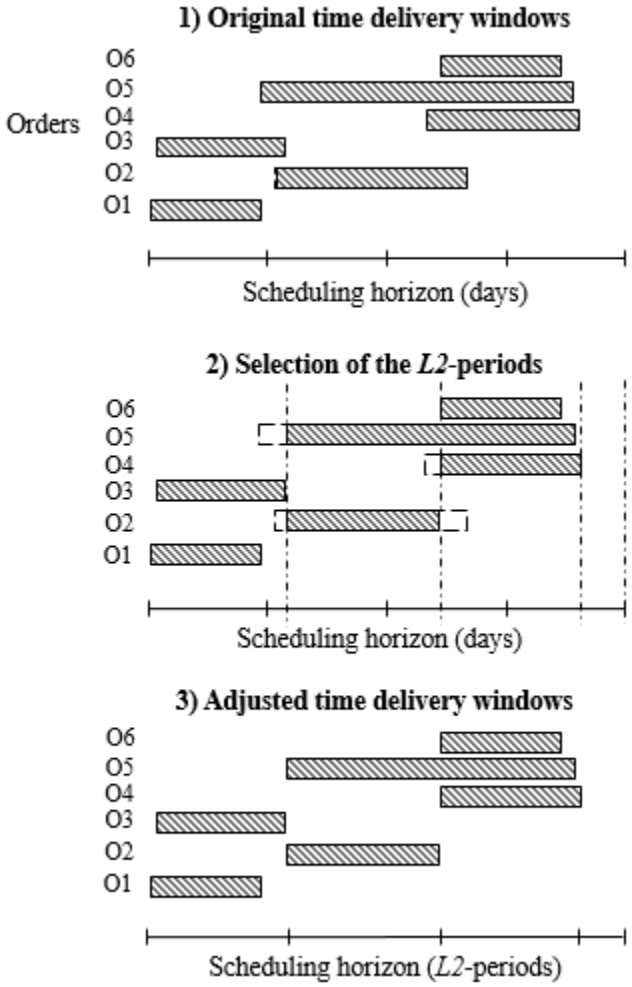

Fig. 4. Time delivery windows can be adjusted according to the *L2*-period boundaries.

The 3rd level model uses a continuous-time formulation with time slots and specific time grids for each unit. The continuous-time model used in this work is the one presented in Castillo and Mahalec (2015) since it considers several key operational constraints without many discrete variables. The model as presented in Castillo and Mahalec (2015) is referred in this work as the full-space model. Following the same principle as with the 1st and 2nd levels, an integer number of time slots is associated with an *L2*-period. Set **MN** = {(*m*, *n*)} is used to represent this information. Eqs. (1)–(3) indicate that the slots associated with *L2*-period *m* should end within the interval of such *L2*-period. Note that the first slot associated with *L2*-period *m* may start during *m* – 1. Fig. 5 shows an example of the time grids at each level.

$$T_{L2}^{end}(m-1) \le T_{bc,L3}(i,n) \le T_{L2}^{end}(m) \qquad \forall i,n \in \mathbf{N1}, m:(m,n) \in \mathbf{MN} \tag{1}$$

$$T_{L2}^{end}(m-1) \le T_{pr,L3}(j,n) \le T_{L2}^{end}(m) \qquad \forall j,n \in \mathbf{N1}, m:(m,n) \in \mathbf{MN} \tag{2}$$

$$T_{L2}^{end}(m-1) \le T_{b,L3}(bl,n) \le T_{L2}^{end}(m) \qquad \forall bl,n \in \mathbf{N1}, m:(m,n) \in \mathbf{MN} \tag{3}$$

In this decomposition framework, the boundaries of the *L2*-periods delineate the boundaries of the fixed time periods of the continuous-time model from Castillo and Mahalec (2015) and they are used to define set **ON** = {(*o*,*n*) | order *o* may be delivered during time slot *n*} and set **JON** = {(*j*,*o*,*n*) | tank *j* may deliver order *o* during time slot *n*}.

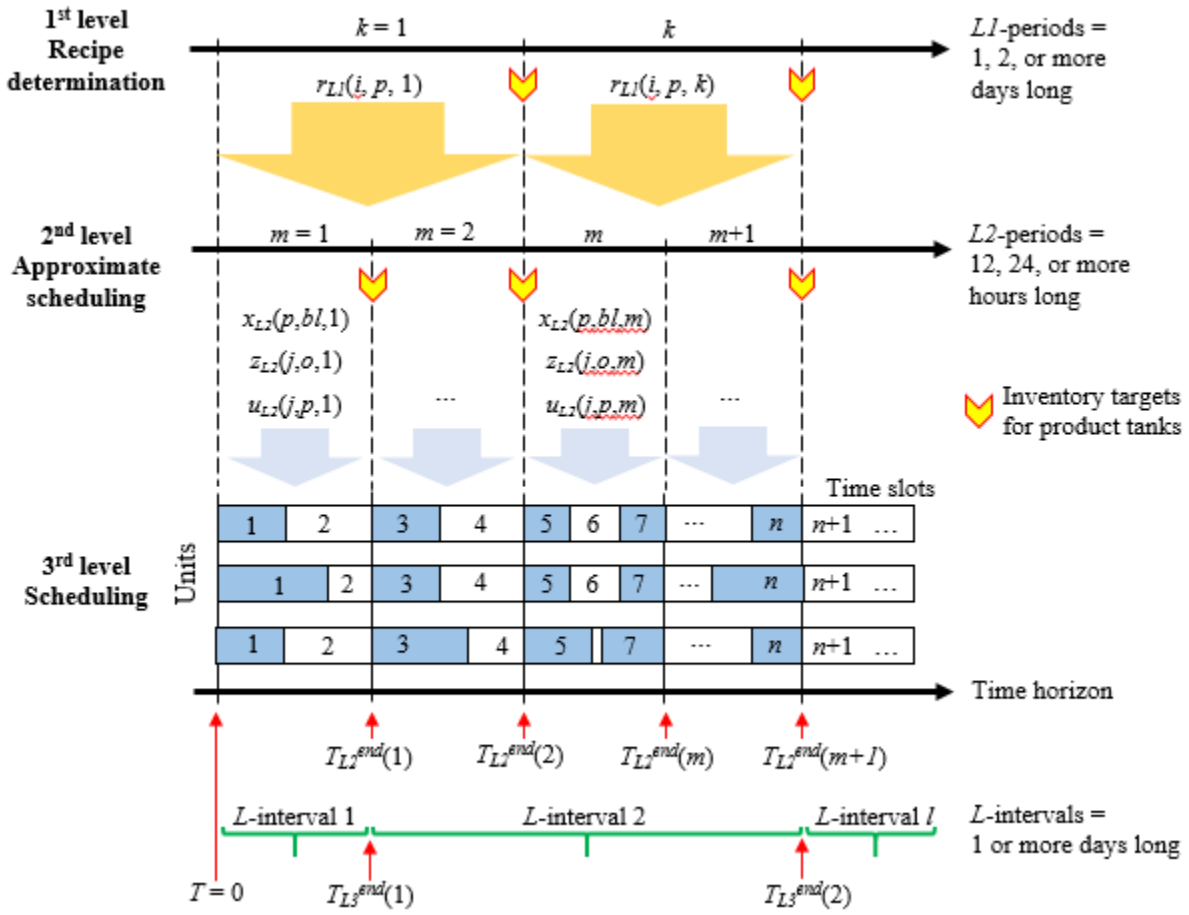

Fig. 5. Graphical example of the proposed decomposition approach.

3.2. Optimizing operating conditions (L1 level)

The 1st level determines which operating conditions are going to be used at the following levels. For the gasoline blending problem, these operating conditions are the blend recipes. The 1st level model uses a discrete-time formulation with no integer variables. Depending on the blending rules being used, this model will be either linear or nonlinear. We use the model presented by Castillo and Mahalec (2014a) which uses the inventory pinch concept to delineate the boundaries of the *L1*-periods, blenders are lumped into a single one, and individual product tanks are modelled as product pools. Only demand, product quality, inventory capacity, and maximum blending capacity constraints are considered. The solution of the 1st level model represents a lower bound for the blend cost of the original problem (Castillo et al., 2013, Castillo & Mahalec, 2014a, 2014b) and it is imposed as such in the 3rd level model.

In this work, some of the examples assume the research octane number (RON) and the motor octane number (MON) as properties that blend nonlinearly following the ethyl RT-70 models (Singh et al., 2000; Healey et al., 1959). Such models define RON and MON as functions of the blend components sensitivity (i.e. $sens(i) = Q_{bc}(i,\text{'RON'}) - Q_{bc}(i,\text{'MON'})$), and olefins and

aromatics content. The model parameter values used by Singh et al. (2000) are: a_1 = 0.03224, a_2 = 0.00101, a_3 = 0, a_4 = 0.04450, a_5 = 0.00081, and a_6 = – 0.0645. Eqs. (4)–(14) are appended to the 1st level model from Castillo and Mahalec (2014a) for those specific examples.

$$r^{RON}_{avg,L1}(p,k) = \sum_i r_{L1}(i,p,k)\cdot Q_{bc}(i,e,k) \qquad \forall e = \text{'RON'},\ p,k \geq 1 \tag{4}$$

$$r^{MON}_{avg,L1}(p,k) = \sum_i r_{L1}(i,p,k)\cdot Q_{bc}(i,e,k) \qquad \forall e = \text{'MON'},\ p,k \geq 1 \tag{5}$$

$$sens_{avg,L1}(p,k) = \sum_i r_{L1}(i,p,k)\cdot sens(i) \qquad \forall i,\ p,k \geq 1 \tag{6}$$

$$sens^{RON}_{avg,L1}(p,k) = \sum_i r_{L1}(i,p,k)\cdot Q_{bc}(i,e,k)\cdot sens(i) \quad \forall e = \text{'RON'},\ p,k \geq 1 \tag{7}$$

$$sens^{MON}_{avg,L1}(p,k) = \sum_i r_{L1}(i,p,k)\cdot Q_{bc}(i,e,k)\cdot sens(i) \quad \forall e = \text{'MON'},\ p,k \geq 1 \tag{8}$$

$$Ol_{avg,L1}(p,k) = \sum_i r_{L1}(i,p,k)\cdot Q_{bc}(i,e,k) \qquad \forall e = \text{'OLF'},\ p,k \geq 1 \tag{9}$$

$$Ar_{avg,L1}(p,k) = \sum_i r_{L1}(i,p,k)\cdot Q_{bc}(i,e,k) \qquad \forall e = \text{'ARO'},\ p,k \geq 1 \tag{10}$$

$$Ol^{sq}_{avg,L1}(p,k) = \sum_i r_{L1}(i,p,k)\cdot [Q_{bc}(i,e,k)^2] \quad \forall e = \text{'OLF'},\ p,k \geq 1 \tag{11}$$

$$Ar^{sq}_{avg,L1}(p,k) = \sum_i r_{L1}(i,p,k)\cdot [Q_{bc}(i,e,k)^2] \quad \forall e = \text{'ARO'},\ p,k \geq 1 \tag{12}$$

$$\begin{aligned} Q_{pr,L1}(p,e,k) = {} & r^{RON}_{avg,L1}(p,k) + a_1\left(sens^{RON}_{avg,L1}(p,k) - r^{RON}_{avg,L1}(p,k)\cdot sens_{avg,L1}(p,k)\right) \\ & + a_2\left(Ol^{sq}_{avg,L1}(p,k) - \left[Ol_{avg,L1}(p,k)\right]^2\right) \\ & + a_3\left(\left[Ar^{sq}_{avg,L1}(p,k)\right]^2 - 2\left[Ar^{sq}_{avg,L1}(p,k)\right]\left[Ar_{avg,L1}(p,k)\right]^2 + \left[Ar_{avg,L1}(p,k)\right]^4\right) \\ & \forall e = \text{'RON'},\ p,k \geq 1 \end{aligned} \tag{13}$$

$$
\begin{aligned}
Q_{pr,L1}(p,e,k) = {} & r^{MON}_{avg,L1}(p,k) + a_4\left(sens^{MON}_{avg,L1}(p,k) - r^{MON}_{avg,L1}(p,k)\cdot sens_{avg,L1}(p,k)\right) \\
& + a_5\left(Ol^{sq}_{avg,L1}(p,k) - \left[Ol_{avg,L1}(p,k)\right]^2\right) \\
& + \left(\frac{a_6}{10000}\right)\left(\left[Ar^{sq}_{avg,L1}(p,k)\right]^2 - 2\left[Ar^{sq}_{avg,L1}(p,k)\right]\left[Ar_{avg,L1}(p,k)\right]^2 + \left[Ar_{avg,L1}(p,k)\right]^4\right) \\
& \forall e = \text{'MON'},\ p,k \geq 1 \qquad (14)
\end{aligned}
$$

3.3. Approximate scheduling model (L2 level)

The 2nd level determines an approximate schedule via a discrete-time MILP model with fixed operating conditions as defined by the 1st level solution. As mentioned in section 3.1, the *L2*-periods are selected depending on operational decisions. The 2nd level model links the product pools with individual tanks, involves decisions such as product allocation of swing tanks, which product and how much to blend in each blender, and which order to deliver from each tank and how much, in each *L2*-period. More than one product can be blended in each blending unit in a given *L2*-period. Constraints such as minimum blend size, minimum running times and setup times for the blenders, inventory limits, and minimum and maximum blending rates are included in this model. Eqs. (16)–(22) are appended to the 2nd level model presented by Castillo and Mahalec (2014a) in order to determine which tanks will deliver specific demand orders and try to minimize the number of different tanks sending product to the same order. Eq. (15) is the objective function when minimizing switching costs, which includes a penalty for delivering an order from different tanks.

Equation (15) minimizes the number of blend runs, product changeovers in the swing tanks, and deliveries to the same order from different tanks (during the same *L2*-period). Eq. (16) forces all blenders to work at some point during the scheduling horizon (this equation assumes that the number of blend runs required is greater than the number of blenders). Eq. (17) establishes that the delivered amount of a certain product from one tank is equal to the delivered amount from that tank to the specific orders constituted by that product, during a *L2*-period. Eq. (18) constraints the delivery of order *o* to be during a *L2*-period which contains completely or partially the corresponding delivery window (i.e. the parameter $of_{L2}(o,m) = 1$ in such a case). Eq. (19) observes that a tank can only deliver order *o* if it is holding the type of product associated with that order. Eq. (20) constraints binary variable $z_{L2}(j,o,m)$ to be equal to 1 if the tank *j* is delivering order *o* during L2-period *m*. Eq. (21) establishes that the delivered amount must be equal to the demand. Continuous variable $ze_{L2}(o,m)$ represents the number of delivery runs from all tanks to order *o* during *L2*-period *m*, minus one. Eq. (22) constrains the minimum value for $ze_{L2}(o,m)$, which is penalized in Eq. (15). The term –1 in Eq. (22) appears because at least one delivery run is always needed from one tank if the decision is to send product.

$$\min Z_{L2} = \sum_{m=1}^{M} \left(\begin{array}{l} \sum_{(bl,p)\in \mathbf{BP}} c_2(bl) \cdot x_{L2}(p,bl,m) \\ + \sum_{j} c_3(j) \cdot ue_{L2}(j,m) \end{array} \right) + c_4 \cdot \sum_{m=1}^{M} \sum_{o} ze_{L2}(o,m) \quad (15)$$

$$\sum_{m=1}^{M} \sum_{p:(p,bl)\in \mathbf{BP}} x_{L2}(p,bl,m) \geq 1 \qquad \forall\, bl \quad (16)$$

$$Deliver_{pr,L2}(j,p,m) = \sum_{o:(p,o)\in \mathbf{PO}} Deliver_{order,L2}(j,o,m) \qquad \forall\,(j,p) \in \mathbf{JP}, m \geq 1 \quad (17)$$

$$Deliver_{order,L2}(j,o,m) \leq Demand(o) \cdot of_{L2}(o,m) \qquad \forall\,(j,o) \in \mathbf{JO}, m \geq 1 \quad (18)$$

$$Deliver_{order,L2}(j,o,m) \leq Demand(o) \cdot u_{L2}(j,p,m)$$
$$\forall\,(j,p) \in \mathbf{JP}, (j,o) \in \mathbf{JO}, (p,o) \in \mathbf{PO}, m \geq 1 \quad (19)$$

$$Deliver_{order,L2}(j,o,m) \leq Demand(o) \cdot z_{L2}(j,o,m) \qquad \forall\,(j,o) \in \mathbf{JO}, m \geq 1 \quad (20)$$

$$\sum_{m=1}^{M} \sum_{j:(j,o)\in \mathbf{JO}} Deliver_{order,L2}(j,o,m) = Demand(o) \qquad \forall\, o \quad (21)$$

$$ze_{L2}(o,m) \geq \left[\sum_{j:(j,o)\in \mathbf{JO}} z_{L2}(j,o,m) \right] - 1 \qquad \forall\, o, m \geq 1 \quad (22)$$

3.4. Continuous-time scheduling model (L3 level)

The 3rd level scheduling model is based on the full-space model presented in Castillo and Mahalec (2015). After solving the 1st and 2nd levels, there is some information that can be integrated into the 3rd level model. The operating conditions from the 1st level (i.e. the blend recipes), the swing tank allocation, and the approximate production and delivery schedule from the 2nd level can be fixed. This will reduce the model size and lead to shorter execution times. For large problems it may be more desirable to use a time-based decomposition of the scheduling horizon, i.e. the horizon is subdivided in smaller intervals which are solved in sequence. Note that since the blend recipes, the swing tank allocation, and the approximate production and delivery schedule are fixed, the solution from the 3rd level scheduling model will be always equal to or higher than the optimal solution from the original full-space model.

3.4.1. Fixing the blend recipes

Different scheduling problems in the literature consider the use of preferred blend recipes and, from an operational point of view, keeping the blend recipes as constant as possible may be desired. At the 3rd level, we fixed the blend recipes to be equal to those computed at the 1st level. Eqs. (23) and (24) are added in order to fix the blend recipes according to the 1st level solution, i.e. $r_{L1}(i,p,k)$. Equations corresponding to product composition and quality constraints are dropped from the continuous-time scheduling model at the 3rd level since the blend recipes are fixed. Therefore, the 3rd level scheduling model is always a MILP, even if the full-space is a MINLP model (i.e. it uses nonlinear blending equations). The set **KN** = {(*k*, *n*)} indicates which time slots are within each *L1*-period (it can be constructed from the intersection of sets **KM** and **MN**). A small tolerance factor δ_1 is included in order to account for numerical differences between the levels, e.g. in our case studies $\delta_1 = 1\times10^{-6}$.

$$V_{comp,L3}(i,bl,n) \geq \left(r_{L1}(i,p,k) - \delta_1\right) \cdot V_{blend,L3}(bl,n) - F_{blend}^{\max}(bl) \cdot H \cdot \left(1 - x_{L3}(p,bl,n)\right)$$
$$\forall (p,bl) \in \mathbf{BP}, n \in \mathbf{N1}, k : (k,n) \in \mathbf{KN} \qquad (23)$$

$$V_{comp,L3}(i,bl,n) \leq \left(r_{L1}(i,p,k) + \delta_1\right) \cdot V_{blend,L3}(bl,n) + F_{blend}^{\max}(bl) \cdot H \cdot \left(1 - x_{L3}(p,bl,n)\right)$$
$$\forall (p,bl) \in \mathbf{BP}, n \in \mathbf{N1}, k : (k,n) \in \mathbf{KN} \qquad (24)$$

Equation (25) constrains the last blend slots of an *L1*-period to be equal to the time when such *L1*-period ends (i.e. when the blend recipe changes). Eq. (6) is required to compute a feasible solution more easily when using the time-based decomposition described in section 3.4.5.

$$T_{b,L3}(bl,n) = T_{L1}^{end}(k) \qquad \forall\, bl,k,n : (k,n) \in \mathbf{KNend} \qquad (25)$$

3.4.2. Fixing the swing tank allocation

By exploiting the fact that the tank allocation is known from the 2nd level solution, before solving the 3rd level model, binary variable $u_{L3}(j,p,n)$ becomes a parameter by fixing its values using the following relationship: $u_{L3}(j,p,n) = u_{L2}(j,p,m)$, $\forall$ $(j, p) \in$ **JP**, $n \in$ **N1**, m:$(m, n) \in$ **MN**.

3.4.3. Reducing the number of possible production sequences

It may be possible to infer that, during a subinterval of the horizon, only some specific products are (or are not) required to be produced. The solution from the 2nd level is used to implement this idea. Equation (26) is added in order to constrain the scheduling model to the 2nd level solution. This reduces the number of possible production sequences.

$$x_{L3}(p,bl,n) \le x_{L2}(p,bl,m) \qquad \forall (p,bl) \in \mathbf{BP}, n \in \mathbf{N1}, m : (m,n) \in \mathbf{MN} \tag{26}$$

3.4.4. Reducing the number of possible delivery sequences

The approximate delivery schedule can be used to construct set **JON**. Variable $z_{L2}(j,o,m)$ from the 2nd level is equal to 1 if product tank *j* delivers material to order *o* during *L2*-period *m*, and 0 otherwise. Therefore, set **JON** = {(*j*,*o*,*n*) | $z_{L2}(j,o,m)$ =1 for (*j*,*o*) ∈ **JO**, (*m*,*n*) ∈ **MN**} in the MPIP-C algorithm.

3.4.5. Time-based decomposition

Although the reduction of binary variables, use of fixed recipes, and constraints from the approximate scheduling decisions allow to solve larger problems (meaning e.g. larger horizons, more different products, swing tanks, blenders), at some point the model size will become large enough that the execution times will become prohibitive once again. In such a case, we use a time-based decomposition of the scheduling horizon, i.e. the horizon is partitioned in subintervals denoted as *L*-intervals, and these subintervals are solved sequentially starting from the first one. This approach does not guarantee global optimal solutions but will lead to good quality solutions very rapidly.

An integer number of *L2*-periods constitute one *L*-interval and set **LM** = {(*l*, *m*)} indicates such relation. The time slots associated with those *L2*-periods become associated with the corresponding *L*-interval. Analogously to the full-space continuous-time model, the following sets are defined for each *L*-interval model:

N0$_l$ = {*n* | all the time slots belonging to *L*-interval *l*}

N1$_l$ = {*n* | all the time slots belonging to *L*-interval *l*, minus the first slot of the *L*-interval}

N2$_l$ = {*n* | all the time slots belonging to *L*-interval *l*, minus the last slot of the *L*-interval}

N3$_l$ = {*n* | all the time slots belonging to *L*-interval *l*, minus the first and last slots of the *L*-interval}

O$_l$ = {*o* | order *o* can be delivered within *L*-interval *l* according to the 2nd level solution}

Sets **N0**$_l$, **N1**$_l$, **N2**$_l$, **N3**$_l$, and **O**$_l$ replace the previous sets **N0**, **N1**, **N2**, **N3**, and **O**, respectively, in the continuous-time scheduling model from Castillo and Mahalec (2015) in order to solve the model only for *L*-interval *l*. Equation (27) fix the end time for the last unit slots in an *L*-interval. Fig. 5 shows an example of the time horizon at the 3rd level decomposed into *L*-intervals. Note that, in order to link the *L*-intervals, the last slot of an *L*-interval is the first slot of the next *L*-interval.

$$T_{bc,L3}(i,n) = T_{pr,L3}(j,n) = T_{b,L3}(bl,n) = T_{L3}^{end}(l) \qquad \forall i, j, bl, l, n : (l,n) \in \mathbf{LNend} \tag{27}$$

3.4.6. Product inventory targets

When solving the scheduling problem with more than one *L*-interval and using only the sequential approach, extra constraints are required in order to reduce backtracking due to infeasible solutions. Since both the 1st and the 2nd level are solved for the entire horizon, their solutions provide global information. We have chosen the product inventory levels at the end of each *L*-interval and at the end of each recipe regimen to match those from the corresponding 2nd level solution, since the component tanks will be then associated by the fixed blend recipe. Therefore the set of time slots that require these constraints is **Ntarget** = {*n* | *n* is the last slot of an *L*-interval or the last slot before the blend recipe changes}.

Eqs. (28) and (29) are added in order to fix the product inventory levels from the 2nd level solution. A small factor δ_2 is included in order to account for numerical differences between the levels, e.g. in our case studies $\delta_2 = 1\times10^{-6}$.

$$V_{pr,L3}(j,n) \geq V_{pr,L3}^{target}(j,n) - \delta_2 \qquad \forall j,n \in \mathbf{N1}_l, n \in \mathbf{Ntarget} \qquad (28)$$

$$V_{pr,L3}(j,n) \leq V_{pr,L3}^{target}(j,n) + \delta_2 \qquad \forall j,n \in \mathbf{N1}_l, n \in \mathbf{Ntarget} \qquad (29)$$

Where $V_{pr,L3}^{target}(j,n)$ is equal to the inventory level of tank *j* of the *L2*-period associated with the end of an *L*-interval, or a recipe change, computed at the 2nd level.

As an example, the following sets are found in Fig. 5:
LNend = {(1, 2), (2, *n*)}
Ntarget = {2, 4, *n*}
KM = {(1, 1), (1, 2), (*k*, *m*), (*k*, *m*+1)}
MN ={(1, 1), (1, 2), (2, 3), (2, 4), (*m*, 5), (*m*, 6), (*m*, 7), (*m*+1, …), (*m*+1, *n*)}
KN = pairs of elements (*k*, *n*) from the intersection of sets **KM** and **MN**.
LM = {(1, 1), (2, 2), (2, *m*), (2, *m*+1)}
$\mathbf{N0}_{l=1}$ ={0, 1, 2}, $\mathbf{N1}_{l=1}$ ={1, 2}, $\mathbf{N2}_{l=1}$ ={0, 1}, and $\mathbf{N3}_{l=1}$ ={1}.
$\mathbf{N0}_{l=2}$ ={2, 3, 4, 5, 6, 7, …, *n*}, $\mathbf{N1}_{l=2}$ ={3, 4, 5, 6, 7, …, *n*}, $\mathbf{N2}_{l=2}$ ={2, 3, 4, 5, 6, 7, …, *n* –1}, and $\mathbf{N3}_{l=2}$ ={3, 4, 5, 6, 7, …, *n* –1}.

Regarding the set **Ntarget**, slot 2 appears in this set because it is the last slot of an *L*-interval, while slot 4 appears because recipe changes afterwards, and slot *n* for both previous reasons.

3.4.7. Versions of the 3rd level scheduling model

The 3rd level continuous-time scheduling model is denoted as F-NoSimRD or F-SimRD, depending on the operational scenario regarding simultaneous receipt and delivery by product tanks. "SimRD" means that simultaneous receipt/delivery by product tanks is permitted, and "NoSimRD" indicates that simultaneous receipt/delivery by product tanks is not allowed. The "F" stands for fixed recipes. Model F-NoSimRD is constituted by model L-NoSimRD (Castillo & Mahalec, 2015), minus the composition and quality constraints, plus Eqs. (1)–(3), and (23)–(29). Similarly, model F-SimRD is constituted by model L-SimRD (Castillo & Mahalec, 2015), minus the composition and quality constraints, plus Eqs. (1)–(3), and (23)–(29).

The 3rd level model is solved in three phases: a feasibility check, an optimization phase, and the schedule adjustment step.

For the feasibility check, the scheduling model is denoted as F-SimRD-feas (or F-NoSimRD-feas). The cost coefficients c_1, c_2, c_3, and c_4 are equal to zero, i.e. the objective is to find only a feasible solution.

For the optimization phase, all the cost coefficients are used, i.e. the objective is to minimize materials, switching, demurrage, and slack costs. The model is denoted as F-SimRD-opt (or F-NoSimRD-opt).

Model F-SimRD-adj (or F-NoSimRD-adj) is employed for the scheduling adjustment procedure to reduce blending rates variations. In this model, production and delivery sequences are fixed and the objective function minimizes the differences between the average and actual blending rates.

3.5. MPIP-C scheduling algorithm steps

The main steps of the MPIP-C algorithm can be divided into two loops. The inner loop computes the blend recipes and approximate schedule by iterating between the 1st and 2nd level models until the sum of slack variables are equal to or less than the tolerance ε. This inner loop is denoted as the MPIP approximate scheduling algorithm and it is shown in Fig. 6 (Castillo & Mahalec, 2014a). Summarizing subsection 3.1, the initial boundaries of the *L1*-periods are based on:

- Inventory pinch points.
- Changes in the quality of blend components.
- Changes in the cost/price of components/products.

And the boundaries of the *L2*-periods are selected based on:

- Time period boundaries from the 1st level.
- Changes in the supply profile of blend components.
- Order delivery windows.
- Minimum time expected for a swing tank to be in one specific service (i.e. holding one specific product).

- Minimum time expected to complete the smallest blend run allowed.

The outer loop, shown in Fig. 7, deals with the solution of the 3rd level model, determines the number of time slots to obtain a feasible solution or if the blend recipes need to be updated by adding one more *L1*-period and resolve the 1st level model, thus returning to the inner loop.

If desired, solution from the MPIP-C algorithm can be used as the starting point for the full-space model, which can then be solved via MILP of global MINLP solver. At the beginning of their calculations, these solvers compute rapidly the optimality gap corresponding to the MPIP-C solution; if the optimality gap is sufficiently small, the calculation can stop at that point.

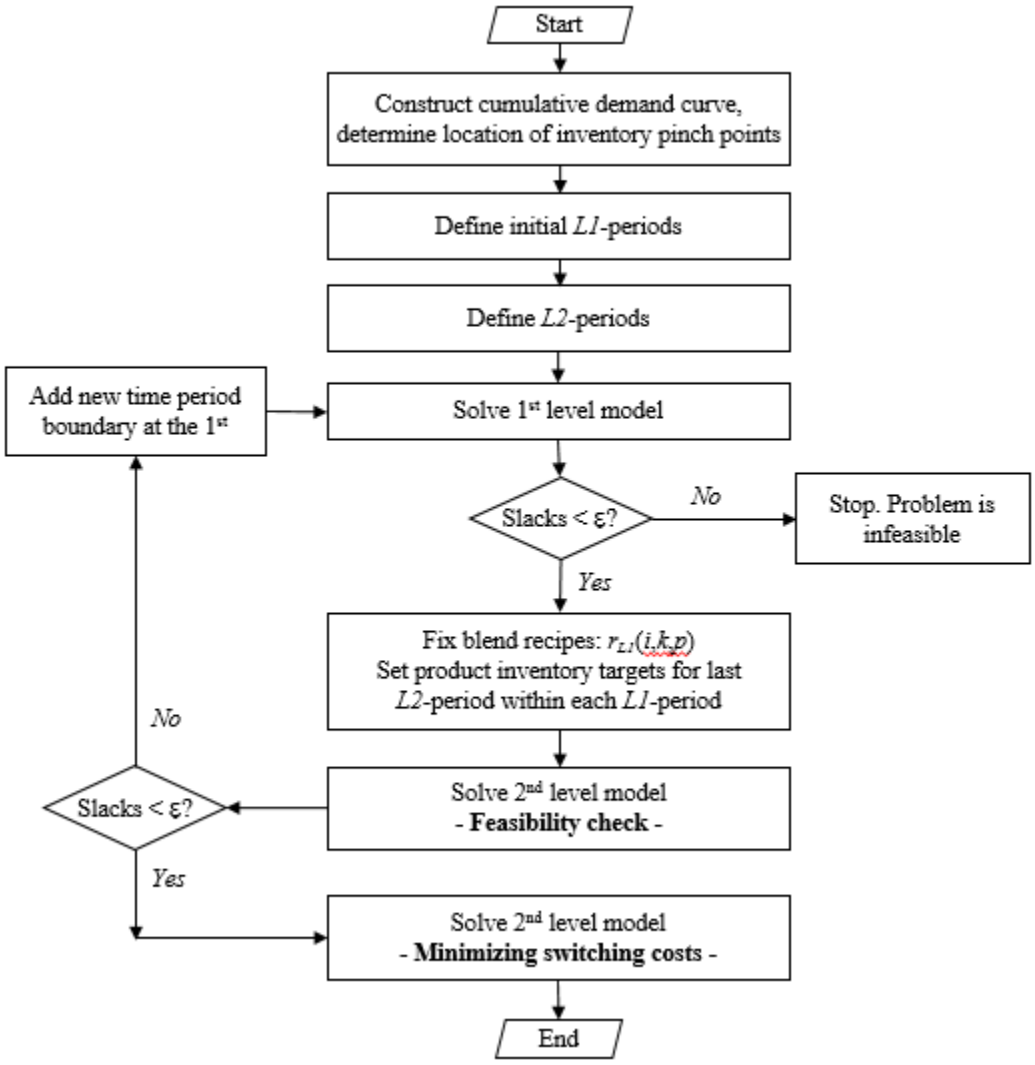

Fig. 6. MPIP approximate scheduling flowchart

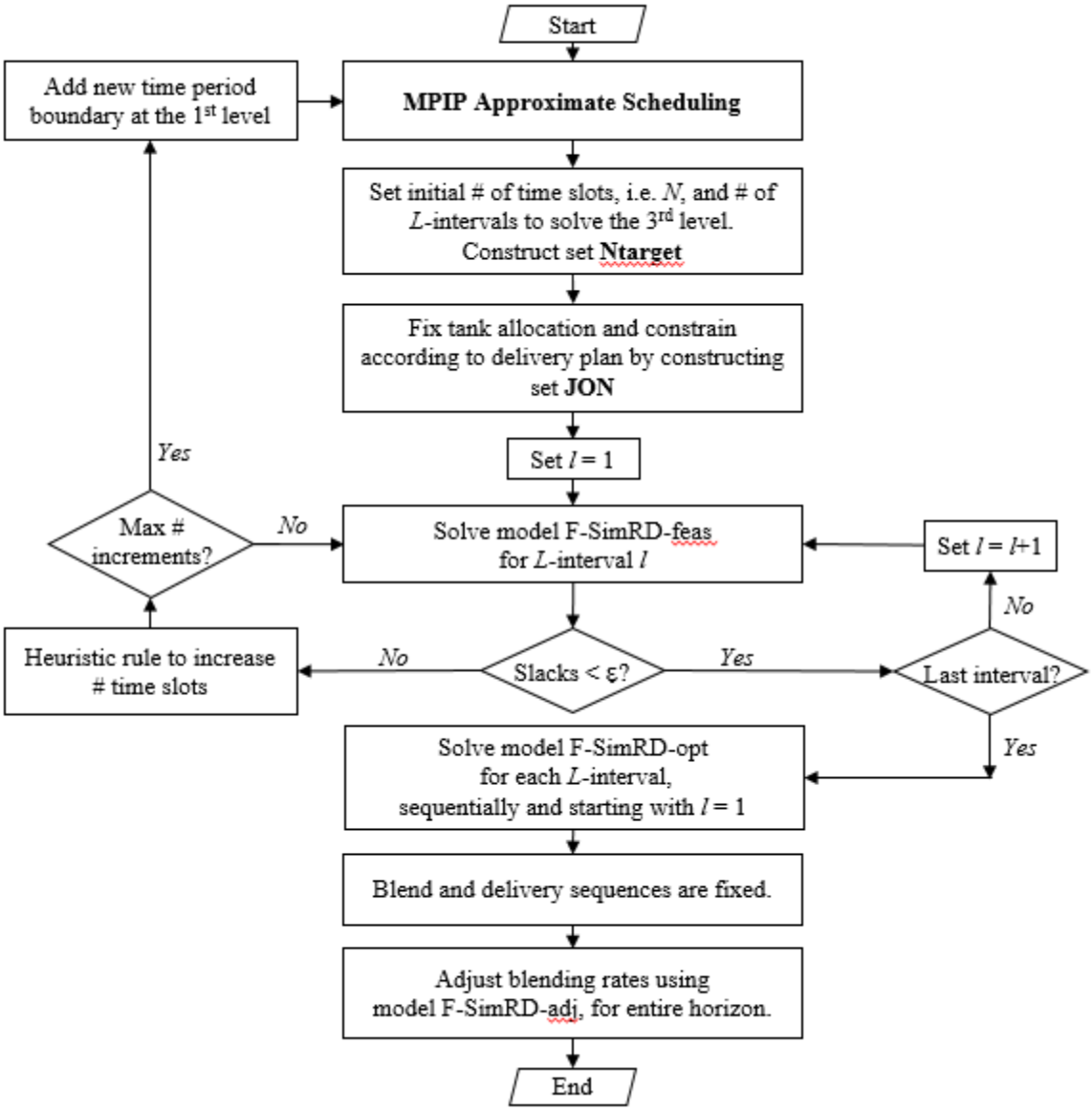

Fig. 7. MPIP-C scheduling algorithm flowchart

4. Test Problems and Computing Machine Used in this Study

All the required models were implemented in GAMS IDE 24.3.2. CPLEX 12.6 was used for MILP models, and BARON 14.0.3 and ANTIGONE 1.1 were employed for MINLP models. CPLEX 12.6 and ANTIGONE 1.1 were selected for LP and NLP models at the 1st level, respectively. All problems have been solved on DELL PowerEdge T310 (Intel® Xeon® CPU, 2.40 GHz, and 12 GB RAM) running Windows Server 2008 R2 OS.

Two sets of test problems have been used in this study:

Test set #1 is composed by a subset of problems from Li and Karimi (2011): Example number 3, 4, 7, 8, 9, 12, and 14. Scheduling horizon is 8 days, linear blending constraints are employed, and component tanks have different supply flowrates along the horizon. Quality properties under specification are RON, Reid vapor pressure, sulfur content, specific gravity, aromatics content, olefin content, benzene content, oxygenates, and flammability limit. According to Li and Karimi (2011), they use the same addition bases and index correlations as Li et al. (2010); therefore, only sulfur content and oxygenates blend on a weight basis. However, Li and Karimi (2011) do not consider specific gravity in example 3, hence in this work sulfur content is assumed to blend linearly on a volumetric basis in example 3. Table 1 shows a description of the blending system for each example.

Test set #2 is composed by example number 4, 8, 12, and 14 from Li and Karimi (2011); however, RON and MON properties are considered to blend nonlinearly following the ethyl RT-70 models. RON index correlation from Li et al. (2010) was used to compute the actual RON values and product specifications. Li and Karimi (2011) do not specify MON values, and these were assumed in this work. For simplicity, MON minimum product specifications were set equal to zero in order to observe only the effect of the RON constraint in the optimum. RON and MON values and specifications are shown in Table 2.

Fig. 8 shows the cumulative curves for example 12 and example 14. Examples 3, 4, 7, 8, 9, and 12 have a single inventory pinch point at time T = 190h, while example 14 does not have an inventory pinch point. Therefore, examples 3 to 12 require two *L1*-periods (boundary between them at 190h mark), and example 14 only one *L1*-period (entire scheduling horizon).

Table 3 shows the *L2*-periods, their corresponding time slots, and the orders that can be delivered during each *L2*-period. Note that for examples 3 to 12, the start of the last *L2*-period matches the inventory pinch point location.

Table 1. Summary of the examples from Test Set #1 and #2 (Li & Karimi, 2011)

Example ID	# Blenders	# Orders	# Products	# Product tanks	# Quality properties (weight basis)
3	1	12	4	11	5
4	1	15	4	11	9 (2)
7	1	20	4	11	9 (2)
8	2	20	4	11	9 (2)
9	2	23	5	11	9 (2)
12	2	35	5	11	9 (2)
14	3	45	5	11	9 (2)

Table 2. RON and MON values for examples from Test Set #2.

Property	**Blend Components**								
	C1	C2	C3	C4	C5	C6	C7	C8	C9
RON	75	90.3	95.6	97.3	83	100	115	118	81
MON	66	80.8	80.5	91.7	74	100	109	100	72
	Product specifications [min, max]								
	P1	P2	P3	P4	P5				
RON	[95, 200]	[96, 200]	[94, 200]	[90, 200]	[98, 200]				
MON	[0, 200]	[0, 200]	[0,200]	[0, 200]	[0, 200]				

Table 3. *L2*-Periods, time slots, and orders that can be delivered in each period (Test Set #1 and #2)

Ex	***L2*-Period**	**Duration (h)**	**Slots**	**Orders that can be delivered**
3, 4	1	100	1, 2	O1 - O7, O12 - O15
	2	90	3 – 5	O8 - O11
	3	2	6	-
7	1	80	1 – 5	O1 - O7, O12 - O19
	2	70	6, 7	O8
	3	40	8	O8 - O11, O20
	4	2	9	-
8	1	80	1 – 3	O1 - O7, O12 - O19
	2	70	4, 5	O8
	3	40	6, 7	O8 - O11, O20
	4	2	8	-
9	1	80	1, 2	O1 - O7, O12 - O19
	2	70	3, 4	O8, O21
	3	40	5, 6	O8 - O11, O20, O22, O23
	4	2	7	-
12	1	24	1, 2	O1 - O4, O13, O15, O19, O33
	2	26	3, 4	O5 - O7, O12, O13, O15, O19, O33
	3	26	5, 6	O14 - O18, O33
	4	24	7, 8	O27, O28
	5	20	9, 10	O21, O24, O29 - O32
	6	30	11, 12	O8, O34, O35
	7	40	13	O8 - O11, O20, O22, O23, O25, O26
	8	2	14	-
14	1	24	1, 2	O1 - O4, O13, O15, O19, O26
	2	26	3, 4	O5 - O7, O12, O13, O15, O19, O26
	3	26	5, 6	O14 - O18, O26
	4	24	7, 8	-
	5	20	9, 10	O21, O24, O45
	6	30	11, 12	O8, O27 - O31
	7	42	13	O8 - O11, O20, O22, O23, O25, O32 - O44

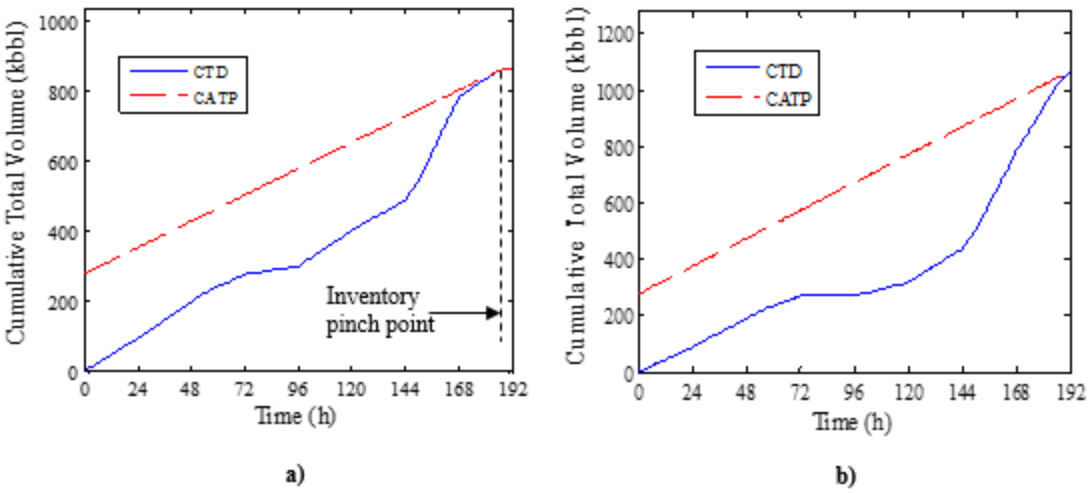

Fig. 8. Cumulative curves and inventory pinch points for a) example 12 and b) example 14.

5. Computational Performance of MPIP-C Algorithm

Castillo and Mahalec (2015) showed that their continuous-time full-space blend scheduling model enables exact solutions of medium-size models when there are no penalties for multiple deliveries from the tanks. Global optimal solutions become much more difficult to compute when minimum number of multiple deliveries from the tanks is desired, and when using nonlinear blending equations. MPIP-C algorithm represents an alternative to compute good quality solutions for those cases and with short execution times. This section summarizes the results of MPIP-C algorithm.

5.1. Test set #1

Simultaneous receipt and delivery by product tanks is considered in this case. Table 4 shows the model size of the full-space model, and the models at each level of the MPIP-C algorithm, for examples from Test Set #1 when employing the data from Table 2. Since the examples have very similar cumulative curves, 1st level model size does not change significantly. 1st level model size increases with the number of products required to be blended and with the number of *L1*-periods. It can be seen that the 3rd level model has approximately half of the total number of binary variables required by the full-space model.

Table 5 presents the results for test set #1. For small problems (examples 3 to 9) MPIP-C computes optimal results, while for examples 12 and 14, the results have 0.13% and 0.09% optimality gaps, respectively. The optimality gap was computed by providing MPIP-C solution as a starting point to the full-space model; the first reported lower bound by CPLEX corresponds to the MPIP-C solution. Execution times for small problems are of the same order of magnitude

as those for the new version of the full-space MILP model (Castillo & Mahalec, 2015). For large problems, the execution times for MPIP-C algorithm are significantly smaller.

Shown in Table 5 are also the solutions which are obtained by providing MPIP-C result as a starting point to the full-space continuous-time model (row "FS-WS"). Using this procedure, example 12 was solved to optimality in 11.9s + 418s = 430 seconds, while example 14 was solved to 0% optimality gap in 19.7s + 1,611s = 1,631 seconds. This is significantly faster than the full-space model without starting point.

Figs. 9 and 10 show the production and delivery schedule, respectively, computed by MPIP-C algorithm for example 12, while Figs. 11 and 12 show the corresponding schedules obtained by the FS-WS approach for the same example.

Table 4. Model size of the full-space model, and the models at each level of the MPIP-C algorithm, for Test Set #1 (linear examples).

Example ID	**3**	**4**	**7**	**8**	**9**	**12**	**14**
Full-space linear model L-SimRD-opt							
# Slots	6	6	9	8	7	14	13
# Eqs	6,730	7,756	15,252	15,725	13,958	27,007	33,309
# Cont	1,946	2,189	3,978	3,653	3,263	7,166	7,889
# Bins	397	433	804	728	642	1,194	1,276
MPIP-C 1st level model (recipe optimization)							
# *L1*-periods	2	2	2	2	2	2	1
# Eqs	423	487	487	487	588	588	316
# Cont	185	185	185	185	224	224	120
MPIP-C 2nd level model (approximate schedule)							
# *L2*-periods	3	3	4	4	4	8	7
# Eqs	2,305	2,560	3,870	4,343	4,880	12,648	13,151
# Cont	1,155	1,248	1,800	2,052	2,366	5,539	5,790
# Bins	393	447	699	827	924	2,511	2,666
MPIP-C 3rd level model F-SimRD-opt (detailed schedule)							
# Slots	6	6	9	8	7	14	13
# Eqs	5,391	5,785	11,043	11,622	10,309	19,788	24,643
# Cont	1,861	2,027	3,550	3,379	3,059	6,794	7,551
# Bins	164	178	372	378	316	529	698

Table 5. Full-space and MPIP-C results, SimRD scenario, not penalizing deliveries ($c_4 = 0$). Test Set #1 (linear examples).

Ex.	Algorithm	Obj. Func. Value (×10^3$)	Gap (%)	Blend Cost	# BR	# TST	# DR	Total CPU time (s)
3	MPIP[a]	3,179.1	0	3,139.1	2	0	19	14.2
	Full-Space[b]	3,179.1	0	3,139.1	2	0	17	2.6
	MPIP-C	3,179.1	0	3,139.1	2	0	18	1.6
	FS-WS	3,179.1	0	3,139.1	2	0	18	2.4
4	MPIP[a]	4,576.7	0	4,536.7	2	0	23	16.2
	Full-Space[b]	4,576.7	0	4,536.7	2	0	20	2.3
	MPIP-C	4,576.7	0	4,536.7	2	0	21	1.7
	FS-WS	4,576.7	0	4,536.7	2	0	21	2.6
7	MPIP[a]	8,120.3	0	8,040.3	4	0	29	17.0
	Full-Space[b]	8,120.3	0	8,040.3	4	0	31	63.1
	MPIP-C	8,120.3	0	8,040.3	4	0	29	4.9
	FS-WS	8,120.3	0	8,040.3	4	0	29	2.8
8	MPIP[a]	8,120.3	0	8,040.3	4	0	28	17.3
	Full-Space[b]	8,120.3	0	8,040.3	4	0	28	37.8
	MPIP-C	8,120.3	0	8,040.3	4	0	26	5.9
	FS-WS	8,120.3	0	8,040.3	4	0	26	4.2
9	MPIP[a]	10,818.8	0	10,704.3	5	1	30	33.2
	Full-Space[b]	10,818.8	0	10,704.3	5	1	32	83.4
	MPIP-C	10,818.8	0	10,704.3	5	1	29	5.1
	FS-WS	10,818.8	0	10,704.3	5	1	29	4.2
12	MPIP[a]	15,281.7	0.13	15,147.2	6	1	42	129
	Full-Space[b]	15,281.7	0.13	15,147.2	6	1	49	43,200[c]
	MPIP-C	15,281.7	0.13	15,147.2	6	1	46	11.9
	FS-WS	15,261.7	0	15,147.2	5	1	45	418
14	MPIP[a]	21,181.4	0.09	21,046.9	6	1	53	333
	Full-Space[b]	21,161.4	0	21,046.9	5	1	62	12,106
	MPIP-C	21,181.4	0.09	21,046.9	6	1	53	19.7
	FS-WS	21,161.4	0	21,046.9	5	1	53	1,611

[a] MPIP scheduling algorithm from Castillo and Mahalec (2014b)

[b] Full-space model from Castillo and Mahalec (2015)

[c] Reached maximum allocated time

FS-WS = Full-space model with warm start (MPIP-C solution as starting point)

#BR = number of blend runs, #TST = number of transitions by swing tanks, #DR = number of delivery runs.

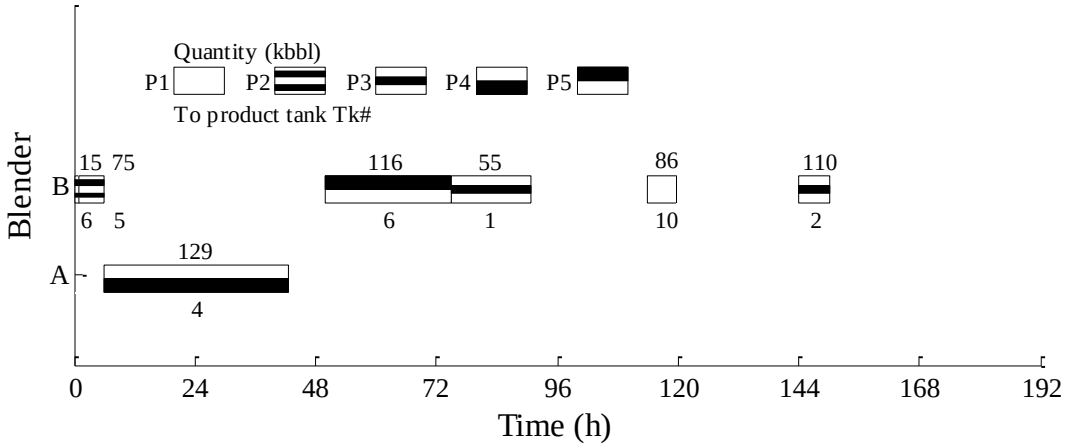

Fig. 9. Production schedule for example 12, Test Set #1, SimRD, MPIP-C solution.

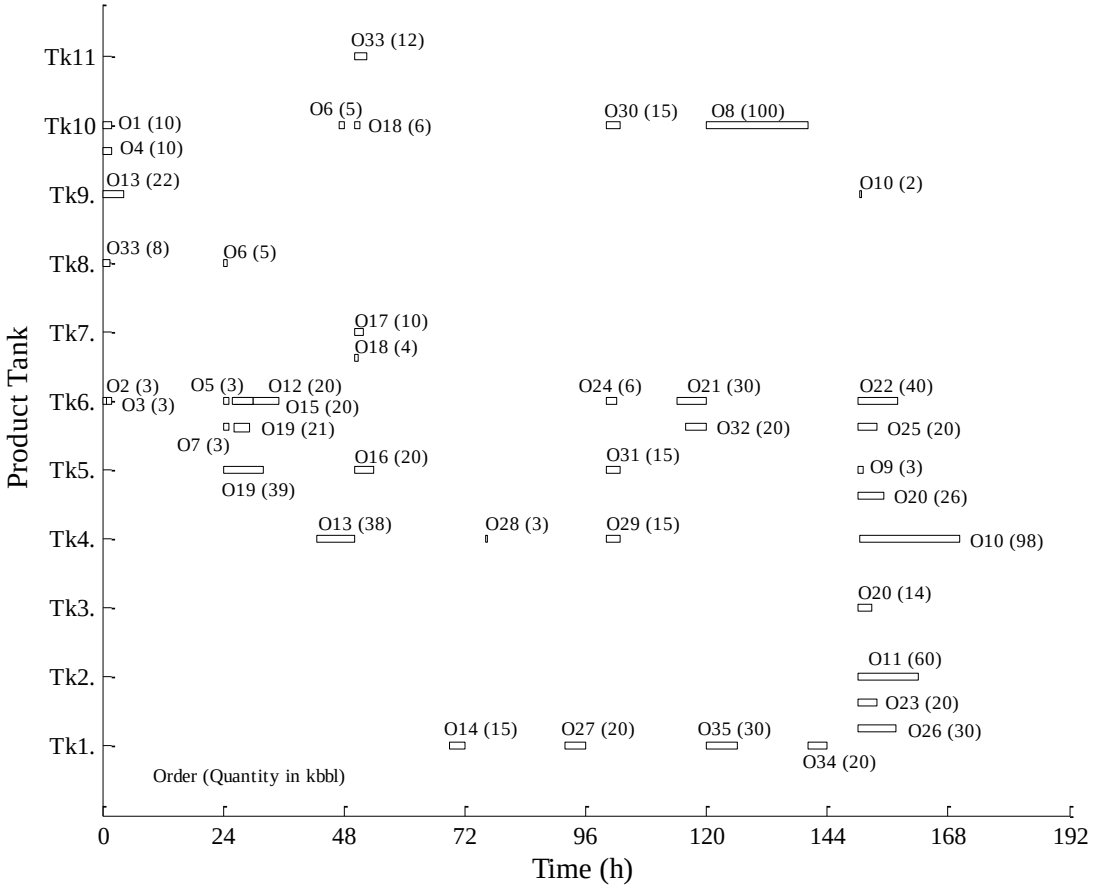

Fig. 10. Delivery schedule for example 12, Test Set #1, SimRD, MPIP-C solution.

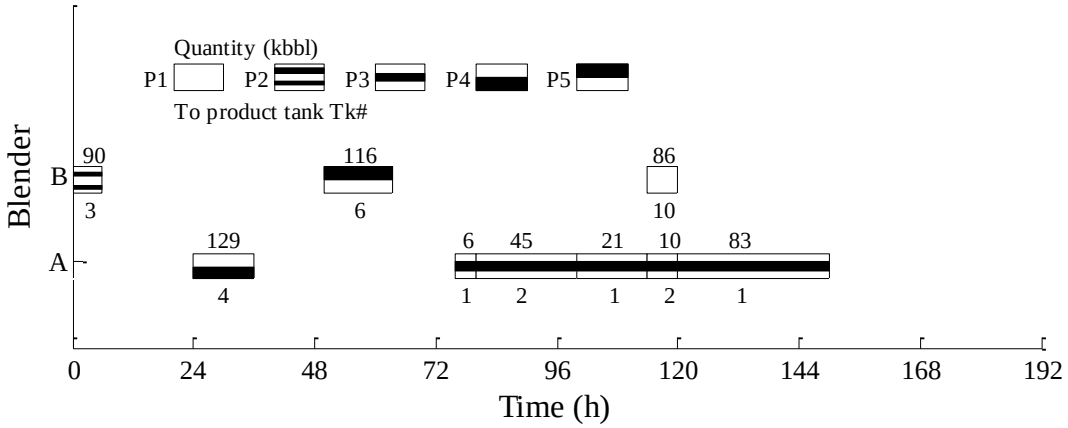

Fig. 11. Production schedule for example 12, Test Set #1, SimRD, FS-WS solution.

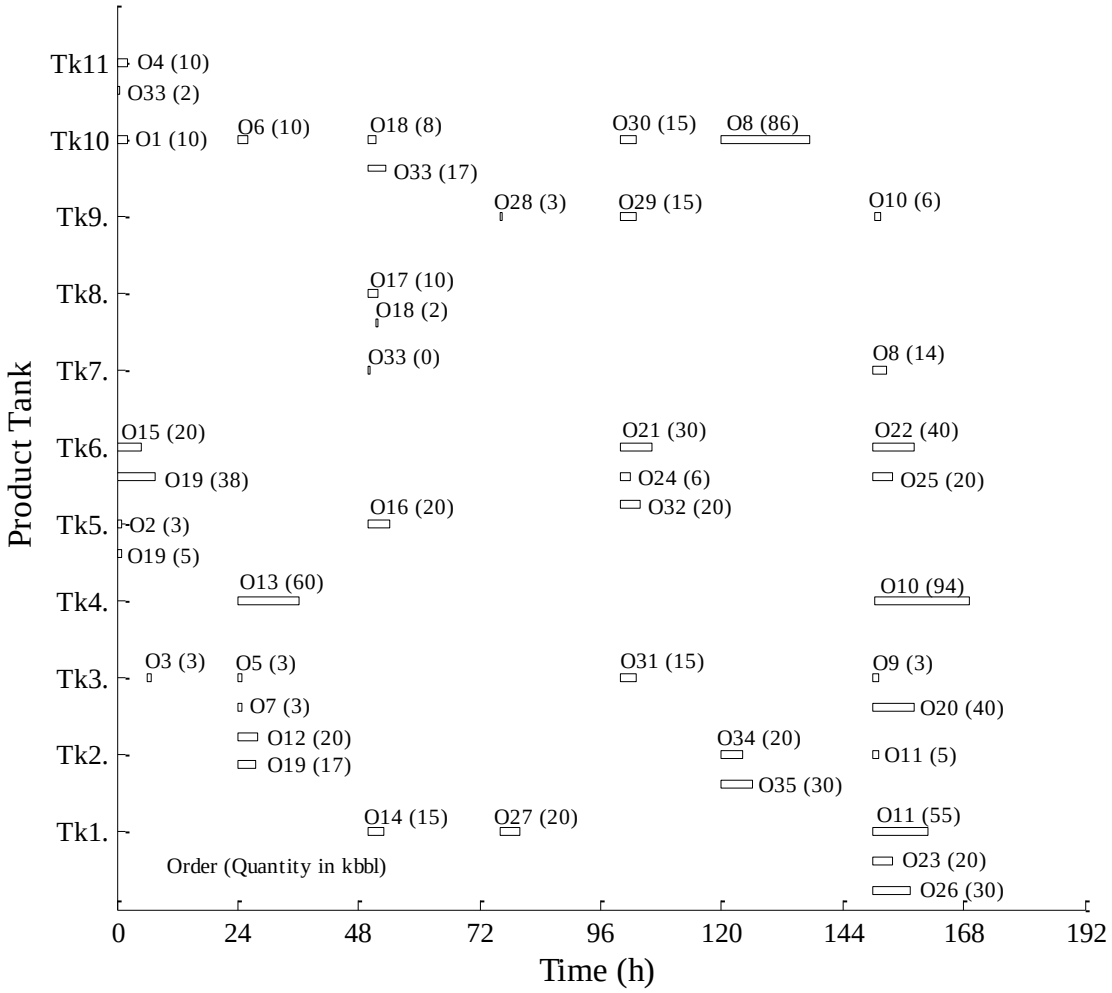

Fig. 12. Delivery schedule for example 12, Test Set #1, SimRD, FS-WS solution.

5.2. Test set #2

For these examples, it is assumed that simultaneous receipt and delivery by product tanks is not allowed. Table 6 shows the model size of the full-space model, and the models at each level of the MPIP-C algorithm, for examples from Test Set #2 when employing the data from Table 2. It can be seen that the number of binary variables of the full-space model is the same for linear and nonlinear models (see Table 4) since the difference is only the addition of equations and continuous variables for the octane blending model. Compared with the linear cases, the nonlinear 1st level model has more equations and continuous variables; however, the 2nd level model has the same size in both cases (this is expected since the 2nd level model only depends on the number of *L2*-periods). The size of the 3rd level model depends on the solution from the 2nd level; therefore, it is not significantly affected by using a linear or nonlinear blending model. Once again, the 3rd level model needs around 50% of the total number of binary variables required by the full-space model.

The stopping criteria for Test Set #2 examples is 0.01% optimality gap or the maximum allocated time. Table 7 shows the results from the full-space MINLP model, MPIP-C algorithm, and the full-space MINLP model with MPIP-C solution as starting point. ANTIGONE is used to solve the 1st level model of the MPIP-C algorithm. Although the full-space model computes solutions with small optimality gaps, it may require more than 900 seconds to find solutions with gaps smaller than 5%. For the large-scale problem (example 14), MPIP-C algorithm computes a solution within 1% of the optimum in 24 seconds, while ANTIGONE and BARON require 1,205s and 3,634s, respectively, to compute a solution within 5% of the optimum. When using the MPIP-C solution as starting point for the full-space model, only examples 4 and 8 (small problems) can be solved to proven optimality within the allocated time.

Table 8 shows the execution times at each level of the MPIP-C algorithm for examples 4, 8, 12, and 14, for both linear and nonlinear cases. It can be seen that the 2nd and 3rd level execution times are very similar in magnitude (2nd and 3rd levels are always linear), while the 1st level execution times increased at least 6 times when the nonlinear blending equations are employed. This suggests that using more complex nonlinear blending models will only affect the 1st level execution times, since the 2nd and 3rd level use fixed recipes.

Table 6. Model size of the full-space model, and the models at each level of the MPIP-C algorithm, for Test Set #2 (nonlinear examples).

Example ID	**4**	**8**	**12**	**14**
Full-space nonlinear model N-NoSimRD-opt				
# Slots	6	8	14	13
# Eqs	6,169	12,297	22,699	27,729
# Cont	2,028	3,536	7,095	8,088
# Bins	433	728	1,194	1,276
# Nonlinear terms	156	416	728	1,014
MPIP-C 1st level model (recipe optimization)				
# *L1*-periods	2	2	2	1
# Eqs	591	591	718	381
# Cont	273	273	334	175
# Nonlinear terms	64	64	80	40
MPIP-C 2nd level model (approximate schedule)				
# *L2*-periods	3	4	8	7
# Eqs	2,560	4,343	12,648	13,151
# Cont	1,248	2,052	5,539	5,790
# Bins	447	827	2,511	2,666
MPIP-C 3rd level model F-NoSimRD-opt (detailed schedule)				
# Slots	6	8	14	13
# Eqs	4,876	9,462	17,436	21,064
# Cont	1,881	3,137	6,516	7,238
# Bins	178	378	532	695

#Eqs = number of equations, #Cont = continuous variables, #Bins = number of binary variables.

Table 7. Full-space and MPIP-C results, SimRD scenario, not penalizing deliveries ($c_4 = 0$). Test Set #2 (nonlinear examples).

Ex	Algorithm	MINLP solver	Obj. Func. Value ($\times 10^3$\$)	Gap (%)	CPU time (s) to reach:			Blend Cost ($\times 10^3$\$)	# BR	# TST	# DR
					Stop criteria	Final upper bound	Opt. gap < 5%				
4	Full-Space[a]	ANTIGONE	4,633.0	0.01	856	856	141	4,593.0	2	0	21
	Full-Space[a]	BARON	4,633.0	0.01	15	15	11	4,593.0	2	0	21
	MPIP-C	-	4,632.7	-	3	3	-	4,592.7	2	0	22
	FS-WS	ANTIGONE	4,632.7	0.01	121	0	0	4,592.7	2	0	22
	FS-WS	BARON	4,632.7	0.01	7	0	0	4,592.7	2	0	22
8	Full-Space[a]	ANTIGONE	8,205.4	0.03	10,800[b]	9,246	380	8,125.4	4	0	28
	Full-Space[a]	BARON	8,207.4	0.05	10,800[b]	1,388	79	8,127.4	4	0	26
	MPIP-C	-	8,203.1	-	6	6	-	8,123.1	4	0	28
	FS-WS	ANTIGONE	8,203.1	0.01	1,209	0	0	8,123.1	4	0	28
	FS-WS	BARON	8,203.1	0.01	16	0	0	8,123.1	4	0	28
12	Full-Space[a]	ANTIGONE	15,406.8	0.16	43,200[b]	43,054	907	15,272.3	6	1	42
	Full-Space[a]	BARON	15,453.1	0.45	43,200[b]	7,933	7,933	15,318.6	6	1	45
	MPIP-C	-	15,402.6	-	17	17	-	15,268.1	6	1	44
	FS-WS	ANTIGONE	15,402.6	0.13	43,200[b]	0	0	15,268.1	6	1	44
	FS-WS	BARON	15,402.6	0.13	43,200[b]	0	0	15,268.1	6	1	44
14	Full-Space[a]	ANTIGONE	21,283.1	0.19	43,200[b]	12,839	1,205	21,128.6	7	1	56
	Full-Space[a]	BARON	21,497.7	1.20	43,200[b]	7,204	3,634	21,383.2	5	1	56
	MPIP-C	-	21,263.1	-	24	24	-	21,128.6	6	1	55
	FS-WS	ANTIGONE	21,263.1	0.09	43,200[b]	0	0	21,128.6	6	1	55
	FS-WS	BARON	21,263.1	0.09	43,200[b]	0	0	21,128.6	6	1	55

[a] Full-space model from Castillo and Mahalec (2015)

[b] Reached maximum allocated time

FS-WS = Full-space model with warm start (MPIP-C solution as starting point)

#BR = number of blend runs, #TST = number of transitions by swing tanks, #DR = number of delivery runs.

Table 8. Comparison of the execution times at each level of the MPIP-C algorithm.

CPU time (seconds)			
MPIP-C level:	**1st level**	**2nd level**	**3rd level**
Test Set #1 (linear)			
Solver	CPLEX	CPLEX	CPLEX
Ex. 4	0.123	0.384	1.193
Ex. 8	0.185	0.592	5.169
Ex. 12	0.200	3.612	8.088
Ex. 14	0.145	4.912	14.658
Test Set #2 (nonlinear)			
Solver	ANTIGONE	CPLEX	CPLEX
Ex. 4	0.766	0.268	1.222
Ex. 8	0.890	0.387	4.475
Ex. 12	2.652	3.460	10.125
Ex. 14	1.605	5.423	16.513

6. Conclusions

In this paper we have presented Multi-Period Inventory Pinch scheduling algorithm with Continuous-time model used for detailed scheduling (MPIP-C scheduling algorithm). The algorithm uses inventory-pinch concept to decouple optimization of the blend recipes from the combinatorial aspects of scheduling. It also combines the strengths of discrete-time models (what to produce, in which time period, and how much to produce, where to store, where from to deliver) with continuous-time model capabilities to compute accurately when to start or to stop an operation.

Multi-period inventory pinch, an essential feature of MPIP-C algorithm, enables computation of approximate schedule by iterations between the top two levels. The 1st level optimizes a linear or nonlinear blending model (LP or NLP, respectively) and the 2nd level optimizes approximate scheduling model (MILP) with fixed recipes from the 1st level.

For small size scheduling problems and linear blending rules, MPIP-C algorithm requires execution times of the same magnitude as the reduced-size full-space continuous-time model (Castillo & Mahalec, 2015). However, as the problem size grows, decomposition employed by MPIP-C enables computation of optimal or near optimal solutions within very short amounts of time. In addition, the execution times for the full-space model grow very rapidly. Rapid convergence of MPIP-C algorithm makes possible to use its solution as a starting point for a full-space model. For medium size problems such approach leads to an optimal solution within 1 hour or less. However, for large problems the global optimum still cannot be computed within 12 hours. Execution times for MPIP-C are determined by the speed of solving the nonlinear

problem at the top level. If more elaborate nonlinear models are used at the 1st level, they will only increase the computational times at the 1st level but not at the 2nd and 3rd levels.

Due to the structure of MPIP-C algorithm, we expect that even larger problems can be solved efficiently. This is possible as long as the approximate schedule (2^{nd} level) is computed very rapidly, since the approximate scheduling solution can be used to partition the time horizon at the 3^{rd} level into smaller intervals. If these intervals are small, their continuous-time models can be solved very rapidly. Even though MPIP-C algorithm is heuristic in nature, most of the solutions obtained are at least as good as the results obtained by full-space models over much longer execution times.

Acknowledgments

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

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.compchemeng.2015.08.005.

Nomenclature

Sets and indices

BL = {*bl*}	Blenders
E = {*e*}	Quality properties (e.g. research and motor octane number)
I = {*i*}	Blend components
J = {*j*}	Product tanks
K = {*k*}	*L1*-periods (time periods defined for the 1st level discrete-time model)
L = {*l*}	*L*-intervals (non-overlapping subintervals of the scheduling horizon for the 3rd level continuous-time model)
M = {*m*}	*L2*-periods (time periods defined for the 2nd level discrete-time model)
N0 = {*n* | 0, 1, …, *N*}	Time slots assigned for the entire horizon (3rd level continuous-time model)
N1 = {*n* | 1, …, *N*}	Subset of **N0**, does not include first time slot
N2 = {*n* | 0, 1, …, *N* – 1}	Subset of **N0**, does not include last time slot
N3 = {*n* | 1, …, *N* – 1}	Subset of **N0**, does not include first and last time slots
$\mathbf{N0}_l$, $\mathbf{N1}_l$, $\mathbf{N2}_l$, $\mathbf{N3}_l$	Analogous sets to **N0**, **N1**, **N2**, and **N3**, respectively, defined for each *L*-interval
O = {*o*}	All demand orders
$\mathbf{O}_l$	Demand orders that can be delivered during *L*-interval *l*
P = {*p*}	Different products
BP = {(*p*, *bl*)}	Blender *bl* can produce product *p*
JO = {(*j*, *o*)}	Product tank *j* can deliver order *o*
KM = {(*k*, *m*)}	*L2*-period *m* is within *L1*-period *k*
KN = {(*k*, *n*)}	Time slot *n* is within *L1*-period *k*
KNend = {(*k*, *n*)}	Time slot *n* is the last one of *L1*-period *k*
LM = {(*l*, *m*)}	*L2*-period *m* is within *L*-interval *l*
LN = {(*l*, *n*)}	Time slot *n* is within *L*-interval *l*
LNend = {(*l*, *n*)}	Last time slot of *L*-interval *l*
MN = {(*m*, *n*)}	Time slot *n* is within *L2*-period *m*
PO = {(*p*, *o*)}	Order *o* consists of product *p*
JON = {(*j*, *o*, *n*)}	Product tank *j* can deliver order *o* during time slot *n*

Parameters

$c_2(bl)$	Cost associated with one blend run in blender *bl*
$c_3(j)$	Cost associated with a product transition in swing tank *j*
c_4	Cost associated with a delivery run
$Demand(o)$	Demand quantity of order *o*
$F_{blend}^{\min}(bl)$, $F_{blend}^{\max}(bl)$	Minimum and maximum blending rates of blender *bl*
H	Length of the entire scheduling horizon
$of_{L2}(o,m)$	If equal to 1, delivery window of order *o* spans, partially or completely, *L2*-period *m*

$Q_{bc,L1}(i,e,k)$	Quality *e* of blend component *i* during *L1*-period *k*
$Q_{pr}^{\min}(p,e),\ Q_{pr}^{\max}(p,e)$	Minimum and maximum specifications for quality property *e* and product *p*
$T_{L1}^{end}(k)$	End time of *L1*-period *k*
$T_{L2}^{end}(m)$	End time of *L2*-period *m*
$T_{L3}^{end}(l)$	End time of *L*-interval *l*
$V_{pr,L3}^{target}(j,n)$	Inventory target for product tank *j* at time slot *n*

Binary variables

$u_{L2}(j,p,m)$	Binary variable at the 2nd level, but a parameter at the 3rd level. If equal to 1, product tank *j* is storing product *p* during *L2*-period *m*
$u_{L3}(j,p,n)$	If equal to 1, product tank *j* is storing product *p* during slot *n*
$x_{L2}(p,bl,m)$	Binary variable at the 2nd level, but a parameter at the 3rd level. If equal to 1, blender *bl* is processing product *p* during slot *n* during *L2*-period *m*
$z_{L2}(j,o,m)$	Binary variable at the 2nd level, but a parameter at the 3rd level. If equal to 1, product tank *j* delivers material to order *o* during *L2*-period *m*
$z_{L3}(j,o,n)$	If equal to 1, product tank *j* delivers material to order *o* during slot *n*

0-1 Continuous variables

$x_{L3}(p,bl,n)$	If equal to 1, blender *bl* is processing product *p* during slot *n*

Continuous variables

$Deliver_{order,L2}(j,o,m)$	Volume delivered from product tank *j* to order *o* during *L2*-period *m*
$Deliver_{pr,L2}(j,p,m)$	Volume delivered of product *p* from product tank *j* during *L2*-period *m*
$DV_{pr}(j,o,n)$	Volume delivered from product tank *j* to order *o* during slot *n*
$Q_{pr,L1}(p,e,k)$	Quality *e* of product *p* during *L1*-period *k*
$r_{L1}(i,p,k)$	Continuous variable at the 1st level, but a parameter at the 2nd and 3rd levels. Blend recipe for product *p* in *L1*-period *k*
$T_{b,L3}(bl,n)$	End time of a blender slot *n*
$T_{bc,L3}(i,n)$	End time of a component tank slot *n*
$T_{pr,L3}(j,n)$	End time of a product tank slot *n*
$ue_{L2}(j,m)$	If equal to 1, product tank *j* changes its service at the end of *L2*-period *m*
$ue_{L3}(j,n)$	If equal to 1, product tank *j* changes its service at the end of slot *n*
$V_{blend,L3}(bl,n)$	Volume blended during slot *n* by blender *bl*
$V_{comp,L1}(i,p,k)$	Volume required of component *i* according to the aggregated model solution
$V_{comp,L3}(i,bl,n)$	Volume of component *i* used in blender *bl* during slot *n*

$V_{pr,L3}(j,n)$	Inventory level of product tank *j* at the end of slot *n*
$ze_{L2}(o,m)$	Number of different product tanks delivering material to order *o* during *L2*-period *m*

References

Bassett MH, Pekny JF, Reklaitis GV. Decomposition techniques for the solution of large-scale scheduling problems. AIChE J 1996; 42(12):3373–3387.

Birewar DB, Grossmann IE. Simultaneous production planning and scheduling in multiproduct batch plants. Ind Eng Chem Res 1990; 29:570-580.

Castillo PA, Kelly JD, Mahalec V. Inventory pinch algorithm for gasoline blend planning. AIChE J 2013; 59(10):3748-3766.

Castillo PA, Mahalec V. Inventory pinch based, multiscale models for integrated planning and scheduling-part I: Gasoline blend planning. AIChE J 2014a; 60(6):2158-2178.

Castillo PA, Mahalec V. Inventory pinch based, multiscale models for integrated planning and scheduling-part II: Gasoline blend scheduling. AIChE J 2014b; 60(7):2475-2497.

Castillo PA, Mahalec V. Reduced size continuous-time model for gasoline blend scheduling. Comput Chem Eng 2015; "submitted for publication".

Chen X, Wang N. Optimization of short-time gasoline blending scheduling problem with a DNA based hybrid genetic algorithm. Chem Eng Process 2010; 49:1076-1083.

Glismann K, Gruhn G. Short-term scheduling and recipe optimization of blending processes. Comput Chem Eng 2001; 25:627–634.

Gothe-Lundgren M, Lundgren JT, Persson JA. An optimization model for refinery production scheduling. Int J Prod Econ 2002; 78:255-270.

Healey WC, Maasen CW, Peterson RT. A new approach to blending octanes. In: Proc. 24th Meeting of American Petroleum Institute's Division of Refining. New York, 1959.

Jia Z, Ierapetritou M. Mixed-integer linear programming model for gasoline blending. Ind Eng Chem Res 2003; 42:825-835.

Jia Z, Ierapetritou M. Efficient short-term scheduling of refinery operations based on a continuous time formulation. Comput Chem Eng 2004; 28:1001–1019.

Kolodziej SP, Grossmann IE, Furman KC, Sawayac NW. A discretization-based approach for the optimization of the multiperiod blend scheduling problem. Comput Chem Eng 2013; 53:122-142.

Li J, Karimi IA. Scheduling gasoline blending operations from recipe determination to shipping using unit slots. Ind Eng Chem Res 2011; 50:9156–9174.

Li J, Karimi IA, Srinivasan R. Recipe determination and scheduling of gasoline blending operations. AIChE J 2010; 56:441-465.

Li J, Misener R, Floudas CA. Continuous-time modeling and global optimization approach for scheduling of crude oil operations. AIChE J 2012; 58:205-226.

Mendez CA, Grossmann IE, Harjunkoski I, Kabore P. A simultaneous optimization approach for off-line blending. Comput Chem Eng 2006; 30:614–634.

Pinto JM, Joly M, Moro LFL. Planning and scheduling models for refinery operations. Comput Chem Eng 2000; 24:2259-2276.

Shah NK, Ierapetritou MG. Short-term scheduling of a large-scale oil-refinery operations: Incorporating logistic details. AIChE J 2011; 57:1570–1584.

Singh A, Forbes JF, Vermeer PJ, Woo SS. Model-based real-time optimization of automotive gasoline blending operations. J Process Control 2000; 10:43-58.

Terrazas-Moreno S, Grossmann IE. A multiscale decomposition method for the optimal planning and scheduling of multi-site continuous multiproduct plants. Chem Eng Sci 2011; 66:4307-4318.

Wolpert DH, Macready WG. No free lunch theorems for optimization. IEEE Trans Evol Comput 1997; 1(1):67-82.

Wu D, Ierapetritou MG. Decomposition approaches for the efficient solution of short-term scheduling problems. Comput Chem Eng 2003; 27:1261-1276.

Zhao J, Wang N. A bio-inspired algorithm based on membrane computing and its application to gasoline blending scheduling. Comput Chem Eng 2011; 35:272-283.