Nonlinear Blend Scheduling via Inventory Pinch-based Algorithm using Discrete- and Continuous-time Models

doi: 10.15255/CABEQ.2014.1941

V. Mahalec* **and P. Castillo Castillo**
Dept. of Chemical Engineering, McMaster University,
Hamilton, ON L8S 4L8, Canada

Original scientific paper
Received: September 23, 2014
Accepted: November 25, 2014

This work uses multi-period, inventory pinch-based algorithm with continuous-time model (MPIP-C algorithm[1]) for scheduling linear or nonlinear blending processes. MPIP-C decomposes the scheduling problem into (i) approximate scheduling and (ii) detailed scheduling. Approximate scheduling model is further decomposed into two parts: a 1st level model which optimizes nonlinear blend models (with time periods delineated by inventory pinch points), and a 2nd level multi-period mixed-integer linear programming model (which uses fixed blend recipes from the 1st level solution) to determine optimal production plan and swing storage allocation, while minimizing the number of blend instances and product changeovers in the swing tanks. The 3rd level computes schedules using a continuous-time model including constraints based on the short-term plan solution. Nonlinear constraints are used for the Reid vapor pressure in our case studies. Excellent computational performance is illustrated by comparisons with previous approach with discrete-time scheduling model.

Key words:
inventory pinch, nonlinear gasoline blending, discrete-time scheduling model, continuous-time scheduling model

Introduction

Mathematical optimization has become an important supply chain management tool for any contemporary enterprise. Increasing profit margins through better design, planning, and operative decisions across the supply chain represents a huge opportunity in the more than ever competitive economic market, especially in a global scale. Industrial and academic researchers have been working intensively in the process systems engineering area, and several advancements have been made in the optimization field, from new mathematical models, more efficient solution algorithms, use of more powerful computing machines and with parallel capabilities, and the development of different optimization frameworks.

A supply chain is composed of the following elements: procurement and storage of raw materials, facilities and processes to transform the raw materials into intermediate and final products, storage of these products and its distribution to warehouses or to the final customers. Operating the complete supply chain structure in the best possible manner (i.e. maximizing profit or minimizing cost) involves making decisions at different levels along the supply chain network. Production planning determines the production targets of each different product along a defined planning horizon (usually ranging from a few months to 2 years), for the entire supply chain, for each different production facility, or for each production line or unit. Production scheduling determines the best operating sequences and the best operating conditions to achieve the inventory and production targets determined by the production plan. Scheduling is done on smaller time horizons (e.g. days or weeks), and includes more operational rules and constraints than planning models. Fig. 1 shows how the length of the time horizon and model accuracy changes, depending on the spatial or time scale, under a hierarchical production planning framework.

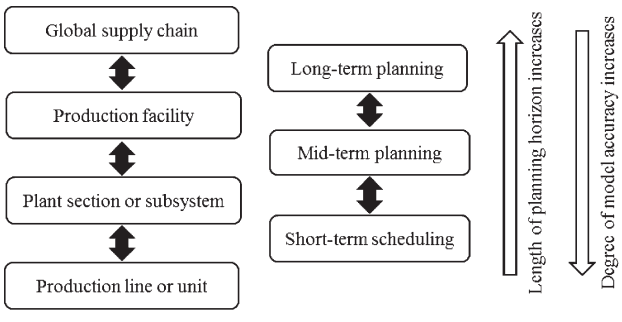

Fig. 1 – *Hierarchical production planning with respect to a) spatial scale, and b) time scale [Adapted from Castillo and Mahalec[17]]*

*Corresponding author: mahalec@mcmaster.ca

Computation of optimal production plans is usually based on linear programming (LP) or successive linear programming (SLP) models[2,3,4] since these approaches have proven to be robust and provide good answers under a variety of circumstances. Production planning models are constructed by dividing the time horizon in a number of periods, the length of which is usually set equal to some calendar unit (e.g. monthly, or quarterly periods). Such approach leads to models which grow in size linearly with the number of periods. If the models are nonlinear, the likelihood of encountering convergence problems increases significantly as the problem size increases. However, formulation of planning models with nonlinearities is becoming more spread in practice[4,5,6,7] in order to compute more accurate solutions and increase profitability. Scheduling is in many cases a NP-complete[8,9] or NP-hard[10] problem, which means that there is no polynomial-bounded algorithm to solve it, even if the underlying supply chain models are linear. Process plants are nonlinear but the usual practice for scheduling is to approximate the system behavior by linear models.[2,11,12,13] Recent advances have made it possible to use nonlinear models for some scheduling problems.[14,15]

MPIP-C algorithm[1] enables use of nonlinear models to solve scheduling problems for plants which produce multiple products by switching from producing one product to another. An example of such system is gasoline blending. The first step is to compute an approximate scheduling solution which imposes some constraints on the detailed scheduling problem, thereby simplifying it and reducing the number of integer variables: it also enables faster solution of detailed scheduling problems by providing additional constraints. MPIP-C algorithm uses inventory-pinch based two-level approach to solve the approximate scheduling problem[16,17] based on nonlinear blending models. The algorithm:

(i) Computes an approximate schedule based on a nonlinear blend model, such that the there is a minimum number of optimal blend recipes along the planning horizon,

(ii) Minimizes the total number of blend instances, and

(iii) Allocates swing tankage.

Solution of the approximate planning model provides additional constraints for the 3rd level continuous-time scheduling problem. If the 3rd level scheduling problem is infeasible, the algorithm resolves a modified approximate scheduling problem, and iterates until a feasible optimal solution is found. We use gasoline blending as an example of the algorithm application. This paper is organized as follows: First, a brief review on prior work on blend planning and scheduling, then we present the problem statement, which is followed by the description of our decomposition approach, the mathematical models used, the MPIP-C algorithm steps, numerical results and conclusions.

Related prior work

In this section, a brief summary of previous works on blend planning and scheduling is presented.

Mendez et al.[3] presented an algorithm to schedule blending operations including blend recipe determination. In order to approximate nonlinear quality constraints, they use linear constraints with correction factors. The algorithm iterates until the correction factors converge. The sequencing problem is avoided since it is assumed that each blender produces only one particular gasoline grade.

Li et al.[18] developed a continuous-time mixed integer linear programming (MILP) model with a common global time grid for all units. The model includes blend recipe optimization, and operational features found in industrial practice such as parallel non-identical blenders, multipurpose tanks, inventory constraints, blender capacity constraints, and order delivery scheduling. Li and Karimi[13] extended the model from Li et al.[18] in order to incorporate blender setup times and consider the case for simultaneous receipt and delivery by product tanks. Li and Karimi[13] used a unique time grid for the time slots of each unit (i.e. tanks and blenders). Solving these models for real-life scale problems (e.g. 8-day horizon, three blenders, and more than 25 orders) to optimality requires large execution times (more than 12 hours for their case studies).

Kolodziej et al.[15] formulated a discrete-time mixed integer nonlinear programming (MINLP) model to solve the pooling problem including inventory, flow, and quality constraints. They developed three different procedures to solve the MINLP problem to global optimality. Two of those algorithms use a radix-based discretization technique to discretize one variable in the bilinear terms and obtain MILP relaxations, and one of them computes better solutions and in less time than commercial MINLP solvers such as BARON and GloMIQO, for their case studies and despite the increment in the number of binary variables in the model.

Glismann and Gruhn[19] developed a two-level method to solve the blend scheduling problem. First, a discrete-time NLP model computes blend recipes and production targets. Then a discrete-time MILP scheduling model, based on a resource-task-network representation, is solved. If the MILP is infeasible, the algorithm re-computes the blend recipes by re-solving the NLP model with extra constraints.

Thakral and Mahalec[20] presented an iterative algorithm that optimizes blend recipes using a multi-period MILP planning model, then minimizes blender switches through the use of a genetic algorithm, and subsequently detects infeasibilities via agent-based simulation. The algorithm is repeated until a feasible solution is found.

Inventory pinch analysis in production planning and scheduling

Singhvi and Shenoy[21] presented a pinch analysis approach to solve an aggregate production planning problem. They constructed composite curves for the demand and production amounts in order to locate the pinch as the point where both curves touch. The idea is to determine the composite production curve that meets the demand and the given inventory policy. Production targets are determined directly from the composite curves and imposed as constraints into a MILP model that minimizes the material, inventory, and labor costs, subject to material balance equations and production capacity constraints. Singhvi et al.[22] extended the work from Singhvi and Shenoy[21] in order to handle multiple products and solve the product sequencing problem based on some heuristic rules and assuming that the demand is met only at the end of the horizon. Ludwig et al.[23] applied the pinch analysis of Singhvi and Shenoy[21] to a process with seasonal supply, and they included the constraint that the composite production curve must not cross either the composite demand curve or the composite supply curve. Foo et al.[24] implemented the algebraic technique equivalent to the graphical pinch methodology from Singhvi and Shenoy[21] for a single product process, and included minimum and maximum product inventory constraints and the capability of scheduling a plant shut down.

Castillo et al.[16] defined an inventory pinch as the point in time where the composite total demand (CTD) curve touches the composite average total production curve (CATP). The CATP curve can be defined as the concave envelope of the CTD curve, with the difference that the CATP curve starts at the total initial product inventory. Fig. 2 shows an example with two inventory pinch points. In contrast with previous approaches to solve planning problems using composite curves, the CATP curve does not represent the actual production rates; it is only used to locate the pinch points, as well as the minimum production volumes to satisfy the demand in each interval delineated by pinch points. The idea is that between pinch points, the same optimal blend recipes can be used to generate a feasible blend plan. This is not always the case due to other operational constraints, and Castillo et al.[16] developed an iterative approach to eliminate such infeasibilities.

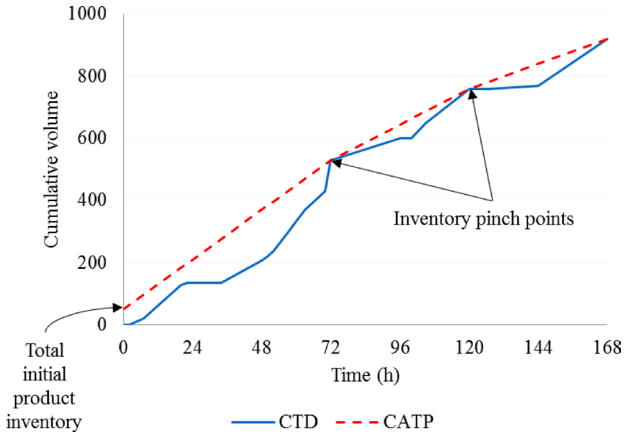

Fig. 2 – *Two inventory pinch points on the cumulative curves*

Castillo et al.[16] introduced a two-level inventory pinch-based decomposition approach to solve multi-period gasoline blend planning models with nonlinear blending rules. The blend recipes are computed at the first level by solving a discrete-time NLP model which only considers the material balance equations around the blenders and storage tanks, the linear and nonlinear quality equations, the minimum and maximum inventory and production capacity constraints, and product quality specifications. Then, these recipes are fixed at the second level which consists of a discrete-time MILP model. The integer variables are required to enforce minimum production thresholds at each time period. The number of time periods at the first level is delineated by the inventory pinch points, and changes in the quality and price of blend components, while the number of time periods at the second level is determined by the planner/scheduler. For the case studies presented by Castillo et al.[16], this approach computes solutions that are optimal or close to the optimum, and with rapid execution times compared with MINLP solvers. Castillo and Mahalec[17,25] extended the algorithm from Castillo et al.[16] to solve blend scheduling problems by introducing a third level to solve the scheduling problem using a discrete-time MILP model. Most recently, Castillo and Mahalec[1] developed a continuous-time model to solve the third level more efficiently.

Problem statement

Given a scheduling horizon [0, H], a blending system constituted of storage tanks and blenders (see Fig. 3 for an example), and sets of blend components, products, and delivery orders which must be fulfilled during specific time delivery windows along the horizon, the objective is to minimize the total cost which consists of the cost of the blended materials (i.e. the blend cost) and the switching costs comprised of the number of different blend runs, product changeovers in the storage tanks, and

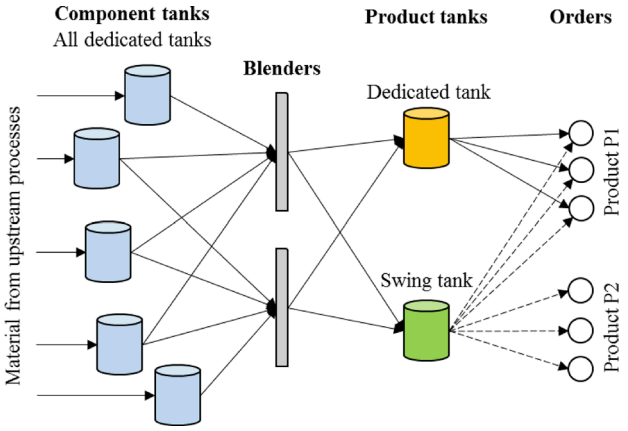

Fig. 3 – *Sample blending system*

number of deliveries from different tanks to the same order. The cost, quality properties, initial inventories, and flow rates into the system of the blend components are known. Minimum and maximum product quality specifications are given. Initial product inventories are known and on specification. Blend components have dedicated storage tanks. Storage tanks for the finished products can be dedicated tanks or swing tanks (i.e. they can store different products, but not at the same time). Product changeover times are negligible for swing tanks. All storage tanks have minimum and maximum capacity constraints. Parallel non-identical blenders are considered, with minimum and maximum blending capacities. Minimum blend size, minimum blend length, and minimum setup time constraints are assumed for each blender and they are product-dependent. Perfect mixing is assumed to occur in the blenders and a blender can only feed one product tank at any time, since this is industrial practice. Each order involves only one product (one original order involving different products can be broken into orders of each specific product).

It is required to determine the blend recipes, the production sequence of each blender, the blending rates of each blend run, the delivery sequence of each product tank, the start and end times of the blend runs and delivery runs, the product allocation of swing tankage, and the inventory profiles of all tanks along the horizon.

Decomposition strategy

In this work we use the decomposition framework from Castillo and Mahalec[1,25]. It is assumed that a long-term (e.g. twelve months) production planning model has been optimized for the refinery or the gasoline blend system. In addition, it is assumed that the long-term plan determines which feedstocks are to be processed, which products are produced, and the total amount for each of them, respectively. Results of the long-term plan are used to optimize a short-term discrete-time multi-period approximate scheduling model (e.g. thirty days), which in turn provide constraints for a continuous-time scheduling model (see Fig. 4).

Approximate scheduling

The purpose of the approximate scheduling model is to minimize the production costs or to maximize the profit for a given selection and quan-

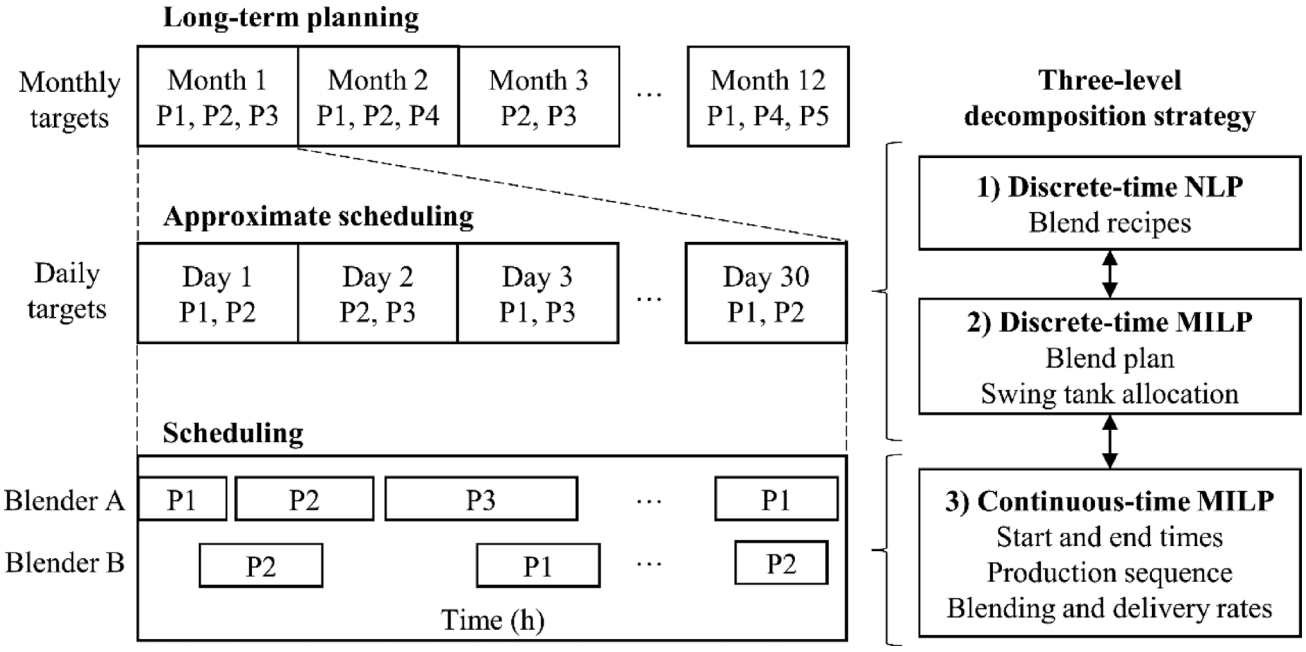

Fig. 4 – *Three-level decomposition approach*

tities of the feedstocks. In addition, the approximate scheduling model:

– Determines how much to produce of each product and each time period.

– Manages allocation of the swing storage capacities; it is assumed that a swing storage capacity can be re-allocated only at the beginning of each period.

– Minimizes the number of instances of different operating modes for each multi-mode unit, e.g. maximum gasoline or maximum diesel for an FCC unit, or number of different grade gasoline blenders for a gasoline blender. In order to eliminate instances of minute production for any operating mode, a minimum threshold production constraint is specified for each production unit.

Another difference between approximate scheduling and detailed scheduling models is that tasks and resources are still aggregated in some way at the short-term planning level. Therefore, solution of approximate scheduling determines a production and inventory profile which is feasible only at the period boundaries (e.g. end of each day).

Objective function at this level includes in general case the costs of carrying the inventories and the switching costs associated with the discrete variable (decisions). Cost of raw materials plus operating costs computed at this level are the same as at the top level, since the operating modes and the total feed amounts are fixed by the top level.

Detailed scheduling

Detailed scheduling model allows only one task in any production unit at any given time, thus guarantying a feasible solution along the entire horizon. Objective function at the scheduling level includes only the switching costs and in general case the costs of transitions from one mode of operation to another. Solution of the approximate scheduling model (i.e. approximate scheduling model) leads to a neighborhood where the optimal schedule is located. This neighborhood is described to the detailed scheduling task in terms of target inventories at the end of each day, operating modes at which each process unit needs to operate during each day, etc. From these data, computation of the best schedule can proceed in several different ways, for example:

a) Determine operating sequence with a minimum number of switches via some evolutionary algorithm and verify its feasibility via simulation (e.g. Thakral and Mahalec[20]).

b) Use fine-grid discrete-time MILP scheduling model (e.g. Castillo and Mahalec[25]).

c) Solve a continuous-time MILP scheduling model (e.g. Castillo and Mahalec[1]).

In this work, we use option c), since the continuous-time model has a smaller size than its corresponding discrete-time version, thus allowing solving of problems in less time. When using the constraints imposed by the approximate scheduling level, the detailed scheduling level is solved much faster. However, due to the consequent reduction in the feasible solution set, optimality of the computed schedule with respect to the original problem (i.e. no constraints from the approximate scheduling level) is not guaranteed, although the optimality gap is expected to be very small. One way to improve this solution is to use it as a starting point for the scheduling model that includes quality constraints and allow the blend recipes to differ from those previously computed at the short-term planning level. The disadvantage is that the resulting MILP or MINLP model may require prohibitive execution times in order to close the optimality gap. Future work will be dedicated to address this issue and compute proven optimal solutions with respect to the original problem.

Mathematical models

In this section, the mathematical models are described. Due to the large number of equations, only the nonlinear equations for Reid vapor pressure (RVP) used in this work are here shown. The reader is encouraged to look at the references where the complete models are described in detail.

1st level model[17] (discrete-time LP or NLP)

The 1st level determines the optimal blend recipes that will be fixed at the next levels. The time periods of this model are the L1-periods. The objective function minimizes the blend cost and the cost associated with slack variables. Slack variables are included in the inventory balances and their penalty coefficients in the objective function are greater than the cost coefficients of the blend components. Therefore, if a problem has a physically feasible solution, the slack variables will be equal to zero. If a problem has no physically feasible solution, the solver will compute a numerical feasible solution with some non-zero slack variables. The values of the blending capacity, component supply, and product demand are the corresponding aggregated values for each of the L1-periods. In addition, individual product tanks are aggregated into product pools, and individual blenders are lumped into a single one.

– Blend recipe optimization –

min BlendCost_{L1} + SlackCost_{L1}
s.t.
Inventory balances (component tanks, product pools)
Inventory constraints (component tanks, product pools)
Quality constraints (linear and nonlinear)
Maximum production capacity (lumped blenders)

The solution of the 1st level model is a lower bound of the global blend cost since the volumes to be blended in each L1-period are the minimum amount required to fulfill the demand.

The nonlinear quality constraints used in this work are presented next. Equation (1) and (2) define the blend recipes. Equation (3) was formulated based on the nonlinear blending rule used by Singh et al.[26] Equation (4) represents the minimum and maximum product quality specifications for RVP property.

$$r(i,p,k) = \frac{V_{comp,L1}(i,p,k)}{V_{blend,L1}(p,k)} \quad \forall\, i,p,k \geq 1 \tag{1}$$

$$\sum_i r(i,p,k) = 1 \quad \forall\, p,k \geq 1 \tag{2}$$

$$Q_{pr}(p,\text{'RVP'},k) = \left[\sum_i Q_{bc}(i,\text{'RVP'},k)^{1.25} \cdot r(i,p,k)\right]^{0.8} \quad \forall\, p,k \geq 1 \tag{3}$$

$$Q_{pr}^{\min}(p,e) \leq Q_{pr}(p,e,k) \leq Q_{pr}^{\max}(p,e) \quad \forall\, e = \{\text{RVP}\}, p,k \geq 1 \tag{4}$$

2nd level model[1,25] (discrete-time MILP)

The 2nd level computes the blend plan, delivery plan, and service allocation of swing tanks. The time periods of this model are the L2-periods. The blend plan determines how much to blend of each product, in which blender, and in each L2-period. The delivery plan defines how much product to send from each individual product tank to each specific order and in which L2-period. The allocation of swing tanks establishes which product can be stored in each individual product tank and in which L2-periods.

The 2nd level model is linear since the blend recipes from the 1st level are not computed at this level (i.e. the blend recipes of each L1-period are fixed in the corresponding L2-periods). Equations regarding individual product tanks and individual blenders are included. Moreover, constraints such as minimum blend size, minimum blend length, and minimum setup time are incorporated. The values of the blending capacity, component supply, and product demand are the corresponding aggregated values for each of the L2-periods. The individual blenders can produce more than one product in each L2-period. A swing tank can only store one specific product during a L2-period.

The 2nd level model is solved in two phases: feasibility of the blend recipes is checked first, and then a solution with minimum number of switches is computed. The reason to do this is to detect inventory infeasibilities (i.e. non-zero slacks) rapidly.

The 2nd level model for feasibility check has slack variables in the inventory balances and the objective is to minimize such variables. If a feasible operation can be obtained using the blend recipes from the 1st level, inventory slack variables will be zero at the solution of the 2nd 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. In order to extend the use of a given blend recipe, the penalty coefficients for the slack variables must decrease along the time horizon as fast as possible and a significant change must take place after each L1-period boundary.

– Feasibility check of blend recipes from 1st level –

min SlackCost_{L2}
s.t.
Fixed recipes (from 1st level)
Product inventory targets (from 1st level)
Inventory balances (component tanks, product pools, product tanks)
Inventory constraints (component tanks, product tanks)
Maximum production capacity (individual blenders)
Minimum product-dependent blend size (individual blenders)
Minimum product-dependent run length (individual blenders)
Minimum product-dependent setup time (individual blenders)
Maximum delivery rates of each individual product tank

When the solution of this phase has inventory infeasibilities, the algorithm will subdivide the required L1-period at the time corresponding to the end boundary of the L2-period with the first infeasibility. Then, the 1st level model is re-solved and a new set of blend recipes is computed.

The 2nd level model for minimizing switching costs (i.e. number of blend runs, number of product changeovers in the swing tanks, and number of de-

livery runs from different tanks to the same order) does not include any slack variables.

– Computing blend plan, delivery plan, and swing tank allocation with minimum number of switches –

min $SwitchingCost_{L2}$
s.t.
$SwitchingCost_{L2} = SwitchBlenderCost_{L2} +$
$+ SwitchTankCost_{L2} + SwitchDeliverCost_{L2}$
Fixed recipes (from 1st level)
Product inventory targets (from 1st level)
Inventory balances (component tanks, product pools, product tanks)
Inventory constraints (component tanks, product tanks)
Maximum production capacity (individual blenders)
Minimum product-dependent blend size (individual blenders)
Minimum product-dependent run length (individual blenders)
Minimum product-dependent setup time (individual blenders)
Maximum delivery rates of each individual product tank

3rd level model[1] (continuous-time MILP)

The 3rd level determines:
- Start and end times of all tasks,
- Production sequence of each blender,
- Blending rates for each blend run,
- Delivery sequence of all product tanks, and
- Delivery rates for each delivery run.

The blend recipes from the 1st level, the swing tank allocation, and the delivery and blend plans from the 2nd level are fixed. Using information from the 2nd level decreases the search space and the model size at the 3rd level. The time slots are unit slots, meaning that each unit (i.e. component tanks, blenders, and product tanks) have their own specific time grid. From here on, the term time slot and unit slot will be used interchangeably. The 3rd level is formulated as a continuous-time model in order to have a smaller model size with less discrete variables than a corresponding discrete-time model. The 3rd level model is based on the continuous-time model presented by Li and Karimi[13] which has been modified by modifying some constraints and adding new constraints in order to avoid infeasible solutions[1].

Only one specific product can be produced by a blender in a given time slot, as well as only one product can be stored in a swing tank. The 3rd level model includes slack variables in the inventory balances and demand constraints. The penalty profile for the slack variables decreases along the scheduling horizon. For large-scale problems, the scheduling horizon can be divided in various subintervals, denoted as *L*-intervals. Then, the 3rd level MILP model is solved sequentially for each *L*-interval. The boundaries of the *L*-intervals must coincide with the boundaries of some *L2*-periods to enable the inventory levels computed at the 2nd level to be fixed at the start and end boundaries of the L-intervals.

Similarly as with the 2nd level, the 3rd level is solved in more than one phase. First, a feasibility check is carried out in order to determine if the blend recipes from the 1st level and the constraints from the 2nd level can produce a feasible schedule, and if the number of time slots can produce a feasible solution. When the feasibility check does not find a physically feasible schedule, the slack variables have non-zero values at the solution. If that is the case, the required L1-period is subdivided at the time corresponding to the end boundary of the L2-period associated with the time slot containing the first infeasibility. After a feasible solution is found, then an optimization phase computes the production and delivery sequences with minimum number of blend runs and delivery runs. Finally, with the production and delivery sequences fixed, blending rate variations are minimized. The 3rd level model for the feasibility phase is denoted as F-SimRD-PlanTDBR-feas, the one for the optimization phase is referred to as F- SimRD-PlanTDBR-opt, and the model to reduce variations in the blending rates is designated as F-SimRD-PlanTDBR-adj. The letter F means fixed recipes, SimRD indicates that product tanks can receive and deliver material simultaneously, PlanTDBR signifies that the plan computed at the 2nd level is enforced, and the last word indicates if it corresponds to the feasibility phase (feas), optimization phase (opt), or the adjustment phase (adj) to reduce blending rate variations.

– Feasibility check of blend recipes from 1st level and constraints from 2nd level –

Model F-SimRD-PlanTDBR-feas
min $SlackCost_{L3}$
s.t.
Fixed recipes (from 1st level)
Product inventory targets (from 1st and 2nd levels)
Fixed swing tank allocation (from 2nd level)
Blend plan constraints (from 2nd level)
Inventory balances (component tanks, product tanks)
Inventory constraints (component tanks, product tanks)

Maximum production capacity (individual blenders)
Minimum product-dependent blend size (individual blenders)
Minimum product-dependent run length (individual blenders)
Minimum product-dependent setup time (individual blenders)
Maximum delivery rates of each individual product tank

– Computing production and delivery sequences with minimum number of switches –

Model F-SimRD-PlanTDBR-opt
min $BlendCost_{L3}$ + $SwitchingCost_{L3}$ + $SlackCost_{L3}$
s.t.
$SwitchingCost_{L3}$ = $SwitchBlenderCost_{L3}$ + $SwitchDeliverCost_{L3}$
Fixed recipes (from 1st level)
Product inventory targets (from 1st and 2nd levels)
Fixed swing tank allocation (from 2nd level)
Blend plan constraints (from 2nd level)
Inventory balances (component tanks, product tanks)
Inventory constraints (component tanks, product tanks)
Maximum production capacity (individual blenders)
Minimum product-dependent blend size (individual blenders)
Minimum product-dependent run length (individual blenders)
Minimum product-dependent setup time (individual blenders)
Maximum delivery rates of each individual product tank

– Minimizing blending rate variations on each blend run –

Model F-SimRD-PlanTDBR-adj
min BlendRateVariations
s.t.
$BlendRateVariations = \Sigma_{bl,n}\ VolumeDifference_{L3}(bl,n)$
$VolumeDifference_{L3}(bl,n) \geq AvgBlendRate(bl,n) \cdot BlendLength_{L3}(bl,n) - VolumeBlended_{L3}(bl,n)$
$VolumeDifference_{L3}(bl,n) \geq VolumeBlended_{L3}(bl,n) - AvgBlendRate(bl,n) \cdot BlendLength_{L3}(bl,n)$
Fixed production sequence
Fixed delivery sequence
Fixed recipes (from 1st level)
Product inventory targets (from 1st and 2nd levels)
Fixed swing tank allocation (from 2nd level)
Blend plan constraints (from 2nd level)
Inventory balances (component tanks, product tanks)
Inventory constraints (component tanks, product tanks)
Maximum production capacity (individual blenders)
Minimum product-dependent blend size (individual blenders)
Minimum product-dependent run length (individual blenders)
Minimum product-dependent setup time (individual blenders)
Maximum delivery rates of each individual product tank

Description of the MPIP-C algorithm

The multiperiod inventory pinch algorithm with a continuous-time scheduling model at the 3rd level for scheduling of blend operations is described next. Although it was developed for gasoline blending, it can be used in any blending process where the quality properties of the blend components are known in advance.

Step 1) Determine the pinch point(s) location on the cumulative curves (CTD and CATP).

Step 2) Set the number of *L1*-periods. The boundaries of these time periods are delineated based on:

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

Step 3) Set the number of *L2*-periods. The boundaries of these time periods are based on:

– Boundaries of the *L1*-periods.
– Variations in supply profile of blend components.
– Time delivery windows for the demand orders.
– Expected time that a swing tank remains in one specific service.
– Time to produce the minimum allowed blend run.

Step 4) Solve the 1st level model. If slack variables have non-zero values at the solution, the problem is infeasible; otherwise, go to Step 5.

Step 5) Solve the 2nd level model for the feasibility check.

Step 6) Continue to Step 8 if solution from Step 5 has all slack variables with zero values; otherwise, go to Step 7.

Step 7) Determine the *L2*-period with the first non-zero slack variable along the horizon and use its end boundary to delineate a new boundary at the 1st level. Return to Step 4.

Step 8) Solve the 2nd level model for minimizing switching costs.

Step 9) Set the number of time slots (N) assigned to the scheduling horizon.

Step 10) Solve model F-SimRD-PlanTDBR-feas.

Step 11) If all slack variables are equal to zero, go to Step 13; otherwise, add more time slots and return to Step 10.

If the number of time slots exceeds the maximum limit, subdivide an *L1*-period and go back to Step 4.

Step 12) Solve model F-SimRD-PlanTDBR-opt.

Step 13) Fix the production and delivery sequences and compute the average blending rates of each blend run.

Step 14) Solve model F-SimRD-PlanTDBR-adj to minimize blending rates variations across a blend run constituted by more than one time slot.

Numerical results

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. The data for all problems appear in Castillo and Mahalec[25]. Table 1 shows the physical size of these problems, and Table 2 shows the size of the corresponding continuous-time MILP scheduling model used at the 3rd level. The 3rd level was solved in five L-intervals (i.e. subintervals of the scheduling horizon, one model instance for each one), and Table 2 only shows the size of the largest model. For all case studies, the scheduling horizon is 14 days, the first L-interval is constituted of the first 4 days, the second L-interval spans day 5 to day 7, the third L-interval is composed of day 8 and 9, the fourth L-interval spans day 10 to day 12, while day 13 and 14 constitute the fifth L-interval. The Reid vapor pressure is the only nonlinear quality property considered.

Table 1 – *Attributes of the case studies*

Case Study ID	# Blenders	# Orders	# Quality properties	# Products	# Product tanks (swing tanks)
8	1	36	8	3	6 (3)
9	1	36	8	3	6 (3)
10	1	37	8	3	6 (3)
11	2	37	8	3	6 (3)
12	2	35	8	3	6 (3)
13	2	37	8	3	6 (3)
14	3	37	8	3	6 (3)

Table 2 – *Size of model F-SimRD-PlanTDBR-opt for the largest model at the 3rd level*

Case Study	# Slots (Entire horizon)	# Slots (Largest model)	# Equations	# Continuous Variables	# Binary Variables
8	47	11	3,755	2,267	215
9	51	16	5,525	3,206	316
10	63	15	4,974	3,007	284
11	43	11	5,418	2,599	384
12	39	10	5,064	2,423	357
13	46	11	5,613	2,743	393
14	44	11	7,336	3,124	564

Table 3 – *Results from MPIP-C and MPIP[a] algorithms. Solvers: IPOPT and CPLEX 12.3*

Case Study	Algorithm/ MINLP solver	# L-intervals at the 3rd level	Obj. Func. Value ($)	Blend cost ($)	# Blend runs	# Delivery runs	CPU time (s)	Gap (%)
8	MPIP-C	5	38,483.4	37,943.4	17	40	298.4	0.79
	MPIP[a]	7	38,498.4	37,943.4	17	43	981.0	0.82
9	MPIP-C	5	39,304.2	38,754.2	18	38	124.0	0.79
	MPIP[a]	7	39,394.2	38,754.2	21	44	3,477.0	1.03
10	MPIP-C	5	39,020.2	38,405.2	21	39	593.2	0.96
	MPIP[a]	7	39,050.2	38,405.2	21	45	2,430.0	1.03
11	MPIP-C	5	38,935.2	38,405.2	17	38	139.3	0.74
	MPIP[a]	7	39,035.2	38,405.2	20	46	5,106.0	1.00
12	MPIP-C	5	38,633.4	38,073.4	19	36	77.4	0.85
	MPIP[a]	7	38,733.4	38,073.4	23	40	3,666.0	1.11
13	MPIP-C	5	38,384.5	37,784.5	20	40	341.5	0.93
	MPIP[a]	7	38,519.5	37,784.5	25	47	1,923.0	1.29
14	MPIP-C	5	38,421.4	37,796.4	22	37	206.1	1.00
	MPIP[a]	7	38,546.4	37,796.4	26	46	2,455.0	1.33

[a]MPIP scheduling algorithm from Castillo and Mahalec[25]

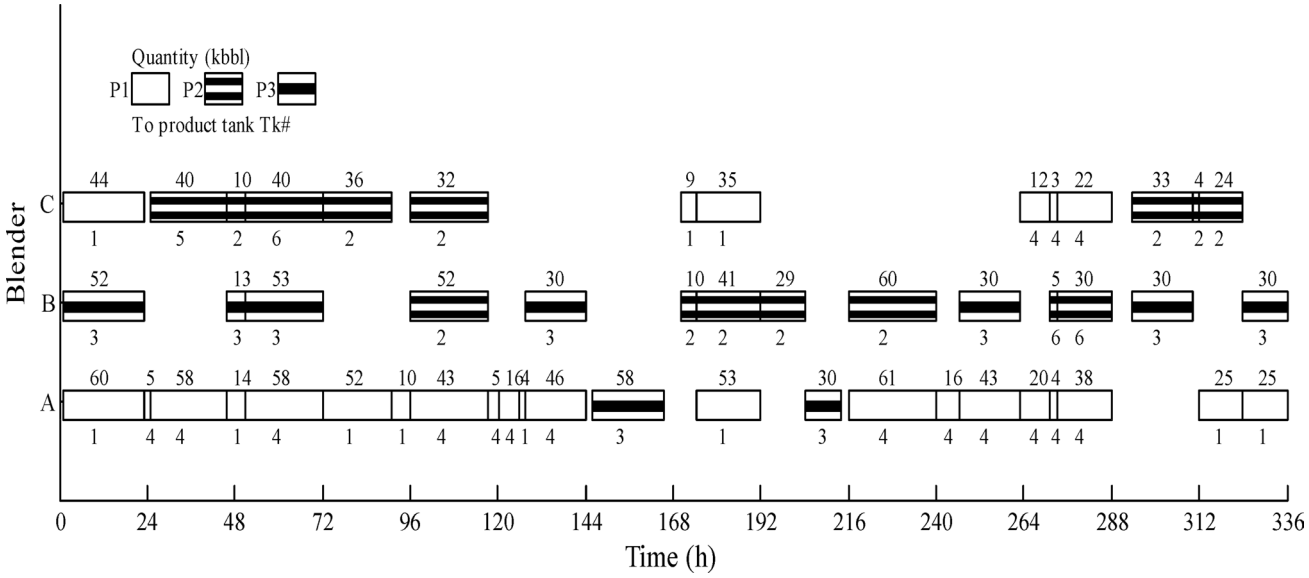

Fig. 5 – *Production sequence, MPIP-C solution for case study 14*

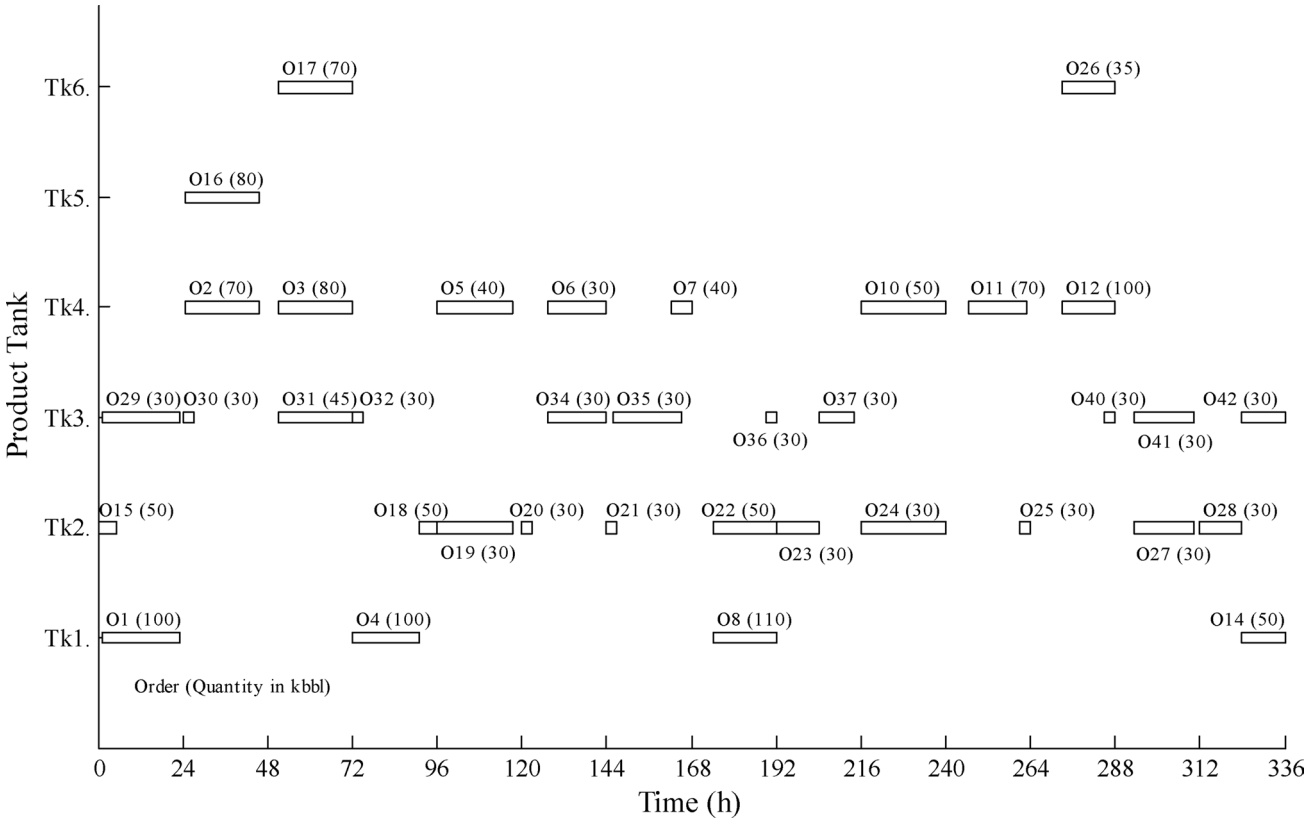

Fig. 6 – *Delivery sequence, MPIP-C solution for case study 14*

Table 3 presents the results obtained by MPIP-C algorithm and compares them with those obtained by MPIP algorithm (from Castillo and Mahalec[25]) which uses a discrete-time MILP scheduling model at the 3rd level. It can be observed that MPIP-C algorithm outperforms MPIP algorithm in both solution quality and execution time. This is due to the reduction in the model size by using a continuous-time scheduling model, which allows subdivision of the scheduling horizon into larger L-intervals and solving them more efficiently.

For illustration purposes, solution for case study 14 is presented. Fig. 5 and Fig. 6 show the production and delivery sequences, respectively.

Conclusions

This work uses MPIP-C algorithm for scheduling of multipurpose systems such as blending operations in petroleum refineries. MPIP-C scheduling algorithm decomposes the problem into three different decision levels. By using information from the upper levels, good quality schedules can be computed with short execution times.

The MPIP-C algorithm uses the inventory pinch concept to compute optimal blend recipes at the 1st level. This computation can be based on either linear or nonlinear models. The 1st level uses aggregated representation of the system, i.e. all blenders are treated as a lumped capacity, all production and demands are lumped across the time periods delineated by the inventory pinch points. Blend recipes computed at the 1st level are employed at the 2nd level to compute an optimal production plan which includes allocation of swing inventories and association of product deliveries with specific tanks. Scheduling of blend runs and delivery liftings is carried out at the 3rd level using a continuous-time MILP model. Solution from the upper levels provides additional information which helps reduce the model size and computational times to solve the 3rd level model.

Seven case studies were presented. MPIP-C computes better solutions and in less time than the previously published MPIP algorithm. The use of a continuous-time MILP scheduling model enables more efficient solving of larger subintervals (i.e. L-intervals) of the scheduling horizon compared with the corresponding discrete-time model.

Nomenclature

Sets and indices

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)

N = $\{n \mid 0, 1, \ldots, N\}$ – Time slots assigned for the entire horizon (3rd level continuous-time model)

O = $\{o\}$ – All demand orders

P = $\{p\}$ – Different products

Parameters

H – Length of the entire scheduling horizon

$Q_{bc,L1}(i,e,k)$ – 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

Continuous variables

$Q_{pr,L1}(p,e,k)$ – Quality e of product p during *L1*-period k

$r(i,p,k)$ – Continuous variable at the 1st level, but a parameter at the 2nd and 3rd levels. Blend recipe for product p in *L1*-period k

$V_{blend,L1}(p,k)$ – Volume blended of product p during *L1*-period k

$V_{comp,L1}(i,p,k)$ – Volume of component i used in product p during *L1*-period k

BlendCost_{L1} – Cost of the materials used in the blend (1st level solution)

BlendCost_{L2} – Cost of the materials used in the blend (2nd level solution)

BlendCost_{L3} – Cost of the materials used in the blend (3rd level solution)

SlackCost_{L1} – Penalty for the non-zero slack variables (1st level solution)

SlackCost_{L2} – Penalty for the non-zero slack variables (2nd level solution)

SlackCost_{L3} – Penalty for the non-zero slack variables (3rd level solution)

$\text{SwitchingCost}_{L2}$ – Cost associated with the switching operations (2nd level solution)

$\text{SwitchingCost}_{L3}$ – Cost associated with the switching operations (3rd level solution)

$\text{SwitchBlenderCost}_{L2}$ – Cost associated with the blend runs (2nd level solution)

$\text{SwitchBlenderCost}_{L3}$ – Cost associated with the blend runs (3rd level solution)

$\text{SwitchDeliverCost}_{L2}$ – Penalty associated with delivering the same order from different tanks (2nd level solution)

$\text{SwitchDeliverCost}_{L3}$ – Penalty associated with delivering the same order from different tanks (3rd level solution)

$\text{SwitchTankCost}_{L2}$ – Penalty associated with product changeovers in the swing tanks (2nd level solution)

BlendRateVariations – Term to minimize in order to reduce variations in the blending rate of a blend run

$\text{VolumeDifference}_{L3}(bl,n)$ – Volume difference in blender bl during slot n between the virtual blend run using the average blending rate and the actual blend run

AvgBlendRate(bl,n) – Average blending rate of a blend run in blender bl during slot n

$\text{BlendLength}_{L3}(bl,n)$ – Duration of the blend in blender bl during slot n

$\text{VolumeBlended}_{L3}(bl,n)$ – Actual volume blended in blender bl in slot n

References

1. *Castillo-Castillo, P. A., Mahalec, V.*, Scheduling of nonlinear blending processes via inventory pinch algorithm combining discrete- and continuous-time models, Comput. Chem. Eng. (submitted).
2. *Jia, Z., Ierapetritou, M.*, Mixed-integer linear programming model for gasoline blending and distribution scheduling, Ind. Eng. Chem. Res. **42** (2003) 825. http://dx.doi.org/10.1021/ie0204843
3. *Mendez, C. A., Grossmann, I. E., Harjunkoski, I., Kabore, P.*, A simultaneous optimization approach for off-line blending and scheduling of oil refinery operations, Comput. Chem. Eng. **30** (2006) 614. http://dx.doi.org/10.1016/j.compchemeng.2005.11.004
4. *Menezes, B. C., Kelly, J. D., Grossmann, I. E.*, Improved swing-cut modeling for planning and scheduling of oil-refinery distillation units, Ind. Eng. Chem. Res. **52** (2013) 18324. http://dx.doi.org/10.1021/ie4025775
5. *Kelly, J. D.*, Formulating production planning models, Chem. Eng. Prog. **100** (2004) 43. http://dx.doi.org/10.1016/j.cej.2003.11.027
6. *Alhajri, I., Elkamel, A., Albahri, T., Douglas, P. L.*, A nonlinear programming model for refinery planning and optimisation with rigorous process models and product quality specifications, Int. J. Oil Gas Coal Technol. **1** (2008) 283. http://dx.doi.org/10.1504/IJOGCT.2008.019846
7. *Elkamel, A., Ba-Shammakh, M., Douglas, P., Croiset, E.*, An optimization approach for integrating planning and CO_2 emission reduction in the petroleum refining industry, Ind. Eng. Chem. Res. **47** (2008) 760. http://dx.doi.org/10.1021/ie070426n
8. *Birewar, D. B., Grossmann, I. E.*, Simultaneous production planning and scheduling in multiproduct batch plants, Ind. Eng. Chem. Res. **29** (1990) 570. http://dx.doi.org/10.1021/ie00100a013
9. *Pinto, J. M., Joly, M., Moro, L. F. L.*, Planning and scheduling models for refinery operations, Comput. Chem. Eng. **24** (2000) 2259. http://dx.doi.org/10.1016/S0098-1354(00)00571-8
10. *Terrazas-Moreno, S., Grossmann, I. E.*, A multiscale decomposition method for the optimal planning and scheduling of multi-site continuous multiproduct plants, Chem. Eng. Sci. **66** (2011) 4307. http://dx.doi.org/10.1016/j.ces.2011.03.017
11. *Gothe-Lundgren, M., Lundgren, J. T., Persson, J. A.*, An optimization model for refinery production scheduling, Int. J. Prod. Econ. **78** (2002) 255. http://dx.doi.org/10.1016/S0925-5273(00)00162-6
12. *Jia, Z., Ierapetritou, M.*, Efficient short-term scheduling of refinery operations based on a continuous-time formulation, Comput. Chem. Eng. **28** (2004) 1001. http://dx.doi.org/10.1016/j.compchemeng.2003.09.007
13. *Li, J., Karimi, I. A.*, Scheduling gasoline blending operations from recipe determination to shipping using unit slots, Ind. Eng. Chem. Res. **50** (2011) 9156. http://dx.doi.org/10.1021/ie102321b
14. *Li, J., Misener, R., Floudas, C. A.*, Continuous-time modeling and global optimization approach for scheduling of crude oil operations, AIChE J. **58** (2012) 205. http://dx.doi.org/10.1002/aic.12623
15. *Kolodziej, S. P., Grossmann, I. E., Furman, K. C., Sawayac, N. W.*, A discretization-based approach for the optimization of the multiperiod blend scheduling problem, Comput. Chem. Eng. **53** (2013) 122. http://dx.doi.org/10.1016/j.compchemeng.2013.01.016
16. *Castillo, P. A., Kelly, J. D., Mahalec, V.*, Inventory pinch algorithm for gasoline blend planning, AIChE J. **59** (2013) 3748. http://dx.doi.org/10.1002/aic.14113
17. *Castillo, P. A., Mahalec, V.*, Inventory pinch based, multiscale models for integrated planning and scheduling-part I: Gasoline blend planning, AIChE J. **60** (2014) 2158. http://dx.doi.org/10.1002/aic.14423
18. *Li, J., Karimi, I. A., Srinivasan, R.*, Recipe determination and scheduling of gasoline blending operations, AIChE J. **56** (2010) 441.
19. *Glismann, K., Gruhn, G.*, Short-term scheduling and recipe optimization of blending processes, Comput. Chem. Eng. **25** (2001) 627. http://dx.doi.org/10.1016/S0098-1354(01)00643-3
20. *Thakral, A., Mahalec, V.*, Composite planning and scheduling algorithm addressing intra-period infeasibilities of gasoline blend planning models, Can. J. Chem. Eng. **91** (2013) 1244. http://dx.doi.org/10.1002/cjce.21766
21. *Singhvi, A., Shenoy, U. V.*, Aggregate planning in supply chains by pinch analysis, Trans. IChemE A., **80** (2002) 597. http://dx.doi.org/10.1205/026387602760312791
22. *Singhvi, A., Madhavan, K. P., Shenoy, U. V.*, Pinch analysis for aggregate production planning in supply chains, Comput. Chem. Eng. **28** (2004) 993. http://dx.doi.org/10.1016/j.compchemeng.2003.09.006
23. *Ludwig, J., Treitz, M., Rentz, O., Geldermann, J.*, Production planning by pinch analysis for biomass use in dynamic and seasonal markets, Int. J. Prod. Res. **47** (2009) 2079. http://dx.doi.org/10.1080/00207540802392577
24. *Foo, D. C. Y., Ooi, M. B. L., Tan, R. R., Tan, J. S.*, A heuristic-based algebraic targeting technique for aggregate planning in supply chains, Comput. Chem. Eng. **32** (2008) 2217. http://dx.doi.org/10.1016/j.compchemeng.2007.10.016
25. *Castillo, P. A., Mahalec, V.*, Inventory pinch based, multiscale models for integrated planning and scheduling-part II: Gasoline blend scheduling, AIChE J. **60** (2014) 2475. http://dx.doi.org/10.1002/aic.14444
26. *Singh, A., Forbes, J. F., Vermeer, P. J., Woo, S. S.*, Model-based real-time optimization of automotive gasoline blending operations, J. Process Control **10** (2000) 43. http://dx.doi.org/10.1016/S0959-1524(99)00037-2