Contents lists available at ScienceDirect

Engineering

journal homepage: www.elsevier.com/locate/eng

Research
Smart Process Manufacturing—Article

Global Optimization of Nonlinear Blend-Scheduling Problems

Pedro A. Castillo Castillo[a], Pedro M. Castro[b], Vladimir Mahalec[a,*]

[a] Department of Chemical Engineering, McMaster University, Hamilton, ON L8S 4L8, Canada
[b] Center for Mathematics, Fundamental Applications and Operations Research, Faculty of Sciences, University of Lisbon, Lisbon 1749-016, Portugal

A R T I C L E I N F O

Article history:
Received 7 December 2016
Revised 16 February 2017
Accepted 20 February 2017
Available online 28 March 2017

Keywords:
Global optimization
Nonlinear gasoline blending
Continuous-time scheduling model
Piecewise linear relaxations

A B S T R A C T

The scheduling of gasoline-blending operations is an important problem in the oil refining industry. This problem not only exhibits the combinatorial nature that is intrinsic to scheduling problems, but also non-convex nonlinear behavior, due to the blending of various materials with different quality properties. In this work, a global optimization algorithm is proposed to solve a previously published continuous-time mixed-integer nonlinear scheduling model for gasoline blending. The model includes blend recipe optimization, the distribution problem, and several important operational features and constraints. The algorithm employs piecewise McCormick relaxation (PMCR) and normalized multiparametric disaggregation technique (NMDT) to compute estimates of the global optimum. These techniques partition the domain of one of the variables in a bilinear term and generate convex relaxations for each partition. By increasing the number of partitions and reducing the domain of the variables, the algorithm is able to refine the estimates of the global solution. The algorithm is compared to two commercial global solvers and two heuristic methods by solving four examples from the literature. Results show that the proposed global optimization algorithm performs on par with commercial solvers but is not as fast as heuristic approaches.

1. Introduction

Computing and implementing an optimal production schedule can reduce operational costs, increase profit margins, and avoid deviations from environmental constraints [1]. However, complex industrial plants can have multiple production, storage, and distribution subsystems, several distinct raw materials and intermediate and final products, and intricate connections between all these elements that make scheduling a difficult decision-making process.

Scheduling problems typically deal with four main decisions [1]: ① determining the required tasks to fulfill the corresponding objectives, requirements, and/or demand targets; ② assigning each task to a processing unit or resource that is available in the network; ③ defining the sequence in which the tasks will be executed; and ④ timing the tasks—that is, determining when to start and stop each one (Fig. 1). Optimal scheduling decisions are those that maximize or minimize a desired objective such as profit, total cost, lead time, and so forth. Scheduling software and tools based on mathematical programming is becoming more usual in practice.

The scheduling of gasoline-blending operations is an important and relevant industrial problem because gasoline accounts for 60%–70% of the total profit of an oil refinery [2–4]. In a gasoline-blending system, components from dedicated supply tanks are mixed in blending tanks or in-line blenders and sent to product tanks. Blending tanks can either have flow in or out (Fig. 2), resembling batch operation. In contrast, in-line blenders operate in a continuous manner (Fig. 3). In addition to the four decisions mentioned earlier, scheduling blend operations should also involve determining the blend recipes—that is, the amounts of components to mix such that products' quality properties meet given specifications.

The gasoline-blending system studied in this work is described in Fig. 3. Gasoline blending is carried out by one or more continuous

* Corresponding author.
E-mail address: mahalec@mcmaster.ca

http://dx.doi.org/10.1016/J.ENG.2017.02.005

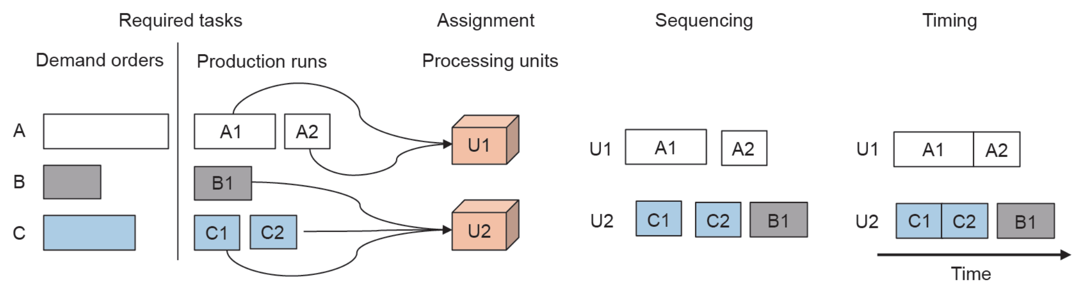

Fig. 1. Main scheduling decisions.

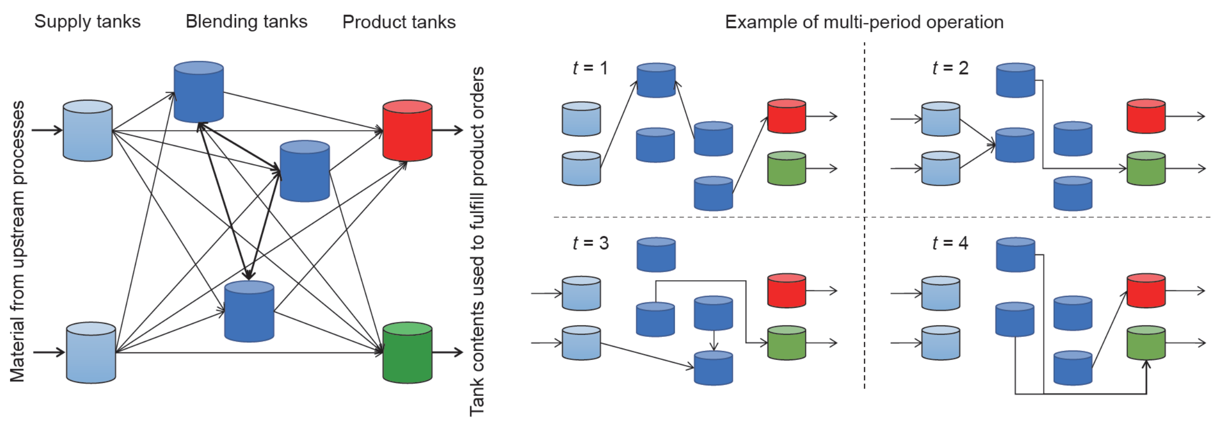
Supply tanks
Blending tanks
Product tanks
Material from upstream processes
Tank contents used to fulfill product orders
Example of multi-period operation
t = 1
t = 2
t = 3
t = 4

Fig. 2. Batch-blending system. The variable *t* means time period.

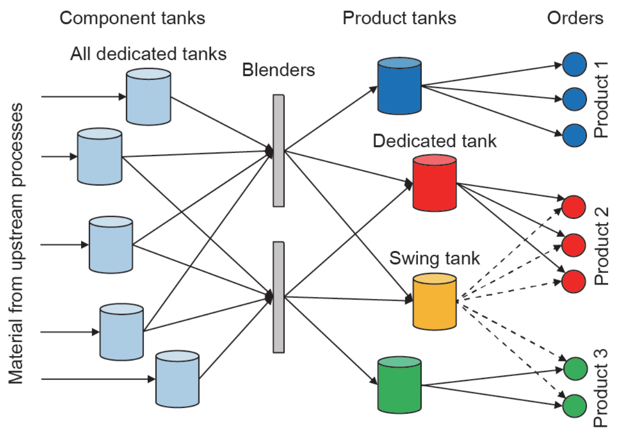

Fig. 3. General scheme of the continuous gasoline-blending system studied in this work.

blenders. Each blender is connected to the sources of blend components. Blended material in some refinery configurations goes to a storage tank, while in other configurations, it can go directly to the pipeline. Since there are several grades of gasoline produced (e.g., regular, medium, premium), the blender switches from blending one grade to another. Each switch requires (partial) realignment of blend feeding lines, which leads to a loss of blending capacity. In addition, switching to a different range of quality properties often requires resetting or recalibration of the analyzers used to measure them.

Some of the most important quality properties of gasoline are research octane number (RON), motor octane number (MON), Reid vapor pressure (RVP), density, sulfur, aromatics (Ar), and olefins (Ol) content. RON, MON, and RVP do not blend linearly; thus, considering nonlinear blending rules for such quality properties in the scheduling model can increase the accuracy of the solution and reduce quality giveaways [4,5].

Since the 1960s, there has been a significant effort to derive so-called "blend indices." These are nonlinear transformations of the actual quality properties of the blend components, which then, as a linear combination, can predict the quality of the blend. Even with this approach, two issues remain:

(1) If a product is blended into a tank, there is always some material in the tank (the so-called "tank heel") left over from previous blends. Any new blend must include the properties of the material in the tank in the calculation of the new blend recipe, which leads to a nonlinear blending model for multi-period scheduling, even when using blend indices.

(2) Blend properties calculated from these indices are not 100% accurate, and a cumulative annual quality giveaway of, for example, octane can amount to very large cost increments. This forces the use of nonlinear blending models for the calculation of, for example, octane numbers.

Mathematical models for scheduling problems are usually formulated as mixed-integer linear programming (MILP). However, for gasoline blending, nonlinear behavior is intrinsic to the corresponding process and mixed-integer nonlinear programming (MINLP) needs to be employed for the sake of accuracy. Most nonlinear terms are non-convex, making convex optimization techniques ineffective. A global optimization approach is thus required. Before describing the proposed global optimization method, a brief review of previous work is presented in the following paragraphs.

1.1. Literature review on refinery scheduling

Scheduling models can be divided into two main categories based on the treatment of the time domain: discrete- and continuous-time formulations. In discrete-time models, the time horizon is divided into several time periods of known duration with fixed start and end time. In continuous-time models, the time horizon is partitioned into time slots whose duration will be determined by the optimization. While continuous-time models generate problems with fewer discrete variables than their discrete counterparts, they are more complex to formulate and often feature many "big-M" constraints that, due to their weak relaxations [6], compromise computational performance. More in-depth reviews of scheduling formulations can be found in Refs. [1,7–9].

Gasoline blending has been studied by many researchers due to its commercial importance and non-convex features, which makes it a suitable subject for testing different formulations and algorithmic approaches. Operational constraints found in gasoline blending are related to the presence of multipurpose tanks and non-identical blenders, to different storage-tank policies (e.g., whether the simultaneous receipt and delivery of material is allowed or not), and to practical aspects such as minimum blend sizes and minimum blender running and setup times. Not all of these constraints are considered in published scheduling models. In some cases, blend recipes are assumed to be fixed (i.e., they cannot be optimized). The downstream distribution or shipping problem (i.e., timing delivery tasks to fulfill the demand) is sometimes also part of the blend-scheduling problem.

Méndez et al. [2] presented both a discrete- and continuous-time MILP model to schedule gasoline-blending operations. An iterative method was employed to handle nonlinear blending rules while preserving the linearity of the models. Several key operational constraints were omitted and the distribution problem was not considered.

Jia and Ierapetritou [10] developed a continuous-time MILP model to simultaneously schedule gasoline-blending tasks and distribution operations. The linearity of the model was maintained by using given preferred recipes. Their model was later extended to schedule operations of the main processing units in an oil refinery [11].

Glismann and Gruhn [12] used a two-level approach based on discrete-time models. Blend recipes and production targets were computed first using a nonlinear programming (NLP) model. Then a MILP model was employed to solve the short-term scheduling problem using such recipes and targets. The scheduling model was based on the resource-task-network (RTN) representation and did not consider multipurpose tanks or the delivery-scheduling problem.

Li et al. [13] formulated a continuous-time MILP model featuring a common time grid for all units (i.e., blenders and tanks). Li and Karimi [3] extended and improved this MILP model by using unit-specific time grids and including most of the operational constraints found in practice. Both models optimized blend recipes using blend indices. Based on these two previous works, Li et al. [4] presented a unit-specific continuous-time MINLP formulation where nonlinear terms arise from enforcing constant blending rates.

Castillo and Mahalec [14] developed a three-level decomposition algorithm through which recipe optimization can be done using linear and/or nonlinear blending rules. They considered the distribution problem, blend-size threshold constraints, parallel non-identical blenders, swing tanks, and product-dependent setup times. A discrete-time model was formulated for each level. The first level optimized the blend recipes, the second level approximated the production schedule, and the third level computed a detailed blend-and-delivery schedule. Due to the large size of the scheduling model at the third level for the entire horizon, it was solved in subintervals. Solutions computed by this approach were better and the execution times for large problems were two orders of magnitude shorter than those from previous methods [3,13]. In their subsequent work, Castillo and Mahalec [15] introduced a significantly modified version of the continuous-time scheduling model from Li and Karimi [3] (with a smaller number of binary variables [16]) for dealing with the third level. Case studies with nonlinear blend-scheduling problems were solved very close to global optimality with short execution times.

Lotero et al. [17] proposed another formulation of the multi-period pooling problem. They denominated this discrete-time MINLP formulation as a hybrid of a source-based model (similar to Castro's split-fraction model [18]) and a concentration-based model [19]. Redundant constraints were added to improve the linear relaxation, and the model was solved using a two-stage MILP-NLP approach. The MILP was a relaxation of the original MINLP model. The NLP model was obtained by fixing the integer variables of the original model to the values computed by the MILP. The algorithm adds integer, optimality, or feasibility cuts to the MILP model at each iteration, and stops when the difference between the MILP and NLP solutions is smaller than a pre-specified tolerance.

Cerdá et al. [20] presented a continuous-time MILP formulation that uses floating slots dynamically allocated to time periods while solving the problem. The model included most of the operational constraints found in practice. Cerdá et al. [21] then extended the model to handle nonlinear blending rules, thus formulating a continuous-time MINLP model. An approximate MILP formulation was derived by replacing the nonlinear blending rules with linear blend indices. The values of the binary variables computed by this MILP were fixed in the original MINLP, thus becoming an NLP that was solved to find a near-optimal solution of the original problem.

1.2. Literature review on global optimization

Global optimization of nonlinear non-convex problems has been a subject of extensive research over the last three decades. Even though powerful commercial solvers have been developed [22,23], there has been a continuous stream of advances in the field.

Global optimization algorithms have in common the generation of a convex relaxation of the problem, which provides a lower bound to the value of the objective function, and a way to generate feasible solutions (the upper bound). In addition, they have a method of bringing the lower and upper bounds together, so that the optimality gap can be reduced to ε-tolerance.

Computing a tight lower bound is absolutely critical. This may involve replacing the original formulation with an equivalent that preserves the feasible space but has a stronger relaxation (the different formulations for the pooling problem are a well-known example [24]). Another option is to reorganize the model constraints and add others that, although redundant in the original space, strengthen the relaxation. This procedure is known as the reformulation linearization technique [25]. The disadvantage is that there is no systematic way of knowing where to act in order to move toward a stronger relaxation.

A widely used method that guarantees convergence to the global optimal solution is known as spatial branch-and-bound [26,27]. It is an essential element of commercial solvers such as BARON [22], ANTIGONE [23], Couenne [28], and SCIP [29]. Spatial branch-and-bound works by iteratively reducing the domain of the variables, one by one, which in turn improves the quality of the convex relaxation. Note that if the initial relaxation is weak, due to the presence of many non-convex terms, convergence can be rather slow. It is thus important to have good branching strategies and bound-tightening techniques. Optimality-based bound tightening (OBBT) is an example of the latter. Although typically applied only at the root node or up to a limited depth [28], recent results have shown that applying OBBT in every node may lead to considerably lower optimality gaps [30]. OBBT involves solving one minimization and one maximization problem for each variable (appearing in non-convex terms) in order to generate tighter lower and upper bounds. It can be solved sequentially—which has the advantage of generating tighter variable bounds and the disadvantage of being computationally expensive when dealing with a large number of variables—or in parallel [31].

Bilinear terms are a common source of non-convexities in process systems engineering. They can be relaxed using the McCormick envelopes, considering either the full domain of the variables [32] or a reduced domain following partitioning [33,34]. Simultaneous domain partitioning involves adding a new set of binary variables to the problem and guarantees global optimality in the limit of an infinite number of partitions. Spatial branch-and-bound can thus be avoided. One critical aspect concerns the scaling of problem size with the number of partitions. Earlier piecewise relaxation techniques [33] scale linearly, while recent formulations scale logarithmically. Examples of the latter are described next.

Misener et al. [35] developed a global optimization algorithm for the standard pooling problem and concluded that the logarithmic scheme is more advantageous for more than eight partitions. Kolodziej et al. [19] proposed a MINLP formulation for the multi-period pooling problem, in which nonlinearities arise from using the dynamic inventories of tanks as blend components. They employed a radix-based discretization technique that partitions one variable in every bilinear term to obtain a MILP relaxation. This discretization technique is known as multiparametric disaggregation [36]. Castro [37] developed the normalized multiparametric disaggregation technique (NMDT) [36], which works by discretizing the range between a variable's lower and upper bounds (belonging to [0, 1]). The advantage is that the number of partitions becomes the same for every discretized variable, even if their domain is different (when using a global discretization level parameter). NMDT has been successfully used to solve multi-period blending problems to global optimality, both as a stand-alone approach [18,38,39] or integrated in a spatial branch-and-bound algorithm [30].

Overall, contributions from cited works have enabled optimal solutions of gasoline blend-scheduling problems up to a certain model size. However, when dealing with large-scale problems, the computation and validation of global optimal solutions remain a difficult challenge.

1.3. Contributions of this work

This work presents a novel deterministic global optimization algorithm to solve non-convex MINLP or NLP models with nonlinearities that are strictly due to bilinear and/or quadratic terms. The main features of this algorithm are:

- The use of different linear and piecewise linear relaxation techniques to derive convex relaxations of the original non-convex model;
- The collection of different feasible solutions from the convex relaxation, which provide starting points for a local nonlinear solver to find feasible solutions of the original model;
- A dynamic increase in the number of partitions for piecewise linear relaxations;
- The reduction of the domain of the variables involved in nonlinear terms by means of an OBBT method; and
- The parallelization of the steps regarding computation of feasible solutions and the OBBT method.

The algorithm is tested on different gasoline blend-scheduling examples. For this class of problems, the results show that the proposed algorithm is on par with or better than two leading commercial global solvers.

The rest of this paper is organized as follows: Section 2 describes the problem statement and the assumptions made. Section 3 reviews the scheduling model employed in this work and presents the nonlinear equations used for octane blending. Section 4 briefly explains the piecewise linear relaxations employed to compute the estimates of the global solution. Section 5 describes the OBBT method to reduce the domain of the variables involved in nonlinear terms. Section 6 presents the steps of the global optimization algorithm. Section 7 contains the data describing the test examples. Section 8 shows the results obtained with the proposed algorithm and provides a comparison with other methods. The paper ends with conclusions in Section 9.

2. Problem statement

Given a blending system (i.e., storage tanks, blenders, and their interconnections; see Fig. 3), a scheduling horizon, a set of blend components and their corresponding supply and quality profiles along the horizon, a set of products and their minimum and maximum quality property specifications, a set of delivery orders for each product, and the initial inventory levels, it is required to determine the blend recipes, the production and delivery sequences, and the inventory profiles of all tanks, 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 (i.e., late deliveries).

The following constraints are considered:

(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.
(6) Product tanks cannot receive and deliver material at the same time.

The assumptions made in this work are:

(1) The flowrate profile of each component from the upstream process is piecewise constant.
(2) The 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 swing tanks can change their product service (i.e., change from storing one product to storing 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 (any original order involving different products can be disaggregated into orders of a single product).
(9) All orders are fulfilled during the scheduling horizon.

In summary, this problem considers the scheduling of blending and delivery operations, recipe determination, and product allocation of swing tanks along the scheduling horizon.

3. Gasoline blend-scheduling model

The scheduling model employed in this work is the one presented by Castillo and Mahalec [16]. It employs a continuous-time formulation, considers nonlinear blending equations, and does not allow simultaneous receipt and delivery by product tanks. This is a non-convex MINLP model, and it will be denoted as model **P** (or problem **P**). The main features of the scheduling model are highlighted in this section.

The scheduling model uses unit-specific time slots of varying length to determine when a specific task needs to be executed in each unit (blenders and tanks in this case). We assign a sufficiently high number of time slots, which will likely be higher than the number of slots required for blending each grade. This ensures that there are sufficient degrees of freedom (enough available switches) to meet the varying product-delivery schedule.

The start time of a unit slot is equal to the end time of the previous one. The first unit slot starts at the beginning of the scheduling horizon, and the end of the last unit slot matches exactly the end of the horizon. Blending tasks begin at the start of a time slot, but can finish before its end. Delivery tasks from product tanks can start and finish within the corresponding slot. It is assumed that component tanks are continuously receiving material at some specified rate (i.e., the supply profile). Time periods are used to delineate the points where changes occur in the supply rates and/or quality of blend components. Time slots are assigned to these time periods. A time slot must end within its assigned period. However, for component tanks, the last time slot of a period must end exactly at that period's boundary (in order to properly respect the changes in supply rates and/or quality of blend components). Fig. 4 shows a graphical representation of these unit-specific time slots for a blending system with two blend component tanks (CT1, CT2), one blender (B1), and two product tanks (PT1, PT2). Unit slots 1 and 2 are pre-assigned to period 1, while slots 3 and 4 are pre-assigned to period 2. Note that the optimization has determined that slot 3 in the CT2 grid, and slot 4 in the PT1 grid, have zero length.

The objective of the scheduling model is to minimize the blend cost (i.e., materials cost), the switching cost associated with each blend run, product changeovers in the swing tanks, the number of "delivery runs" (i.e., the number of time slots used to deliver a specific order from a given tank), and the demurrage cost. Delivery runs are penalized in order to avoid computing delivery schedules that deliver the same order from several tanks at the same time, and to minimize intermittent deliveries of the same order from the same tank.

Binary variables are employed in the model to determine, at each time slot, the following discrete decisions:

- Which product tank each blender is feeding (one variable for each blender-tank connection);
- What gasoline grade is stored in each product tank (one variable for each grade-tank pair); and
- What demand order each product tank is partially or completely fulfilling (one variable for each tank-order connection).

With these binary variables, other discrete decisions can be modeled with 0–1 continuous variables, such as:

- What gasoline grade each blender is producing;
- The status of a blender (running or idle);
- The transition of a blender from running to being idle, or vice versa;
- When a new blend run starts;
- Product transitions in the blenders; and
- Product changeovers in the swing tanks.

The scheduling model also considers variable blending rates,

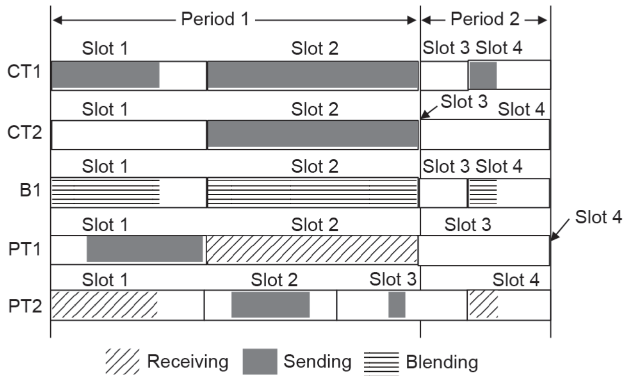

Fig. 4. Representation of unit-specific time slots employed in the scheduling model.

variable delivery rates, blender- and product-specific setup times for the blenders (i.e., idle times for, e.g., cleaning or sensor recalibration purposes), maximum delivery rates from blend component tanks to the blenders, minimum blend size, and minimum running times for each blender and product. Other constraints include the material balances, product composition specifications, product quality specifications, and linear and/or nonlinear blending equations.

The difficulty in solving this scheduling model to global optimality arises from the following factors:

- The significant number of discrete decisions that can be made, which are directly related to the number of time slots, gasoline grades, blenders, product tanks, and demand orders (the combinatorial nature of the problem);
- The inclusion of nonlinear blending equations (the non-convex nature of the problem); and
- All the considered operational constraints.

Castillo and Mahalec [16] found that introducing constraints regarding minimum blend cost and minimum switching cost can improve the quality of the solution and reduce the execution times for small- to medium-size problems. The minimum blend cost is computed using the approach delineated in Castillo et al. [40].

The nonlinear blending equations are presented next, since they were rewritten in such a way that nonlinear terms are only bilinear or quadratic.

Nonlinear blending equations

Eq. (1) to Eq. (19) are the proposed reformulation of the ethyl RT-70 model for octane blending [5,41]. Bilinear terms appear in Eqs. (1), (13), (14), and (18). Quadratic terms appear in Eqs. (15), (16), (17), and (19). Main sets, subscripts, variables, and parameters are described next. Set $\mathbf{I} = \{i\}$ consists of the blend components, $\mathbf{BL} = \{bl\}$ is constituted by the blenders, $\mathbf{N1} = \{n\}$ is the time slots, and set $\mathbf{QN} = \{(\theta, n)\}$ represents the time slots associated with each quality profile θ. Variable $V_{\text{comp}}(i, bl, n)$ indicates the volume of blend component i to blender bl during slot n. Variable $V_{\text{blend}}(bl, n)$ is the volume being processed by blender bl during slot n. The volume fraction of component i going into blender bl during slot n is denoted by variable $r(i, bl, n)$. Parameter $Q_{\text{bc}}(i, e, \theta)$ represents the value of quality property e for blend component i and quality profile θ. $sens(i, \theta)$ is a parameter known as the octane number sensitivity; it is the difference between the octane numbers, that is, RON – MON, of blend component i and quality profile θ. The values for the ethyl RT-70 model coefficients are taken from Singh et al. [5] and are as follows: $a_1 = 0.03224$, $a_2 = 0.00101$, $a_3 = 0$, $a_4 = 0.0445$, $a_5 = 0.00081$, and $a_6 = -0.0645 \times 10^{-4}$.

$$V_{\text{comp}}(i, bl, n) = r(i, bl, n) V_{\text{blend}}(bl, n) \quad \forall i, bl, n \in \mathbf{N1} \tag{1}$$

$$r_{\text{avg}}^{\text{RON}}(bl, n) = \sum_i r(i, bl, n) Q_{\text{bc}}(i, e, \theta) \quad \forall e = \text{"RON"}, bl, n \in \mathbf{N1}, \theta : (\theta, n) \in \mathbf{QN} \tag{2}$$

$$r_{\text{avg}}^{\text{MON}}(bl, n) = \sum_i r(i, bl, n) Q_{\text{bc}}(i, e, \theta) \quad \forall e = \text{"MON"}, bl, n \in \mathbf{N1}, \theta : (\theta, n) \in \mathbf{QN} \tag{3}$$

$$Ol_{\text{avg}}(bl, n) = \sum_i r(i, bl, n) Q_{\text{bc}}(i, e, \theta) \quad \forall e = \text{"Ol"}, bl, n \in \mathbf{N1}, \theta : (\theta, n) \in \mathbf{QN} \tag{4}$$

$$Ar_{\text{avg}}(bl, n) = \sum_i r(i, bl, n) Q_{\text{bc}}(i, e, \theta) \quad \forall e = \text{"Ar"}, bl, n \in \mathbf{N1}, \theta : (\theta, n) \in \mathbf{QN} \tag{5}$$

$$Ol_{\text{avg}}^{\text{sq}}(bl, n) = \sum_i r(i, bl, n) \left[Q_{\text{bc}}(i, e, \theta)^2 \right] \quad \forall e = \text{"Ol"}, bl, n \in \mathbf{N1}, \theta : (\theta, n) \in \mathbf{QN} \tag{6}$$

$$Ar_{\text{avg}}^{\text{sq}}(bl, n) = \sum_i r(i, bl, n) \left[Q_{\text{bc}}(i, e, \theta)^2 \right] \quad \forall e = \text{"Ar"}, bl, n \in \mathbf{N1}, \theta : (\theta, n) \in \mathbf{QN} \tag{7}$$

$$sens_{\text{avg}}(bl,n)=\sum_i r(i,bl,n)sens(i,\theta)\quad \forall i,bl,n\in \mathbf{N1} \tag{8}$$

$$sens_{\text{avg}}^{\text{RON}}(bl,n)=\sum_i r(i,bl,n)Q_{\text{bc}}(i,e,\theta)sens(i,\theta)\\ \forall e=\text{“RON”}, bl,n\in \mathbf{N1}, \theta:(\theta,n)\in \mathbf{QN} \tag{9}$$

$$sens_{\text{avg}}^{\text{MON}}(bl,n)=\sum_i r(i,bl,n)Q_{\text{bc}}(i,e,\theta)sens(i,\theta)\\ \forall e=\text{“MON”}, bl,n\in \mathbf{N1}, \theta:(\theta,n)\in \mathbf{QN} \tag{10}$$

$$\begin{aligned}Q_{\text{pr}}(bl,e,n)=&r_{\text{avg}}^{\text{RON}}(bl,n)+a_1\left[sens_{\text{avg}}^{\text{RON}}(bl,n)-rs_{\text{avg}}^{\text{RON}}(bl,n)\right]\\ &+a_2\left[Ol_{\text{avg}}^{\text{sq}}(bl,n)-Ol2_{\text{avg}}(bl,n)\right]\\ &+a_3\left[Ar2_{\text{avg}}^{\text{sq}}(bl,n)-2\cdot Ar3_{\text{avg}}(bl,n)+Ar4_{\text{avg}}(bl,n)\right]\\ &\forall e=\text{“RON”}, bl,n\in\mathbf{N1}\end{aligned} \tag{11}$$

$$\begin{aligned}Q_{\text{pr}}(bl,e,n)=&r_{\text{avg}}^{\text{MON}}(bl,n)+a_4\left[sens_{\text{avg}}^{\text{MON}}(bl,n)-rs_{\text{avg}}^{\text{MON}}(bl,n)\right]\\ &+a_5\left[Ol_{\text{avg}}^{\text{sq}}(bl,n)-Ol2_{\text{avg}}(bl,n)\right]\\ &+a_6\left[Ar2_{\text{avg}}^{\text{sq}}(bl,n)-2\cdot Ar3_{\text{avg}}(bl,n)+Ar4_{\text{avg}}(bl,n)\right]\\ &\forall e=\text{“MON”}, bl,n\in\mathbf{N1}\end{aligned} \tag{12}$$

$$rs_{\text{avg}}^{\text{RON}}(bl,n)=r_{\text{avg}}^{\text{RON}}(bl,n)sens_{\text{avg}}(bl,n)\quad\forall bl,n\in\mathbf{N1} \tag{13}$$

$$rs_{\text{avg}}^{\text{MON}}(bl,n)=r_{\text{avg}}^{\text{MON}}(bl,n)sens_{\text{avg}}(bl,n)\quad\forall bl,n\in\mathbf{N1} \tag{14}$$

$$Ar2_{\text{avg}}(bl,n)=Ar_{\text{avg}}(bl,n)^2\quad\forall bl,n\in\mathbf{N1} \tag{15}$$

$$Ol2_{\text{avg}}(bl,n)=Ol_{\text{avg}}(bl,n)^2\quad\forall bl,n\in\mathbf{N1} \tag{16}$$

$$Ar2_{\text{avg}}^{\text{sq}}(bl,n)=Ar_{\text{avg}}^{\text{sq}}(bl,n)^2\quad\forall bl,n\in\mathbf{N1} \tag{17}$$

$$Ar3_{\text{avg}}(bl,n)=Ar_{\text{avg}}^{\text{sq}}(bl,n)Ar2_{\text{avg}}(bl,n)\quad\forall bl,n\in\mathbf{N1} \tag{18}$$

$$Ar4_{\text{avg}}(bl,n)=Ar2_{\text{avg}}(bl,n)^2\quad\forall bl,n\in\mathbf{N1} \tag{19}$$

4. Piecewise linear relaxations

As mentioned in Section 1, the use of piecewise linear relaxations is becoming more widespread due to the maturity of MILP solvers. Piecewise McCormick relaxation (PMCR) and the NMDT will be employed in this work. These techniques replace each bilinear term in model **P** with a single variable, thus linearizing the corresponding equations. This single variable is then subject to various linear constraints, which add extra continuous and binary variables to the model. If equal to 1, these extra binary variables activate a specific interval of the domain (i.e., partition) of one of the variables in the bilinear term (denoted as the discretized variable). The number of partitions is denoted as *NP*, and it is assumed that all discretized variables have the same number of partitions. PMCR has a linear relation between *NP* and the number of extra binary variables required per discretized variable, while NMDT exhibits a logarithmic relation. For a more detailed explanation of these methods, the reader is encouraged to review Refs. [16,37].

The resulting MILP model is denoted as model **PR** and is a relaxation of problem **P**. This means that the optimal solution of model **PR** is a valid estimate of the global solution of **P** (in the minimization case, this will be a lower bound, *LB*). Moreover, an estimate of the best possible solution of model **PR** is a valid estimate of the global optimum of **P**. Therefore, even if model **PR** is not solved to optimality by a MILP solver within a given allocated time, a new estimate of the global solution can still be found. The larger the number of partitions, the closer model **PR** is to model **P**; see Fig. 5 for an illustration with an example involving a single discretized variable.

If the relaxation is tight, then its optimal solution will be very close to the original optimum. Hence, a strategy to find a feasible solution to the original problem **P** (in the minimization case, this will be an upper bound, *UB*) is to initialize **P** with the optimal solution of model **PR**. Since some MILP solvers, such as CPLEX, can store multiple feasible solutions to the MILP problem, potentially leading to different solutions of **P** due to the different starting points, we use a multi-start strategy in parallel fashion. Note that, for practical reasons related to the speed and robustness of commercial solvers, it is more convenient to solve NLP models instead of MINLPs. This is the reason why the values of the binary variables are fixed, converting problem **P** (MINLP) into **PF** (NLP). The compact notations of models **P**, **PR**, and **PF** are as follows:

Model **P**:

$$\begin{aligned}&\min f_0(x,y)\\ \text{s.t.}\quad &f_m(x,y)\le 0\quad \forall m\in\mathbf{M}/\{0\}\\ &f_m(x,y)=\sum_{(i,j)\in\mathbf{BLT}}a_{ijm}x_ix_j+B_mx+C_my+d_m\quad\forall m\in\mathbf{M}\\ &0\le x^{\text{L}}\le x\le x^{\text{U}}\\ &x\in\mathbb{R}^{lx},\ y\in\{0,1\}^{ly}\end{aligned}$$

Model **PR**:

$$\begin{aligned}&\min f_0^{\text{R}}(x,y)\\ \text{s.t.}\quad &f_m^{\text{R}}(x,y)\le 0\quad \forall m\in\mathbf{M}/\{0\}\\ &f_m^{\text{R}}(x,y)=\sum_{(i,j)\in\mathbf{BLT}}a_{ijm}w_{ij}+B_mx+C_my+d_m\quad\forall m\in\mathbf{M}\end{aligned}$$

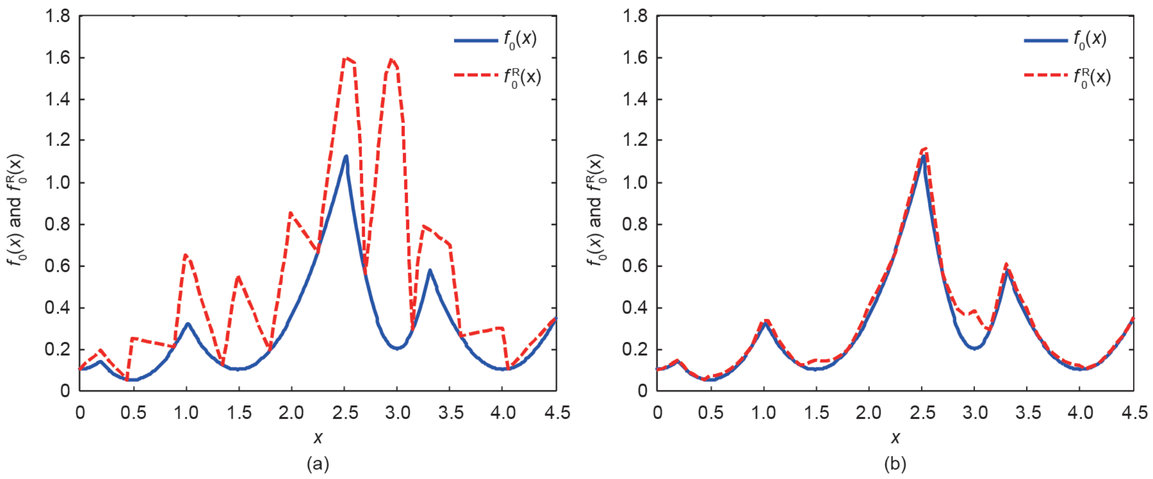

Fig. 5. Accuracy of the relaxation (f_0^{R}) with respect to exact representation (f_0) of the boundaries of a feasible region increases with the number of partitions (maximization problem). (a) 10 partitions; (b) 100 partitions.

$$g_n^{\mathrm{R}}(x, w, v, z) \le 0 \quad \forall n \in \mathbf{N}$$
$$g_n^{\mathrm{R}}(x, w, v, z) = H_n'x + A_n'w + B_n'v + C_n'z + d_n' \quad \forall n \in \mathbf{N}$$
$$0 \le x^{\mathrm{L}} \le x \le x^{\mathrm{U}}$$

Model **PF**:

$$\min f_0^{\mathrm{F}}(x)$$
$$\text{s.t.} \quad f_m^{\mathrm{F}}(x) \le 0 \quad \forall m \in \mathbf{M}/\{0\}$$
$$f_m^{\mathrm{F}}(x) = \sum_{(i,j)\in\mathbf{BLT}} a_{ijm} x_i x_j + B_m x + C_m \hat{y} + d_m \quad \forall m \in \mathbf{M}$$
$$0 \le x^{\mathrm{L}} \le x \le x^{\mathrm{U}}$$
$$x \in \mathbb{R}^{lx}$$

Note that, in this section, set **M** = {m} represents all the original constraints, set **N** = {n} represents all the constraints required by the piecewise linear relaxation technique, and set **BLT** = {(i, j)} represents all the bilinear terms. Variables x and y are the original continuous and binary variables, respectively, and v and z are the extra continuous and binary variables, respectively, required by the relaxation strategy. Variable w_{ij} is the continuous variable that replaces the bilinear term $x_i x_j$. Scalars lx, ly, lw, lv, and lz represent the size of vectors x, y, w, v, and z, respectively. Parameters x^{L} and x^{U} are respectively the lower and upper bounds of the x variables. Note that quadratic terms can be treated as bilinear terms.

5. Tightening bounds on the variables

Model **PR** becomes tighter (i.e., closer to model **P**) as the number of partitions of the discretized variables is increased. However, an increase in the number of partitions produces an increment in the size of model **PR** and, after a certain number of partitions, model **PR** can become computationally intractable. Therefore, another technique is required in order to avoid the necessity of a large number of partitions. In this work, an OBBT method is employed [34,42]. The idea is to reduce the domain of the variables involved in nonlinear terms by computing new bounds of these variables by solving two optimization problems (a maximization problem and a minimization problem per variable). This is done after a new and better feasible solution to **P** is computed. After reducing the domain of the variables, model **PR** becomes closer to **P** without increasing the number of partitions, as shown in Fig. 6.

The mathematical model used in OBBT is denoted as model **PRB**, which is constructed as a relaxation of **P**, but with a different objective function and an extra constraint. To compute a lower bound of variable x_{h}, that is, $x_{\mathrm{h}}^{\mathrm{L}}$, the objective function is to minimize this variable. To compute an upper bound of variable x_{h}, that is, $x_{\mathrm{h}}^{\mathrm{U}}$, the objective function is to maximize this variable. In order to compute new bounds, the extra constraint added imposes the condition that the value of the relaxed version of the original objective function, that is, $f_0^{\mathrm{R}}(x, y)$, must be at least as good as the current best feasible solution.

Note that models **PR** and **PRB** can use different relaxations. In this work, model **PRB** employs standard McCormick envelopes [32] and integrality requirements on variables y are dropped, thus reducing **PRB** to linear programming (LP). The lower and upper bounds of variable x_{h} are updated with the optimal solutions of the corresponding LP model. Compact notation of model **PRB** is shown below for a minimization problem.

Model **PRB**:

$$x_{\mathrm{h}}^{\mathrm{L}} = \min x_{\mathrm{h}} \quad \left(x_{\mathrm{h}}^{\mathrm{U}} = \max x_{\mathrm{h}}\right)$$
$$\text{s.t.} \quad f_0^{\mathrm{R}}(x, y) \le UB$$
$$f_m^{\mathrm{R}}(x, y) \le 0 \quad \forall m \in \mathbf{M}/\{0\}$$
$$f_m^{\mathrm{R}}(x, y) = \sum_{(i,j)\in\mathbf{BLT}} a_{ijm} w_{ij} + B_m x + C_m y + d_m \quad \forall m \in \mathbf{M}$$
$$g_k^{\mathrm{RB}}(x, w) \le 0 \quad \forall k \in \mathbf{K}$$
$$g_k^{\mathrm{RB}}(x, w) = H_k'x + A_k'w + d_k' \quad \forall k \in \mathbf{K}$$
$$0 \le x^{\mathrm{L}} \le x \le x^{\mathrm{U}}$$
$$x \in \mathbb{R}^{lx}, \; y \in [0, 1]^{ly}, \; w \in \mathbb{R}^{lw}$$

OBBT consists of solving these LP models for all the variables involved in nonlinear terms, in a parallel framework, to reduce execution times. Thus, the bounds will generally be weaker than when solving the problems sequentially. Since the number of instances to solve can be very large, instances are solved in different blocks. These blocks are defined by a maximum number of problems to be solved in parallel. After one block is solved, the corresponding bounds are updated and then the next block is solved. Fig. 7 shows the flowchart of the OBBT method. Note that OBBT is applied only once per variable.

6. Global optimization algorithm

The steps of the proposed global optimization algorithm are presented next for a minimization problem. Fig. 8 shows the corresponding flowchart. Note that the algorithm can be applied to any MINLP problem with nonlinearities exclusive to bilinear or quadratic terms.

(1) Initialize algorithm parameters. Define the number of partitions to be used {NP_1, NP_2, …, NP_{last}} and set $NP = NP_1$. Set the

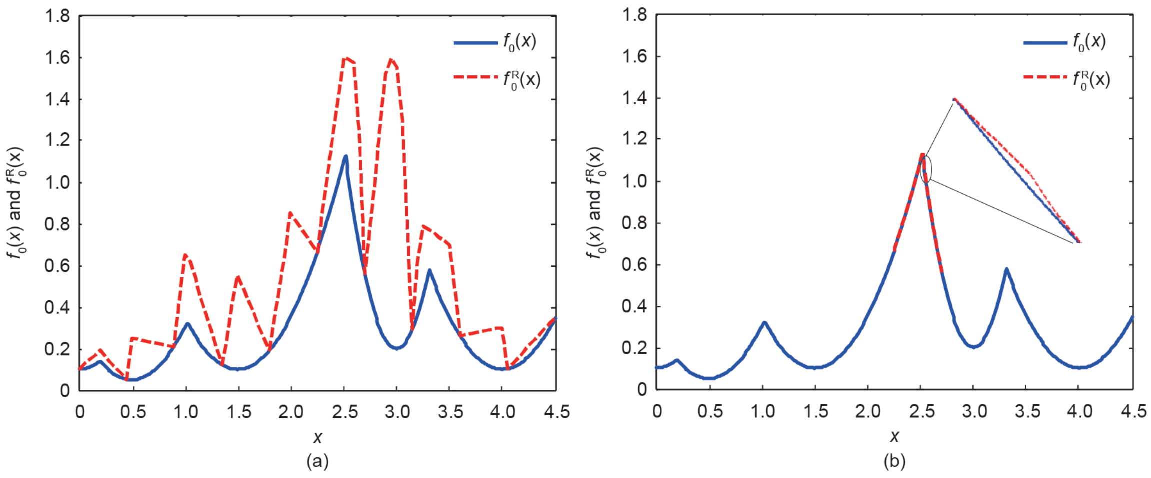

Fig. 6. Accuracy of the relaxation increases when the domain of the variables involved in nonlinear terms is reduced. (a) 10 partitions with $x \in [0, 4.5]$; (b) 10 partitions with $x \in [2.25, 2.7]$.

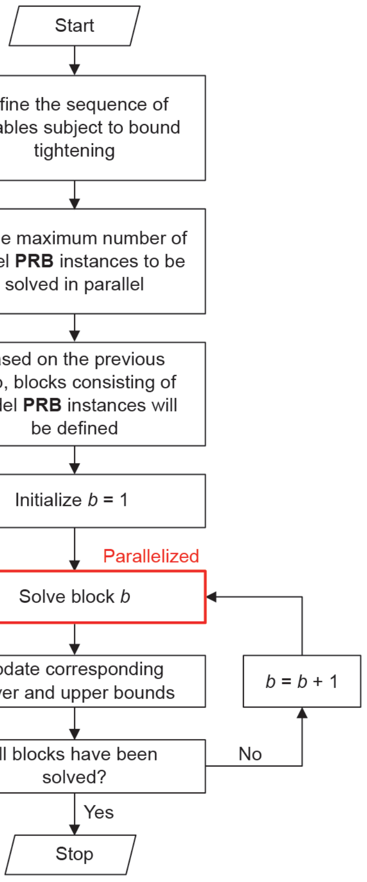
Start
Define the sequence of variables subject to bound tightening
Define maximum number of model PRB instances to be solved in parallel
Based on the previous step, blocks consisting of model PRB instances will be defined
Initialize b = 1
Parallelized
Solve block b
Update corresponding lower and upper bounds
b = b + 1
All blocks have been solved?
No
Yes
Stop

Fig. 7. Flowchart of the OBBT method.

lower bound $LB = -\infty$, upper bound $UB = +\infty$, total number of iterations counter $IT_{\text{total}} = 1$, iterations with the same number of partitions $IT_{\text{sameNP}} = 1$, maximum number of total iterations $IT_{\text{total}}^{\max}$, maximum number of iterations with the same number of partitions $IT_{\text{sameNP}}^{\max}$, maximum total time $TIME_{\text{total}}^{\max}$, and minimum relative tolerance ε.

(2) Lower bound computation. Solve MILP model **PR** using the CPLEX solver with parallel and solution pool options active. Update LB with the best possible solution from CPLEX, if this value is greater than the previous LB.

(3) Upper bound computation. Use the solutions stored in the CPLEX solution pool as starting points for NLP model **PF**. Solve NLP model **PF** instances in parallel using a local nonlinear solver. Update UB if any of the computed solutions is feasible and has a smaller objective function value than the previous UB.

(4) Update optimality gap. The following formula is used in this step: $OptGap = (UB - LB)/LB \times 100$.

(5) Check termination criteria. Stop if $OptGap \le \varepsilon$, if $IT_{\text{total}} = IT_{\text{total}}^{\max}$, if the total execution time is equal to or greater than $TIME_{\text{total}}^{\max}$, or if the number of partitions has already reached NP_{last}. Otherwise, continue to Step 6.

(6) If the upper bound UB did not improve in Step 3, or if $IT_{\text{sameNP}} = IT_{\text{sameNP}}^{\max}$, continue to Step 7. Otherwise, reduce the domain of the variables involved in nonlinear terms using the OBBT method described in Section 5; set $IT_{\text{total}} = IT_{\text{total}} + 1$ and $IT_{\text{sameNP}} = IT_{\text{sameNP}} + 1$, and then go back to Step 2.

(7) Increase the number of partitions to the next specified value. Set $IT_{\text{total}} = IT_{\text{total}} + 1$ and go back to Step 2.

Although the main elements of the algorithm have already been proposed (e.g., PMCR, NMDT, OBBT), the novelty is related to the way in which they are implemented. More specifically: ① the CPLEX solution

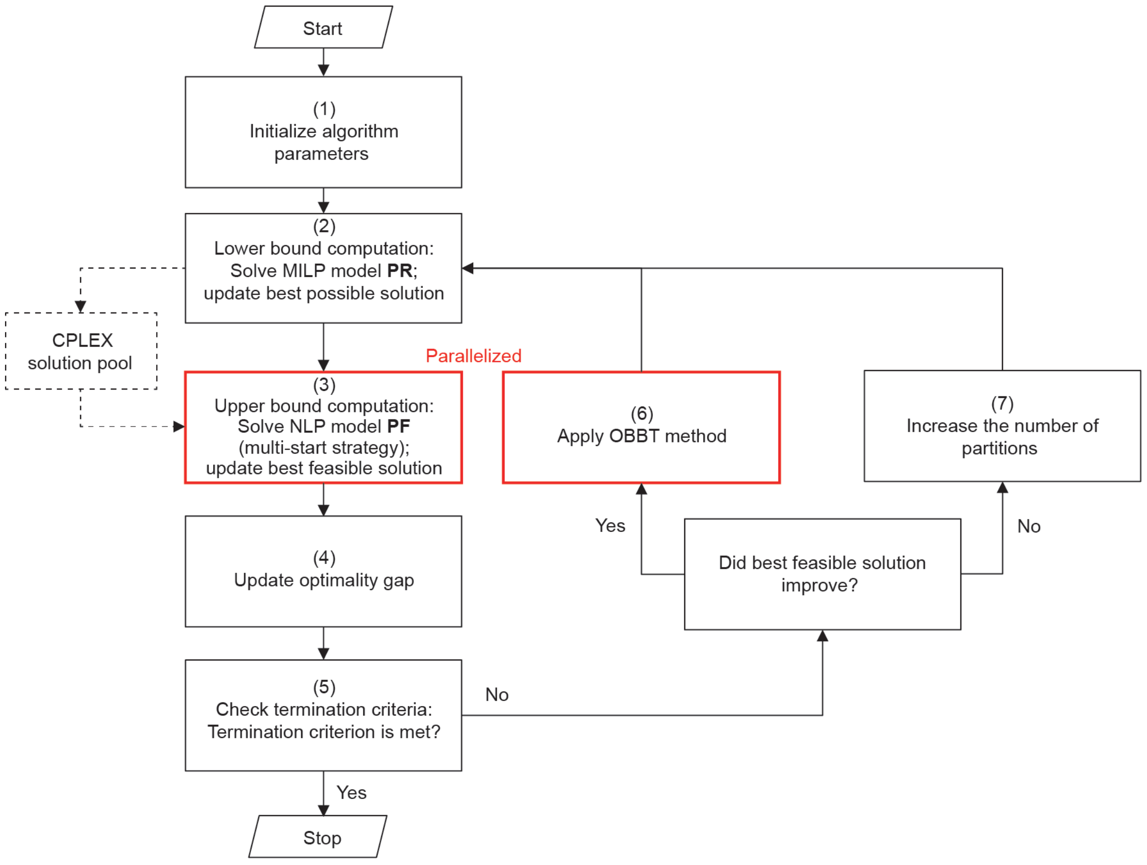
Start
(1) Initialize algorithm parameters
(2) Lower bound computation: Solve MILP model PR; update best possible solution
CPLEX solution pool
Parallelized
(3) Upper bound computation: Solve NLP model PF (multi-start strategy); update best feasible solution
(6) Apply OBBT method
(7) Increase the number of partitions
Yes
No
Did best feasible solution improve?
(4) Update optimality gap
(5) Check termination criteria: Termination criterion is met?
No
Yes
Stop

Fig. 8. Flowchart of the global optimization algorithm.

pool is used to store starting points for model **PF**, ② instances of model **PF** are solved in parallel, ③ OBBT is applied to blocks of variables and in a parallel framework, and ④ no branching strategy is employed.

7. Case studies

The tests in this paper consist of Examples 4, 8, 12, and 14 from Li and Karimi [3]. The difference in this work is that the ethyl RT-70 models are considered for blending RON and MON properties (as described in Section 3.1) instead of blend indices. RON index correlations from Li et al. [13], shown in Eq. (20) and Eq. (21), were used to compute the actual RON values from the blend indices given by Li and Karimi [3]. RBN denotes the research octane number blend index. MON values were assumed in this work and the corresponding minimum product specifications were set equal to zero; therefore, MON constraints will not be active at the optimal solution. Quality properties of blend components are assumed to be known (recall Section 2); therefore, Eq. (20) and Eq. (21) are only used to convert the blend components' RBN values to RON values before the optimization runs (i.e., they do not appear in the optimization problems).

$$RBN = RON + 11.5 \quad 0 \le RON \le 85 \tag{20}$$

$$RBN = \exp\left(0.0135RON + 3.422042\right) \quad RON > 85 \tag{21}$$

Table 1 describes the size of the blending system examples. Information about the periods, their duration, their corresponding time slots, and the orders that can be delivered within such periods is presented in Table 2. RON and MON values and their respective specifications are shown in Table 3. Table 4 presents the statistics regarding the size of model **P** when not using the constraints on the minimum blend and switching costs. When using such constraints, four equations are added to the model (minimum blend cost, minimum number of delivery runs, minimum number of blend runs, and minimum number of product changeovers in the swing tanks). Note that the size of the blending system and its corresponding scheduling model increase from Example 4 to Example 14.

Table 1
Summary of the blending system examples.

Example ID	Number of blenders	Number of orders	Number of products	Number of product tanks	Number of quality properties
4	1	15	4	11	9
8	2	20	4	11	9
12	2	35	5	11	9
14	3	45	5	11	9

Table 2
Periods, duration, time slots, and orders that can be delivered in each period.

Example ID	Period	Duration (h)	Slots	Orders that can be delivered
4	1	100	1, 2	O1–O7, O12–O15
	2	92	3, 4	O8–O11
8	1	80	1, 2	O1–O7, O12–O19
	2	70	3, 4	O8
	3	42	5, 6	O8–O11, O20
12	1	50	1–3	O1–O7, O12, O13, O15, O19, O33
	2	50	4–6	O14–O18, O27, O28, O33
	3	50	7–9	O8, O21, O24, O29–O32, O34, O35
	4	42	10–12	O8–O11, O20, O22, O23, O25, O26
14	1	50	1–3	O1–O7, O12, O13, O15, O19, O26
	2	50	4–7	O14–O18, O26
	3	50	8–10	O8, O21, O24, O27–O31, O45
	4	42	11–13	O8–O11, O20, O22, O23, O25, O32–O44

Table 3
RON and MON values and specifications.

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

Table 4
Statistics of model **P**.

Example ID	Number of equations	Number of variables	Number of binary variables	Number of bilinear terms
4	6 207	2 503	433	168
8	9 297	3 323	553	336
12	23 087	8 170	1 317	672
14	32 574	10 693	1 628	1 092

8. Results

All the examples were solved on a computing machine Intel® Core™ i7-4710HQ central processing unit (CPU), 2.50 GHz, 12 GB random-access memory (RAM), Windows 10 (8-core). The global optimization algorithm was implemented in Python 2.7. The Python script generates general algebraic modeling system (GAMS) files with the corresponding mathematical models, which are then solved by employing the GAMS-Python application program interface (API). The selected solvers were CPLEX 12.6 for model **PR** and model **PRB**, and CONOPT 3 for model **PF**.

The global optimization algorithm termination criteria were as follows: 0.01% optimality gap or 3600 s (1 h). There was no limit on the total number of iterations, nor on the iterations with the same number of partitions. The numbers of partitions in model **PR** when using PMCR were {2, 4, 8, 16, 32}, and when using NMDT were {10, 100, 1000}.

The termination criteria for the MILP problems (instances of model **PR**) were: an optimality gap of 0.01% or 600 s. The CPLEX parallel option was active (in deterministic mode) with a maximum number of threads equal to 8. In addition, the CPLEX solution pool option was active with a maximum pool capacity of 30 and the replacement option that generates diverse solutions. Thus, a maximum of 30 instances of model **PF** was solved per iteration using the GAMS parallel computing grid facility. CONOPT 3 default termination criteria were used.

The termination criteria for the LP problems (instances of model **PRB**) were: optimality or 60 s. A maximum number of 100 LP problems were solved in parallel using the GAMS parallel computing grid facility.

For comparison purposes, the global commercial solvers BARON 15.9 [33] and ANTIGONE 1.1 [34] were employed to solve the original model **P** using the same termination criteria as the proposed global optimization algorithm.

Section 8.1 presents the results obtained when not including the constraints on the minimum blend cost and minimum switching cost, while Section 8.2 shows the results when such constraints are added to the model. A comparison with previously published heuristic methods is included in Section 8.3.

8.1. Not using constraints on the minimum blend and switching costs

A comparison of the results obtained by the proposed algorithm with those obtained by commercial solvers is presented in Table 5. For simplicity, we refer to our proposed algorithm as GO-PMCR when it uses piecewise McCormick relaxation to construct model **PR** and as GO-NMDT when it employs the NMDT.

ANTIGONE, BARON, and GO-PMCR compute the same solution for Examples 4 and 8. GO-NMDT computes the same solution for Example 4, but the final solution for Example 8 is slightly higher. GO-PMCR computes better solutions than GO-NMDT and ANTIGONE in all examples. In this work, this is because GO-PMCR can use more distinct numbers of partitions (2, 4, 8, 16, 32) than GO-NMDT (10, 100, 1000); thus, it generates more feasible solutions from the MILP relaxation.

BARON does not find a feasible solution for Examples 12 and 14 within 1 h. Based on the log files generated by commercial solvers, it seems that BARON relies more on the branching procedure, while ANTIGONE focuses more on the MILP relaxation and bound-tightening steps (as our proposed algorithm does). Feasible integer solutions for scheduling problems may be found only at deep nodes in the branch-and-bound tree [43], which can be of significant size when the number of binary variables is large.

In Examples 4 and 8, the algorithm and the commercial solvers compute similar optimality gaps. For Examples 12 and 14, BARON cannot compute an optimality gap (no feasible solution was found), and GO-PMCR obtains a lower optimality gap than GO-NMDT and ANTIGONE.

Both commercial solvers and the proposed algorithm did not solve all four examples to the desired tolerance within 1 h. The times reported in Table 5 are the times in which the best solution was found. GO-PMCR and GO-NMDT require shorter times than both commercial solvers. GO-NMDT is significantly faster than GO-PMCR only in the small-sized Example 4. Note that, for the number of partitions selected, the size of the MILP relaxation grows faster with GO-NMDT than with GO-PMCR. Therefore, MILP relaxations are often faster to solve to optimality with GO-PMCR. However, GO-PMCR will require more iterations. Based on the three factors considered (i.e., best solution found, optimality gap, and time to best solution), GO-PMCR shows the best performance. Fig. 9 shows the total number of binary variables in the relaxation of the scheduling model (i.e., model **PR**) when using PMCR and NMDT, at each iteration of the algorithm and for each example. It shows that, for the selected partition values, PMCR requires fewer binary variables in the first 4–5 iterations than NMDT at any iteration. This is expected since the partitions when using PMCR are fewer than 8 in those iterations, and NMDT starts with 10. Note that flat sections in the curves indicate that the OBBT method was applied instead of increasing the number of partitions.

The major differences between the proposed algorithm and BARON are:

- The use of piecewise relaxation methods for bilinear terms instead of standard McCormick envelopes; and
- Dynamically increasing the number of partitions in order to tighten the MILP relaxation, instead of implementing a branching strategy.

These features seem to be adequate for the scheduling problems presented here. We do not claim that our proposed algorithm will always be better than BARON when solving a different type of problem.

Our proposed algorithm and ANTIGONE perform similarly, but differ mainly in the following areas:

- The use of NMDT as a piecewise relaxation technique;
- How the number of partitions is increased in each iteration;
- When and how to apply OBBT; and
- The use of the CPLEX solution pool to store feasible solutions from the MILP relaxations and use them as starting points to solve the NLP problem.

Finally, ANTIGONE and BARON can handle more than just bilinear and quadratic terms; in addition, they apply other mathematical

Table 5
Summary of results (not using constraints on the minimum blend and switching costs).

Example ID	Best solution found (1000 USD)				Optimality gap (%)				Time to best solution (s)			
	ANTIGONE	BARON	GO-PMCR	GO-NMDT	ANTIGONE	BARON	GO-PMCR	GO-NMDT	ANTIGONE	BARON	GO-PMCR	GO-NMDT
4	4 633	4 633	4 633	4 633	6.90	6.37	6.31	6.37	2 350	930	616	120
8	8 203	8 203	8 203	8 223	9.53	9.53	9.19	9.49	1 708	3 273	714	1 391
12	16 650	NF	15 408	15 440	20.51	NA	14.00	14.22	1 631	3 600	1 411	1 438
14	21 360	NF	21 316	31 639	12.50	NA	12.31	40.92	3 600	3 600	2 911	2 904

NF = not found; NA = not available.

techniques capable of improving performance (e.g., reformulation-linearization technique, cutting planes, feasibility-based bound tightening, different branching strategies, etc.).

8.2. Using constraints on the minimum blend and switching costs

For this case, the results computed by the algorithm and commercial solvers are presented in Table 6. The most notable differences with respect to Table 5 are the smaller optimality gaps and the shorter times for Examples 4 and 8.

Both commercial solvers and the algorithm find similar solutions for Examples 4 and 8. BARON still does not find a feasible solution for Examples 12 and 14 within the allocated time. GO-PMCR computes better solutions than GO-NMDT for Examples 12 and 14, which in turn has better solutions than ANTIGONE. Note that the addition of bounds caused an increment in the number of solutions for Example 8. Since the solutions from Section 8.1 are still feasible even with the inclusion of the constraints regarding minimum blend and switching costs, this suggests that such constraints are affecting the solvers. This effect is also observed in ANTIGONE in Examples 12 and 14.

Regarding optimality gaps, most of the same observations as in Section 8.1 can be made. Similar optimality gaps are computed by all methods for Examples 4 and 8. For Examples 12 and 14, GO-PMCR calculates lower optimality gaps than GO-NMDT and ANTIGONE.

Both commercial solvers and the proposed algorithm solve Example 4 to the desired tolerance; BARON is the slowest. For Example 12, the time to the best solution required by GO-PMCR is larger than that required by GO-NMDT; however, it must be considered that GO-PMCR computes a better solution.

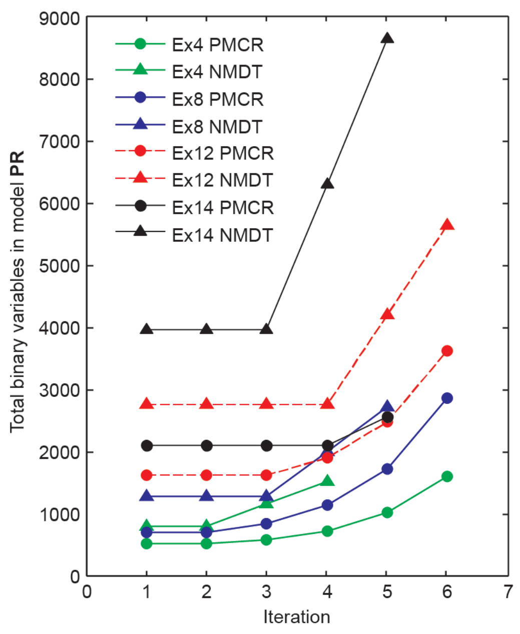

Fig. 9. Number of binary variables in model **PR** at each iteration of the algorithm.

Overall, GO-PMCR shows the best performance once again. For illustration purposes, the blend and delivery schedules computed for Example 14 by the algorithm using PMCR are shown in Fig. 10 and Fig. 11, respectively.

8.3. Comparison with heuristic methods

In this section, the proposed algorithm is compared with previously published heuristic methods [15,21]. Table 7 [15,21] contains the best solution found by those methods and the time required to compute such solutions. Note that heuristic methods do not compute an optimality gap since they aim to find close-to-optimal solutions very rapidly, and they do not spend time estimating and refining the value of the global optimal solution. These heuristic methods are tailored to the examples used in this work. These methods construct the final solution by decomposing the original problem into different levels, each one with different accuracy and complexity. Short execution times are achieved by solving the least complex level first and then, in each subsequent level, fixing the values of the most important variables to those from the previous level's solution.

The objective function of the scheduling model employed in this work is the same as the one used by Castillo and Mahalec [15]. This objective function penalizes each individual blend run, even when the same product is being blended in contiguous blend runs. On the other hand, Cerdá et al. [21] did not penalize the number of individual blend runs, but only penalized the product transitions in the blenders. We show the adjusted values of the solutions reported by Cerdá et al. [21]; that is, individual blend runs are penalized.

All methods find the same solution for Example 4. In general, all the methods compute very similar solutions for the remaining examples. Solutions from Cerdá et al. [21] have higher costs for Examples 8, 12, and 14 because they did not originally penalize individual blend runs. The method from Cerdá et al. [21] might compute similar solutions to those from the other methods if it used the same objective function.

Heuristic methods still require smaller execution times than the proposed global optimization algorithm. This is expected, because those methods do not involve as many steps as global optimization techniques. The proposed global optimization algorithm does not find solutions of the same quality as quickly as the two selected heuristic methods. To compute feasible solutions in each iteration, our proposed algorithm needs to first solve a MILP (i.e., model **PR**). The solution of the MILP is the most time-consuming step, thus reducing the speed required to compute new feasible solutions. Moreover, the small number of partitions at the beginning of the algorithm may result in weak MILP relaxations, which generate starting points for the NLP models that are far from the global optimum.

These results indicate the need to improve the corresponding step to compute feasible solutions, or to simply integrate heuristic methods into the proposed algorithm.

9. Conclusions

In this work, we presented a global optimization algorithm that

Table 6
Summary of results (using constraints on the minimum blend and switching costs).

Example ID	Best solution found (1000 USD)				Optimality gap (%)				Time to best solution (s)			
	ANTIGONE	BARON	GO-PMCR	GO-NMDT	ANTIGONE	BARON	GO-PMCR	GO-NMDT	ANTIGONE	BARON	GO-PMCR	GO-NMDT
4	4 633	4 633	4 633	4 633	0.01	0.01	0.01	0.01	26	296	30	14
8	8 207	8 204	8 206	8 204	0.05	0.02	0.04	0.02	557	1 218	103	140
12	23 590	NF	15 384	15 403	34.80	NA	0.01	0.13	3 333	3 600	2 674	742
14	23 520	NF	21 270	21 360	9.68	NA	0.13	0.55	1 636	3 600	2 574	2 845

NF = not found; NA = not available.

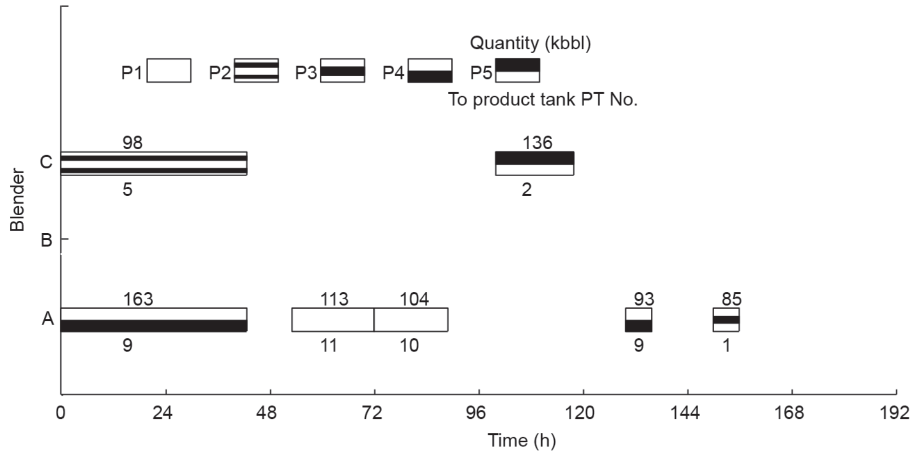

Fig. 10. Blend schedule computed for Example 14 by the proposed algorithm using PMCR. Kbbl is short for kilobarrel, 1 kbbl = 158.9873 m^3.

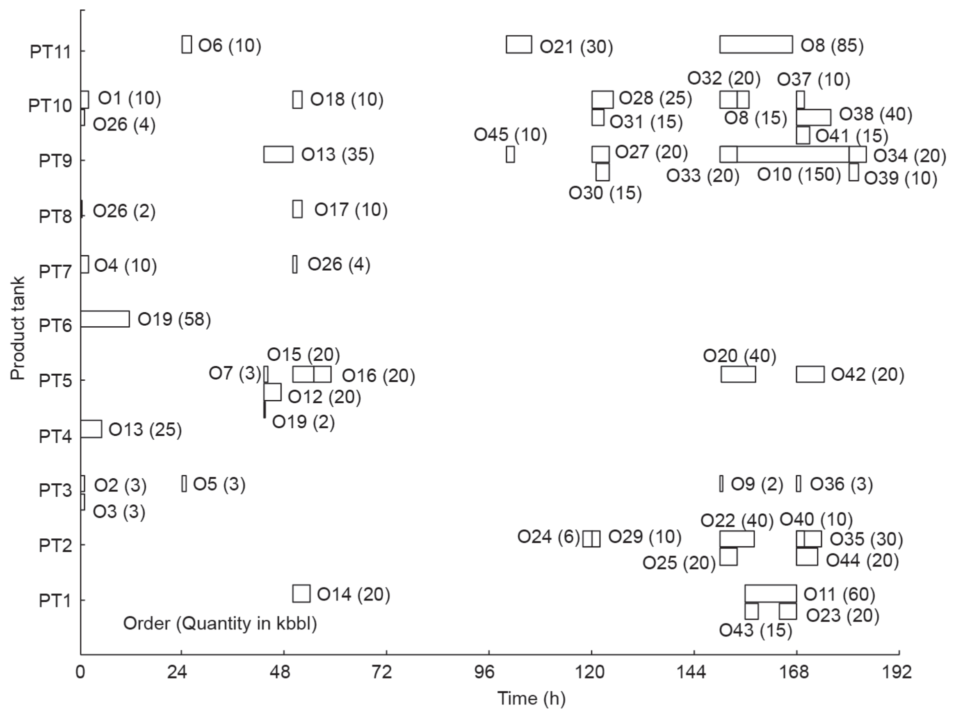

Fig. 11. Delivery schedule computed for Example 14 by the proposed algorithm using PMCR.

Table 7
Comparison with heuristic methods.

Example ID	Best solution found (1000 USD)					Time to best solution (s)			
	Castillo and Mahalec [15]	Cerdá et al. [21]	Cerdá et al. [21] adjusted values	GO-PMCR	GO-NMDT	Castillo and Mahalec [15]	Cerdá et al. [21]	GO-PMCR	GO-NMDT
4	4 633	4 613	4 633	4 633	4 633	3	0.4	30	14
8	8 203	8 163	8 223	8 206	8 204	6	7.5	103	140
12	15 403	15 342	15 442	15 384	15 403	17	31.0	2 674	742
14	21 263	21 181	21 301	21 270	21 360	24	21.0	2 574	2 845

can solve MINLP problems with bilinear and quadratic terms. The algorithm computes estimates of the global solution by constructing and solving MILP problems that are relaxations of the original problem obtained by using either PMCR or NMDT. These methods discretize the domain of one of the variables of a bilinear term into several partitions, and introduce extra binary and continuous variables into the model. To improve the estimates of the global optimum, the number of partitions is increased during the algorithm.

To avoid a rapid increase in the model size due to a large number of partitions, an OBBT method is used. The MILP relaxation will be closer to the original problem if the number of partitions stays the same but the domain of the variables is reduced. The OBBT method solves several LPs in a parallel setting.

The CPLEX solution pool is active and stores different feasible solutions found during the branch-and-bound procedure to solve the MILP relaxation. These solutions are then used as starting points for a nonlinear solver to find feasible solutions to the original problem. This step is also parallelized.

We showed that the proposed algorithm can be used to schedule gasoline-blending operations, taking into consideration the distribution problem and the most important operational constraints. We employ a continuous-time MINLP scheduling model [16] where the ethyl RT-70 models are used for octane blending.

The proposed algorithm was compared with two commercial solvers and two heuristic methods. The elements under evaluation were: the best solution found, the corresponding optimality gap, and the time to best solution. The proposed algorithm with PMCR showed a better performance than with NMDT. In our large-sized examples, the proposed algorithm with either PMCR or NMDT performed better than the commercial global solvers. This result shows that further research on this algorithm may be very promising. Both selected heuristic methods provided good solutions in shorter execution times than the global algorithms. This result indicates that the step to compute feasible solutions can still be improved.

We tested the performance of the algorithm by solving the scheduling model for two scenarios: ① not including lower bounds on the blend cost and switching costs, and ② including such bounds. The first problem is harder to solve and is representative of a kind of model one may write without diligently trying to reduce the search space as much as possible. Adding a tight lower bound to the blending cost as a constraint, as well as adding the lower bound to the switching costs, enables algorithms to search smaller spaces and improves their performance. This result also indicates that the relaxations are still not tight enough. Future work will include the derivation and addition of redundant and symmetry-breaking constraints; the testing of a different relaxation scheme for quadratic, cubic, and higher order terms (e.g., outer approximation); and the modification of the bound-tightening method in order to speed up the algorithm.

Acknowledgements

Support by Ontario Research Foundation, McMaster Advanced Control Consortium, and Fundação para a Ciência e Tecnologia (Investigador FCT 2013 program and project UID/MAT/04561/2013) is gratefully acknowledged.

Compliance with ethics guidelines

Pedro A. Castillo Castillo, Pedro M. Castro, and Vladimir Mahalec declare that they have no conflict of interest or financial conflicts to disclose.

Nomenclature

Sets and subscripts

BL = $\{bl\}$	Blenders
E = $\{e\}$	Quality properties
I = $\{i\}$	Blend components and corresponding storage tanks
N1 = $\{n\}$	Time slots
QN = $\{(\theta, n)\}$	Time slot n is associated with the period with quality profile θ

Parameters

$a_1, a_2, \ldots, a_6$	Coefficients for the ethyl RT-70 model
$Q_{\text{bc}}(i, e, \theta)$	Value of quality property e of blend component i during quality profile θ
$sens\ (i, \theta)$	Octane number sensitivity, i.e., octane difference RON – MON for blend component i during quality profile θ

Continuous variables

$Ar_{\text{avg}}(bl, n)$	Volumetric average of the aromatics content of the processed material by blender bl during slot n
$Ar^{\text{sq}}_{\text{avg}}(bl, n)$	Volumetric average of the squared value of the aromatics content of the processed material by blender bl during slot n
$Ar2_{\text{avg}}(bl, n)$	Squared value of $Ar_{\text{avg}}(bl, n)$
$Ar2^{\text{sq}}_{\text{avg}}(bl, n)$	Squared value of $Ar^{\text{sq}}_{\text{avg}}(bl, n)$
$Ar3_{\text{avg}}(bl, n)$	Product of $Ar^{\text{sq}}_{\text{avg}}(bl, n)$ and $Ar2_{\text{avg}}(bl, n)$
$Ar4_{\text{avg}}(bl, n)$	Squared value of $Ar2_{\text{avg}}(bl, n)$
$Ol_{\text{avg}}(bl, n)$	Volumetric average of the olefins content of the processed material by blender bl during slot n
$Ol^{\text{sq}}_{\text{avg}}(bl, n)$	Volumetric average of the squared value of the olefins content of the processed material by blender bl during slot n
$Ol2_{\text{avg}}(bl, n)$	Squared value of $Ol^{\text{sq}}_{\text{avg}}(bl, n)$
$Q_{\text{pr}}(bl, e, n)$	Value of quality property e of the processed material by blender bl during slot n
$r(i, bl, n)$	Volume fraction of blend component i going into blender bl during slot n
$r^{\text{MON}}_{\text{avg}}(bl, n)$	Volumetric average of the motor octane number of the processed material by blender bl during slot n
$r^{\text{RON}}_{\text{avg}}(bl, n)$	Volumetric average of the research octane number of the processed material by blender bl during slot n
$rs^{\text{MON}}_{\text{avg}}(bl, n)$	Product of $r^{\text{MON}}_{\text{avg}}(bl, n)$ and $sens_{\text{avg}}(bl, n)$
$rs^{\text{RON}}_{\text{avg}}(bl, n)$	Product of $r^{\text{RON}}_{\text{avg}}(bl, n)$ and $sens_{\text{avg}}(bl, n)$
$sens_{\text{avg}}(bl, n)$	Volumetric average of the octane number sensitivity of the processed material by blender bl during slot n
$sens^{\text{MON}}_{\text{avg}}(bl, n)$	Volumetric average of the octane number sensitivity times the motor octane number
$sens^{\text{RON}}_{\text{avg}}(bl, n)$	Volumetric average of the octane number sensitivity times the research octane number
$V_{\text{blend}}(bl, n)$	Volume being processed by blender bl during slot n
$V_{\text{comp}}(i, bl, n)$	Volume of blend component i transferred to blender bl during slot n

References

[1] Harjunkoski I, Maravelias CT, Bongers P, Castro PM, Engell S, Grossmann IE, et al. Scope for industrial applications of production scheduling models and solution methods. Comput Chem Eng 2014;62:161–93.
[2] Méndez CA, Grossmann IE, Harjunkoski I, Kaboré P. A simultaneous optimization approach for off-line blending and scheduling of oil-refinery operations. Comput Chem Eng 2006;30(4):614–34.
[3] Li J, Karimi I. Scheduling gasoline blending operations from recipe determination to shipping using unit slots. Ind Eng Chem Res 2011;50(15):9156–74.
[4] Li J, Xiao X, Floudas CA. Integrated gasoline blending and order delivery operations: Part I. Short-term scheduling and global optimization for single and multi-period operations. AIChE J 2016;62(6):2043–70.
[5] Singh A, Forbes JF, Vermeer PJ, Woo SS. Model-based real-time optimization of automotive gasoline blending operations. J Process Contr 2000;10(1):43–58.
[6] Joly M, Pinto JM. Mixed-integer programming techniques for the scheduling of fuel oil and asphalt production. Chem Eng Res Des 2003;81(4):427–47.
[7] Floudas CA, Lin X. Continuous-time versus discrete-time approaches for scheduling of chemical processes: A review. Comput Chem Eng 2004;28(11):2109–29.
[8] Sundaramoorthy A, Maravelias CT. Computational study of network-based mixed-integer programming approaches for chemical production scheduling. Ind Eng Chem Res 2011;50(9):5023–40.
[9] Maravelias CT. General framework and modeling approach classification for chemical production scheduling. AIChE J 2012;58(6):1812–28.
[10] Jia Z, Ierapetritou M. Mixed-integer linear programming model for gasoline blending and distribution scheduling. Ind Eng Chem Res 2003;42(4):825–35.
[11] Jia Z, Ierapetritou M. Efficient short-term scheduling of refinery operations based on a continuous time formulation. Comput Chem Eng 2004;28(6–7):1001–19.
[12] Glismann K, Gruhn G. Short-term scheduling and recipe optimization of blending processes. Comput Chem Eng 2001;25(4–6):627–34.
[13] Li J, Karimi I, Srinivasan R. Recipe determination and scheduling of gasoline blending operations. AIChE J 2010;56(2):441–65.
[14] Castillo PAC, Mahalec V. Inventory pinch based, multiscale models for integrated planning and scheduling—Part II: Gasoline blend scheduling. AIChE J 2014;60(7):2475–97.
[15] Castillo PAC, Mahalec V. Inventory pinch gasoline blend scheduling algo-

rithm combining discrete- and continuous-time models. Comput Chem Eng 2016;84:611–26.
[16] Castillo PAC, Mahalec V. Improved continuous-time model for gasoline blend scheduling. Comput Chem Eng 2016;84:627–46.
[17] Lotero I, Trespalacios F, Grossmann IE, Papageorgiou DJ, Cheon MS. An MILP-MINLP decomposition method for the global optimization of a source based model of the multiperiod blending problem. Comput Chem Eng 2016;87:13–35.
[18] Castro PM. New MINLP formulation for the multiperiod pooling problem. AIChE J 2015;61(11):3728–38.
[19] Kolodziej SP, Grossmann IE, Furman KC, Sawaya NW. A discretization-based approach for the optimization of the multiperiod blend scheduling problem. Comput Chem Eng 2013;53:122–42.
[20] Cerdá J, Pautasso PC, Cafaro DC. A cost-effective model for the gasoline blend optimization problem. AIChE J 2016;62(9):3002–19.
[21] Cerdá J, Pautasso PC, Cafaro DC. Optimizing gasoline recipes and blending operations using nonlinear blend models. Ind Eng Chem Res 2016;55(28):7782–800.
[22] Tawarmalani M, Sahinidis NV. A polyhedral branch-and-cut approach to global optimization. Math Program 2005;103(2):225–49.
[23] Misener R, Floudas CA. ANTIGONE: Algorithms for continuous/integer global optimization of nonlinear equations. J Glob Optim 2014;59(2):503–26.
[24] Boland N, Kalinowski T, Rigtering F. New multi-commodity flow formulations for the pooling problem. J Glob Optim 2016;66(4):669–710.
[25] Sherali HD, Alameddine A. A new reformulation-linearization technique for bilinear programming problems. J Glob Optim 1992;2(4):379–410.
[26] Ryoo HS, Sahinidis NV. A branch-and-reduce approach for global optimization. J Glob Optim 1996;8(2):107–38.
[27] Smith EMB, Pantelides CC. Global optimization of nonconvex MINLPs. Comput Chem Eng 1997;21(Suppl):S791–6.
[28] Belotti P, Lee J, Liberti L, Margot F, Wächter A. Branching and bounds tightening techniques for non-convex MINLP. Optim Methods Softw 2009;24(4–5):597–634.
[29] Achterberg T. SCIP: Solving constraint integer programs. Math Program Comput 2009;1(1):1–41.
[30] Castro PM. Spatial branch-and-bound algorithm for MIQCPs featuring multiparametric disaggregation. Optim Methods Softw. Epub 2016 Dec 13.
[31] Castillo PC, Castro PM, Mahalec V. Global optimization algorithm for large-scale refinery planning models with bilinear terms. Ind Eng Chem Res 2017;56(2):530–48.
[32] McCormick GP. Computability of global solutions to factorable nonconvex programs: Part I—Convex underestimating problems. Math Program 1976;10(1):147–75.
[33] Karuppiah R, Grossmann IE. Global optimization for the synthesis of integrated water systems in chemical processes. Comput Chem Eng 2006;30(4):650–73.
[34] Castro PM. Tightening piecewise McCormick relaxations for bilinear problems. Comput Chem Eng 2015;72:300–11.
[35] Misener R, Thompson JP, Floudas CA. APOGEE: Global optimization of standard, generalized, and extended pooling problems via linear and logarithmic partitioning schemes. Comput Chem Eng 2011;35(5):876–92.
[36] Kolodziej S, Castro PM, Grossmann IE. Global optimization of bilinear programs with a multiparametric disaggregation technique. J Glob Optim 2013;57(4):1039–63.
[37] Castro PM. Normalized multiparametric disaggregation: An efficient relaxation for mixed-integer bilinear problems. J Glob Optim 2016;64(4):765–84.
[38] Castro PM, Grossmann IE. Global optimal scheduling of crude oil blending operations with RTN continuous-time and multiparametric disaggregation. Ind Eng Chem Res 2014;53(39):15127–45.
[39] Castro PM. Source-based discrete and continuous-time formulations for the crude oil pooling problem. Comput Chem Eng 2016;93:382–401.
[40] Castillo PAC, Mahalec V, Kelly JD. Inventory pinch algorithm for gasoline blend planning. AIChE J 2013;59(10):3748–66.
[41] Healy WC, Maassen CW, Peterson RT. A new approach to blending octanes. In: Proceedings of the 24th Midyear Meeting of American Petroleum Institute's Division of Refining; 1959 May 27; New York, US; 1959. p. 132–136.
[42] Castro PM, Grossmann IE. Optimality-based bound contraction with multiparametric disaggregation for the global optimization of mixed-integer bilinear problems. J Glob Optim 2014;59(2):277–306.
[43] Kallrath J. Planning and scheduling in the process industry. OR Spectrum 2002;24(1):219–250.