Petroleum Science
https://doi.org/10.1007/s12182-019-00359-3

ORIGINAL PAPER

Efficient decomposition-based algorithm to solve long-term pipeline scheduling problem

S. Moradi[1] · S. A. MirHassani[1] · F. Hooshmand[1]

Received: 10 July 2018

Abstract
This paper addresses the scheduling and inventory management of a straight pipeline system connecting a single refinery to multiple distribution centers. By increasing the number of batches and time periods, maintaining the model resolution by using linear programming-based methods and commercial solvers would be very time-consuming. In this paper, we make an attempt to utilize the problem structure and develop a decomposition-based algorithm capable of finding near-optimal solutions for large instances in a reasonable time. The algorithm starts with a relaxed version of the model and adds a family of cuts on the fly, so that a near-optimal solution is obtained within a few iterations. The idea behind the cut generation is based on the knowledge of the underlying problem structure. Computational experiments on a real-world data case and some randomly generated instances confirm the efficiency of the proposed algorithm in terms of the solution quality and time.

Keywords Multi-product oil pipeline · Batch sequencing · Decomposition-based algorithm · Combinatorial cuts · Heuristic method

1 Introduction

Planning the transmission of oil products is a challenging problem in the oil industry. It is done by different modes including trucks, trains, vessels and pipelines; among them, pipelines have a great role in long-distance oil transportation because they are reliable and cheap, have little effect on traffic and environment, and provide the possibility of transferring large volumes.

In a pipeline system, oil products are produced at the refinery and injected at appropriate pumping rates into the pipeline in the form of batches, and then, these are discharged at distribution centers (DCs). Since there is no separator between consecutive batches, a small fraction of each batch will be mixed with the previous one and the resulting mixture is called an interface. A schedule is required to describe the product type of batches, their volume, and the beginning and ending times of each injection and discharging. The schedule should be feasible in the sense that daily demands of DCs are satisfied with no delay, pumping rate and storage capacity limits in the refinery and DCs are respected, forbidden sequences are avoided, etc. Additionally, the schedule should be optimal in the sense that the number of interactions between consecutive batches and the cost of energy consumed by different pumping rates is minimized. Achieving an optimal or even a feasible schedule is a difficult task in practice. Thus, using mathematical approaches instead of experimental methods is inevitable. In the last two decades, this problem has attracted the attention of many researchers from different viewpoints, and often, it is formulated as mixed-integer programming (MIP) models and solved using exact or heuristic algorithms. A broad overview on the related literature has been presented by Magatão et al. (2015) and Kirschstein (2018).

Straight pipelines are the simplest systems connecting a single refinery to a single DC (Bai and Rubin 2009; MirHassani and BeheshtiAsl 2013; Relvas et al. 2013; Moradi and MirHassani 2015; Zhang et al. 2016), a single refinery to multiple DCs (Rejowski and Pinto 2004; MirHassani et al. 2011; Cafaro et al. 2015) or multiple refineries to multiple DCs (Cafaro and Cerdá 2009; MirHassani et al. 2013). Pipeline systems with a tree structure are more complex than

Edited by Xiu-Qiu Peng

S. A. MirHassani
a_mirhassani@aut.ac.ir

[1] Department of Mathematics and Computer Science, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran

straight pipelines (MirHassani and Ghorbanalizadeh 2008; MirHassani and Fani Jahromi 2011; Mostafaei et al. 2015). In some studies, additional conditions such as dual-purpose points (Mostafaei et al. 2016) and reversible flow (Cafaro and Cerda 2014) have also been considered. Pipeline scheduling under demand uncertainty is another important topic, recently addressed by Moradi and MirHassani (2016).

Research on pipeline scheduling problems has led to different mathematical models, in which variables corresponding to injection and discharging times as well as volume and product type of batches play important roles. In addition, different heuristic methods have been proposed to tackle with this problem. In the heuristic method presented by Relvas et al. (2009), first a proper sequence of products is determined, and then, a feasible schedule is obtained by fixing the sequence into the model. MirHassani et al. (2011) proposed an algorithm that provides high-quality solutions for scheduling a system with one refinery and multiple DCs. At each iteration of their algorithm, the product type of some batches is fixed; then, it is continued to reach the final sequence. The heuristic method addressed by MirHassani and BeheshtiAsl (2013) is efficient for straight systems connecting a single refinery to a single DC. It first determines the sequence of products and effectively manages the inventory level of storage tanks by tracing the location of batches inside the pipeline. Fabro et al. (2014) proposed a new model to cope with heating constraints and sharing of tanks, and presented a decomposition-based heuristic to solve the problem. Magatão et al. (2011) combined constraint logic programming and mixed-integer linear programming (MILP) to formulate a variant of the pipeline scheduling problem.

On the one hand, the size of mathematical models proposed for pipeline scheduling problems becomes very large by increasing the number of batches and time periods, and linear programming (LP) relaxation-based methods such as branch and bound as well as commercial solvers would be inefficient to solve moderate- and large-sized instances of the problem in a reasonable time. On the other hand, the gap between the solution obtained by heuristic methods and the optimal solution typically increases by the number of days (MirHassani et al. 2011). Thus, providing an efficient method and facilitating the process to reach a near-optimal solution are useful and necessary. For this purpose, we make an attempt to use the advantage of decomposition-based algorithms.

A decomposition-based algorithm splits the original model into a master problem (MP) and a subproblem (SP) which are solved iteratively until a given optimality condition is observed. The advantage of such algorithms is that the MP and SP may be solved efficiently, whereas there may not exist an algorithm capable of tackling the whole problem at once. Benders' algorithm (Benders 1962) is a well-known decomposition-based method widely used to efficiently solve different complex problems. In this approach, the variables obtained by the MP are fixed to the SP, and the SP is solved for remaining variables. If the solution obtained by the MP is infeasible or non-optimal, feasibility or optimality cuts are generated and added to the MP. This process is repeated, and as a result of the addition of these cuts, the search space of the MP is gradually narrowed down as the algorithm proceeds, and the optimal solution is found within a finite number of iterations. For a recent overview on different applications and improvements of the Benders' decomposition algorithm, see Rahmaniani et al. (2017) and Beheshti Asl and MirHassani (2018).

Feasibility and optimality cuts of Benders' algorithm are generated with respect to the dual solution of the SP, and hence, the convexity of the SP is a necessary condition for the applicability of Benders' algorithm. However, if the SP is an MILP model, conventional Benders' decomposition is not applicable. In this situation, the combinatorial Benders' decomposition method may be beneficial. In this method, instead of Benders' cuts, which are generated based on the dual solution of the SP, combinatorial cuts (which are generated logically based on dominant logical rules of the problem) are utilized (Codato and Fischetti 2006). Combinatorial Benders' decomposition has been extensively used by many researchers. Maravelias (2006) utilized this method to tackle the problem of scheduling batch processes, where the main model was decomposed into an MP consisting of assignment decisions and SP corresponding to sequencing decisions. Hooker (2007), by using logic-based Benders' decomposition, combined mixed-integer linear programming and constraint logic programming to solve an important class of planning and scheduling problems. Bai and Rubin (2009) and Chen et al. (2012) utilized combinatorial Benders' decomposition algorithms to tackle the quayside operation problem and minimum tollbooth problem, respectively. Verstichel et al. (2015) and Akpinar et al. (2017) applied this approach to solve the lock scheduling problem and assembly line balancing problem, respectively. Recently, Hooshmand and MirHassani (2018) have utilized combinatorial cuts to develop an efficient decomposition-based algorithm for a bi-level programming problem.

In this paper, we concentrate on the problem of scheduling and inventory management of a straight pipeline system connecting a single refinery to multiple DCs, referred to as the long-term multi-product pipeline scheduling problem (LMPSP). In this regard, we consider the model proposed by MirHassani et al. (2011) as a base. Since directly solving moderate- and large-sized instances of the problem is not possible in a reasonable time, developing an efficient method is important. Motivated by combinatorial Benders' decomposition algorithms, in this paper, we present a decomposition-based heuristic method to efficiently solve the LMPSP. The algorithm starts with a relaxed version of the model and adds a family of combinatorial cuts on the fly, so that a near-optimal solution is obtained within a few iterations. The idea behind our cut generation is novel and it is based on the knowledge of the underlying problem structure.

The rest of this paper is organized as follows. Section 2 provides a detailed description as well as the formulation of the LMPSP. Section 3 describes a decomposition-based heuristic algorithm to solve the problem. The efficiency of the proposed approach is investigated in Sect. 4. Finally, Sect. 5 draws conclusions and offers directions for future research.

2 Mathematical formulation

2.1 Problem description

A straight pipeline connecting a refinery to a specific set of DCs is given. Different oil products are produced at the refinery and transmitted to DCs through the pipeline. The pipeline is always full, and the only way to discharge a volume of a product (at one or more DCs) is to inject the same quantity into the pipeline from the refinery. Each product is injected into the pipeline in the form of several batches with a large volume. Various batches are pumped from the refinery into the pipeline back to back, without any separator, and the same quantity is discharged at one or more DCs. During an injection, a small fraction of the incoming batch will be mixed with the previous one, making a mixture, namely the interface or contamination volume. In each DC, there are some tanks and each product is stored in its own tank.

The planning horizon is divided in days, and the demand of each DC for each product on each day is a deterministic parameter. At the beginning of the planning horizon, the pipeline is full with a given set of batches referred to as old batches. Additionally, the term "new batches" refers to the batches injected during the planning horizon. The product type and volume of each old batch are assumed to be known.

The following assumptions are made:

A1 Consecutive batches cannot contain the same product.
A2 Consecutive injection of some pairs of products is forbidden.
A3 Contamination volume between every pair of batches is a known constant, regardless of product type and batch volume.
A4 To prevent more contamination in the pipeline, the volume of contamination is never discharged at intermediate DCs and will remain in the pipeline until reaching the last DC.
A5 Depending on the product type, the volume of each batch is limited to a specific interval.
A6 During the injection of a batch, two different pumping rates (i.e., minimum and maximum pumping rates) are available. It should be decided how much of a batch volume is injected at the minimum pumping rate and how much is injected at the maximum pumping rate.
A7 During the injection of a new batch, discharging priority is with the batches that are placed at the entrance of DCs farther away from the refinery. In other words, the rule "farthest DC first served" is applied.
A8 The level of inventory of each product at each DC is limited to a specific interval.
A9 Daily demand of each product at each DC must be met before the end of the day.
A10 The capacity of production in the refinery is unlimited.
A11 A batch may be discharged at multiple DCs.
A12 Pumping costs are equal regardless of the product being pumped.

Injection decisions (including the sequence of products that should be injected into the pipeline, volume, injection time and pumping rate of each batch) as well as discharging decisions (including volume and delivering time of discharged lots at each DC) should be made so that daily demands at DCs of each product are supplied on time and operational restrictions are satisfied. Two types of costs are considered here: the cost of interfaces between batches and costs associated with minimum and maximum pumping rates. Decisions should be made so that the total cost is minimized. As mentioned earlier, we refer to this problem as the LMPSP.

2.2 Mathematical model

In this section, the problem is formulated as an MILP model inspired by MirHassani et al. (2011).

2.2.1 Notations

Let $\mathbb{I} = \{1, 2, \dots, I\}$ (indexed by i) be the set of batches and assume that the elements of $\mathbb{I}$ are sorted in ascending order, according to which $i < i'$ implies that the batch i is injected earlier than the batch i'. The set $\mathbb{I}$ is divided into two subsets: The first one, denoted by $\mathbb{I}^{old}$, contains the batches that had already been injected into the pipeline, and the second one, denoted by $\mathbb{I}^{new}$, includes the batches that may be injected within the planning horizon. Additionally, let $\mathbb{N} = \{1, \dots, N\}$ (indexed by n, n') be an ordered set of DCs sorted in ascending order in their distance to the refinery. Other notations are summarized in Table 1.

Remark 1 Regarding the parameter BD_i, it is worth mentioning that the upper volumetric distance of the batch i indicates the volumetric distance from the refinery to the right boundary of the batch i. However, the lower volumetric distance

Table 1 Sets, indices and parameters

$\mathbb{I} = \{1, \dots, I\} = \mathbb{I}^{old} \cup \mathbb{I}^{new}$	Set of batches indexed by i, i'
$\mathbb{I}^{old}$	A subset of $\mathbb{I}$ indicating the set of old batches
$\mathbb{I}^{new}$	A subset of $\mathbb{I}$ indicating the set of new batches Note that except for the indices i, i', we may refer to elements of $\mathbb{I}^{new}$ by indices j, j' as well
NEW_1	The first element of the set $\mathbb{I}^{new}$ with respect to the predefined order on $\mathbb{I}^{new}$
$\mathbb{P}$	Set of oil products indexed by p, p'
T	Number of days in the planning horizon
$\mathbb{T} = \{1, 2, \dots, T\}$	Set of days, indexed by t, t'
N	Number of DCs along the pipeline
$\mathbb{N} = \{1, \dots, N\}$	Set of DCs along the pipeline indexed by n, n'
$\hat{\delta}_{i,p}$	Binary parameter that is 1 if the old batch $i \in \mathbb{I}^{old}$ contains the product p and 0 otherwise
$INIT_{p,n}$	Initial inventory level of the product p at the nth DC
BD_i	Upper volumetric distance of the old batch $i \in \mathbb{I}^{old}$ from the refinery at the beginning of the planning horizon (see Remark 1)
REM_i	Remaining volume of old batch $i \in \mathbb{I}^{old}$ inside the pipeline at the beginning of the planning horizon
$FOR_{p,p'}$	Binary parameter indicating whether the injection of the product p immediately after the product p' is forbidden ($=1$) or not ($=0$). Note that since consecutive batches cannot contain the same product, we set $FOR_{p,p} = 1$ for every $p \in \mathbb{P}$
DCD_n	Volumetric distance of the nth DC from the refinery
H_t	Total hours from the beginning of the planning horizon to the end of the day t
H_{MAX}	Total hours from the beginning to the end of the planning horizon (note that $H_{\mathrm{MAX}} = H_{\mathrm{T}}$)
$DEM_{p,t,n}$	Demand of the nth DC from the product p on the day t
$WASTE$	Contaminated volume of each batch due to the interface
$\overline{PUMP}$	Maximum pumping rate
$\underline{PUMP}$	Minimum pumping rate
$\overline{VOL}_p$	An upper bound on the volume of a batch containing the product p
$\underline{VOL}$	A lower bound on the volume of a batch
$\overline{DEL}$	An upper bound on the volume delivered from a batch to a DC
$\underline{DEL}$	A lower bound on the volume delivered from a batch to a DC
$\overline{INV}_{p,n}$	An upper bound on the level of inventory of the product p at the nth DC
$\underline{INV}_{p,n}$	A lower bound on the level of inventory of the product p at the nth DC
c_1	The cost of each interface
c_2	The cost of pumping a volumetric unit at the minimum pumping rate
c_3	The cost of pumping a volumetric unit at the maximum pumping rate
α_p	A lower bound on the number of required batches of the product p in any feasible solution of the problem (see Remark 4)

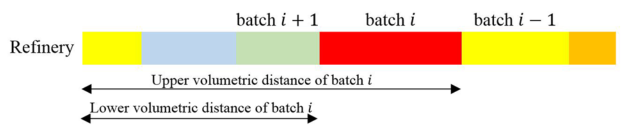
batch $i+1$
batch i
batch $i-1$
Refinery
Upper volumetric distance of batch i
Lower volumetric distance of batch i

Fig. 1 Illustration of upper and lower volumetric distances

of the batch i means the volumetric distance from the refinery to the left boundary of the batch i. See Fig. 1 for more illustration.

Remark 2 A specific batch i may be discharged at a given nth DC during the injection of multiple new batches. However, the volumetric fraction of the batch i which is delivered to the nth DC is said to be available to satisfy demands just at the time the delivery of the batch i to the nth DC is quite terminated (i.e., after this time, the remaining amount of batch i (if any) is not delivered to nth DC).

Decision variables are defined in Table 2.

2.2.2 Mathematical model

With respect to the notations, introduced in the previous subsection, the problem is formulated as the following MILP model.

Table 2 Decision variables

$\delta_{i,p}$	Binary variable that is 1 if the batch i contains the product p, otherwise 0 ($\forall i \in \mathbb{I},\ p \in \mathbb{P}$)
$\gamma_{i,j,n}$	Binary variable that is 1 if a portion of the batch i is delivered to the nth DC during the injection of the batch j, otherwise 0 ($\forall i \in \mathbb{I}, j \in \mathbb{I}^{new} : j \geq i, \forall p \in \mathbb{P}$)
$\eta_{i,t,n}$	Binary variable equals 1 if the batch i is available at the nth DC before the end of the day t, otherwise 0 ($\forall i \in \mathbb{I}, t \in \mathbb{T}, n \in \mathbb{N}$). See Remark 2
x_j^V	Volume of the batch j injected to the pipeline ($\forall j \in \mathbb{I}^{new}$)
x_j^{MIN}	Volume of the batch j that is injected at the minimum pumping rate ($\forall j \in \mathbb{I}^{new}$)
x_j^{MAX}	Volume of the batch j injected at the maximum pumping rate ($\forall j \in \mathbb{I}^{new}$)
x_j^L	Duration of the injection of the batch j expressed in hours ($\forall j \in \mathbb{I}^{new}$)
x_j^H	The time point at which the injection of the batch j is completed ($\forall j \in \mathbb{I}^{new}$)
$y_{i,j}^D$	Upper volumetric distance of the batch i from the refinery at the time x_j^H ($\forall i \in \mathbb{I}, j \in \mathbb{I}^{new} : j \geq i$)
$y_{i,j}^R$	Remaining volume of the batch i inside the pipeline at the time x_j^H ($\forall i \in \mathbb{I}, j \in \mathbb{I}^{new} : j \geq i$)
$u_{i,j,n}^D$	The volume of the batch i discharged at the nth DC during the injection of the batch j($\forall i \in \mathbb{I}, j \in \mathbb{I}^{new} : j \geq i, \forall n \in \mathbb{N}$)
$u_{i,j,n}^L$	Discharging duration of the batch i at the nth DC during the injection of the batch j($\forall i \in \mathbb{I}, j \in \mathbb{I}^{new} : j \geq i, \forall n \in \mathbb{N}$)
$w_{i,p,t,n}$	Volumetric fraction of the batch i which contains the product p and is available at the nth DC on the day t($\forall i \in \mathbb{I}, p \in \mathbb{P}, t \in \mathbb{T}, n \in \mathbb{N}$). See Remark 2 for more information on the concept of availability
$\theta_{i,n}$	The time point at which the batch i is available at the nth DC ($\forall i \in \mathbb{I}, n \in \mathbb{N}$). See Remark 2 for more information on the concept of availability

LMPSP:

$$\min z = c_1 \sum_{i \in \mathbb{I}} \sum_{p \in \mathbb{P}} \delta_{i,p} + c_2 \sum_{j \in \mathbb{I}^{new}} x_j^{\text{MIN}} + c_3 \sum_{j \in \mathbb{I}^{new}} x_j^{\text{MAX}} \tag{1}$$

$$\delta_{i,p} = \hat{\delta}_{i,p} \quad \forall i \in \mathbb{I}^{old},\ p \in \mathbb{P} \tag{2}$$

$$\sum_{p \in \mathbb{P}} \delta_{j,p} \leq 1 \quad \forall j \in \mathbb{I}^{new} \tag{3}$$

$$\sum_{p \in \mathbb{P}} \delta_{j+1,p} \leq \sum_{p \in \mathbb{P}} \delta_{j,p} \quad \forall j \in \mathbb{I}^{new},\ j < I \tag{4}$$

$$\delta_{j,p} + \delta_{j-1,p'} \leq 1 \quad \forall j \in \mathbb{I}^{new},\ p, p' \in \mathbb{P} : FOR_{p,p'} = 1 \tag{5}$$

$$\underline{VOL} \sum_{p \in \mathbb{P}} \delta_{j,p} \leq x_j^V \leq \sum_{p \in \mathbb{P}} \left(\delta_{j,p} \overline{VOL}_p \right) \quad \forall j \in \mathbb{I}^{new} \tag{6}$$

$$x_j^V = x_j^{\text{MIN}} + x_j^{\text{MAX}} \quad \forall j \in \mathbb{I}^{new} \tag{7}$$

$$x_j^V = \sum_{i \in \mathbb{I} : i \leq j} \sum_{n \in \mathbb{N}} u_{i,j,n}^D \quad \forall j \in \mathbb{I}^{new} \tag{8}$$

$$y_{i,j}^D = y_{i+1,j}^D + y_{i,j}^R \quad \forall i \in \mathbb{I},\ j \in \mathbb{I}^{new} : i < j \tag{9}$$

$$y_{i,NEW_1}^D = BD_i + x_{NEW_1}^V - \sum_{i' \in \mathbb{I} : i' \geq i} \sum_{n \in \mathbb{N}} u_{i',NEW_1,n}^D \quad \forall i \in \mathbb{I}^{old} \tag{10}$$

$$y_{i,j}^D = y_{i,j-1}^D + x_j^V - \sum_{i' \in \mathbb{I} : i' \geq i} \sum_{n \in \mathbb{N}} u_{i',j,n}^D \quad \forall i \in \mathbb{I},\ j \in \mathbb{I}^{new} : i \langle j \text{ and } j \rangle NEW_1 \tag{11}$$

$$y_{j,j}^D = x_j^V - \sum_{n \in \mathbb{N}} u_{j,j,n}^D \quad \forall j \in \mathbb{I}^{new} \tag{12}$$

$$y_{j,j}^R = y_{j,j}^D \quad \forall j \in \mathbb{I}^{new} \tag{13}$$

$$y_{i,NEW_1}^R = REM_i - \sum_{n \in \mathbb{N}} u_{i,NEW_1,n}^D \quad \forall i \in \mathbb{I}^{old} \tag{14}$$

$$y_{i,j}^R = y_{i,j-1}^R - \sum_{n \in \mathbb{N}} u_{i,j,n}^D \quad \forall i \in \mathbb{I},\ j \in \mathbb{I}^{new} : j > i \text{ and } j > NEW_1 \tag{15}$$

$$\sum_{n \in \mathbb{N}} \sum_{i \in \mathbb{I} : i \leq j} \gamma_{i,j,n} \geq \sum_{p \in \mathbb{P}} \delta_{j,p} \quad \forall j \in \mathbb{I}^{new} \tag{16}$$

$$\sum_{n \in \mathbb{N}} u_{i,NEW_1,n}^D \leq REM_i \quad \forall i \in \mathbb{I}^{old} \tag{17}$$

$$\sum_{n \in \mathbb{N}} u_{i,j,n}^D \leq y_{i,j-1}^R \quad \forall i \in \mathbb{I},\ j \in \mathbb{I}^{new},\ j > i \text{ and } j > NEW_1 \tag{18}$$

$$\sum_{n \in \mathbb{N} : n < N} u_{i,NEW_1,n}^D \leq REM_i - \left(WASTE \sum_{p \in \mathbb{P}} \delta_{i,p} \right) \quad \forall i \in \mathbb{I} : i \leq NEW_1 \tag{19}$$

$$\sum_{n\in\mathbb{N}:n<N} u^D_{i,j,n} \le y^R_{i,j-1} - \left(WASTE \sum_{p\in\mathbb{P}} \delta_{i,p}\right) \quad \forall i \in \mathbb{I},\ j \in \mathbb{I}^{new} j \ge i \text{ and } j > NEW_1 \tag{20}$$

$$\underline{DEL}\gamma_{i,j,n} \le u^D_{i,j,n} \le \overline{DEL}\gamma_{i,j,n} \quad \forall i \in \mathbb{I},\ j \in \mathbb{I}^{new},\ j \ge i,\ \forall n \in \mathbb{N} \tag{21}$$

$$y^D_{i,j} - \left(WASTE \sum_{p\in\mathbb{P}} \delta_{i,p}\right) \ge \gamma_{i,j,n} DCD_n \quad \forall i \in \mathbb{I},\ j \in \mathbb{I}^{new} : j \ge i,\ \forall n \in \mathbb{N} :\ n < N \tag{22}$$

$$y^D_{i,j} \ge \gamma_{i,j,N} DCD_N \quad \forall i \in \mathbb{I},\ j \in \mathbb{I}^{new} : j \ge i \tag{23}$$

$$y^D_{i+1,j-1} + \sum_{i'\in\mathbb{I}:i'\le i} \sum_{n'\in\mathbb{N}:n'\ge n} u^D_{i',j,n'} \le DCD_n + DCD_N\left(1-\gamma_{i,j,n}\right)$$
$$\forall i \in \mathbb{I},\ j \in \mathbb{I}^{new} : j > i,\ \forall n \in \mathbb{N} \tag{24}$$

$$x^L_j = \frac{x^{\text{MIN}}_j}{\underline{PUMP}} + \frac{x^{\text{MAX}}_j}{\overline{PUMP}} \quad \forall j \in \mathbb{I}^{new} \tag{25}$$

$$x^L_j = \sum_{n\in\mathbb{N}} \sum_{i\in\mathbb{I}:i\le j} u^L_{i,j,n} \quad \forall j \in \mathbb{I}^{new} \tag{26}$$

$$\frac{u^D_{i,j,n}}{\overline{PUMP}} \le u^L_{i,j,n} \le \frac{u^D_{i,j,n}}{\underline{PUMP}}, \quad \forall i \in \mathbb{I},\ j \in \mathbb{I}^{new} : j \ge i,\ \forall n \in \mathbb{N} \tag{27}$$

$$x^H_j = x^H_{j-1} + x^L_j \quad \forall j \in \mathbb{I}^{new} \tag{28}$$

$$x^H_j \le H_{\text{MAX}} \quad \forall j \in \mathbb{I}^{new} \tag{29}$$

$$\theta_{i,n} \ge x^H_j - x^L_j + \sum_{i'\in\mathbb{I}:i'\le i} \sum_{n'\in\mathbb{N}:n'\ge n} u^L_{i',j,n'} - H_{\text{MAX}}\left(1-\gamma_{i,j,n}\right)$$
$$\forall i \in \mathbb{I},\ j \in \mathbb{I}^{new} : j \ge i,\ \forall n \in \mathbb{N} \tag{30}$$

$$\theta_{i,n} \le x^H_j - x^L_j + \sum_{i'\in\mathbb{I}:i'\le i} \sum_{n'\in\mathbb{N}:n'\ge n} u^L_{i',j,n'} + H_{\text{MAX}} \sum_{j'\in\mathbb{I}^{new}:j'>j} \gamma_{i,j',n}$$
$$\forall i \in \mathbb{I},\ j \in \mathbb{I}^{new} : j \ge i, \forall n \in \mathbb{N} \tag{31}$$

$$H_t\left(1-\eta_{i,t,n}\right) \le \theta_{i,n} \le H_t + H_{\text{MAX}}\left(1-\eta_{i,t,n}\right) \quad \forall i \in \mathbb{I},\ t \in \mathbb{T},\ n \in \mathbb{N} \tag{32}$$

$$\eta_{i,t,n} \le \eta_{i,t+1,n} \quad \forall i \in \mathbb{I},\ t \in \mathbb{T} :\ t < T,\ \forall n \in \mathbb{N} \tag{33}$$

$$\underline{INV}_{p,n} \le INIT_{p,n} + \sum_{i\in\mathbb{I}} w_{i,p,t,n} - \sum_{t'\in\mathbb{T}:t'\le t} DEM_{p,t',n} \le \overline{INV}_{p,n}$$
$$\forall t \in \mathbb{T},\ p \in \mathbb{P},\ n \in \mathbb{N} \tag{34}$$

$$w_{i,p,t,n} \le \sum_{j\in\mathbb{I}^{new}:j\ge i} u^D_{i,j,n} \quad \forall i \in \mathbb{I},\ t \in \mathbb{T},\ p \in \mathbb{P},\ n \in \mathbb{N} \tag{35}$$

$$w_{i,p,t,n} \ge \sum_{j\in\mathbb{I}^{new}:j\ge i} u^D_{i,j,n} - M\left(2-\delta_{i,p}-\eta_{i,t,n}\right) \quad \forall i \in \mathbb{I},\ t \in \mathbb{T},\ p \in \mathbb{P},\ n \in \mathbb{N} \tag{36}$$

$$w_{i,p,t,n} \le M\delta_{i,p} \quad \forall i \in \mathbb{I},\ t \in \mathbb{T},\ p \in \mathbb{P},\ n \in \mathbb{N} \tag{37}$$

$$w_{i,p,t,n} \le M\eta_{i,t,n} \quad \forall i \in \mathbb{I},\ t \in \mathbb{T},\ p \in \mathbb{P},\ n \in \mathbb{N} \tag{38}$$

$$\delta_{i,p} \in \{0,1\} \quad \forall i \in \mathbb{I},\ p \in \mathbb{P} \tag{39}$$

$$\gamma_{i,j,n} \in \{0,1\} \quad \forall i \in \mathbb{I},\ j \in \mathbb{I}^{new} : j \ge i,\ \forall p \in \mathbb{P} \tag{40}$$

$$\eta_{i,t,n} \in \{0,1\} \quad \forall i \in \mathbb{I},\ t \in \mathbb{T},\ n \in \mathbb{N} \tag{41}$$

$$x^V_j, x^{\text{MIN}}_j, x^{\text{MAX}}_j, x^L_j, x^H_j \ge 0 \quad \forall j \in \mathbb{I}^{new} \tag{42}$$

$$y^D_{i,j}, y^R_{i,j} \ge 0 \quad \forall i \in \mathbb{I},\ j \in \mathbb{I}^{new} : j \ge i \tag{43}$$

$$u^L_{i,j,n}, u^D_{i,j,n} \ge 0 \quad \forall i \in \mathbb{I}, j \in \mathbb{I}^{new} : j \ge i, \forall n \in \mathbb{N} \tag{44}$$

$$q_i \ge 0 \quad \forall i \in \mathbb{I} \tag{45}$$

$$w_{i,p,t,n} \ge 0 \quad \forall i \in \mathbb{I},\ p \in \mathbb{P},\ t \in \mathbb{T},\ n \in \mathbb{N} \tag{46}$$

$$\theta_{i,n} \ge 0 \quad \forall i \in \mathbb{I},\ n \in \mathbb{N} \tag{47}$$

Objective function (1) minimizes the cost of interfaces between the batches and the cost associated with pumping rates. Constraint set (2) expresses the product type of the old batches. Constraint set (3) ensures that each batch contains at most one product. Constraint set (4) indicates that if a batch is empty, all of its subsequent batches should be empty as well. Constraint set (5) prohibits the consecutive injection of the products p and p' if $FOR_{p,p'} = 1$. Constraint set (6) indicates that the volume of an empty batch is zero and the volume of a non-empty batch containing the product p is limited to the interval $\left[\underline{VOL}, \overline{VOL}_p\right]$. Constraint set (7) ensures that the volume of each batch is equal to the amount of that batch injected at the minimum pumping rate plus that injected at the maximum pumping rate. Constraint set (8) expresses that when a new batch j with volume x^V_j is pumped into the pipeline, a volume equal to x^V_j of some lots inside the pipeline is discharged.

Injecting any new batch causes the movement of previous batches inside the pipeline. Constraint sets (9)–(12) update the volumetric distance of the batches inside the pipeline when the injection of a new batch j is terminated. Specifically, as pointed out by MirHassani et al. (2011) constraint set (9) indicates that the upper volumetric distance of batch

i at the completion time of pumping the batch j is equal to the remaining volume of batch j in the pipeline plus the upper volumetric distance of the batch behind it (i.e., the batch $i+1$) at the time x_j^H. Consider the value of the upper volumetric distance of batch i at the completion time of injecting batch $j-1$ (i.e., $y_{i,j-1}^D$). After the injection of the new batch j, this value is increased at most as much as the volume of the new batch j, but during the injecting of batch j a portion of batch i and the batches behind it ($i' \geq i$) may be discharged. Thus, $y_{i,j-1}^D$, after the injection of new batch j, is increased as much as $x_j^V - \sum_{i' \in \mathbb{I}: i' \geq i} \sum_{n \in \mathbb{N}} u_{i',j,n}^D$. This is ensured by constraint sets (10) and (11). For more details, see MirHassani et al. (2011). Constraint set (12) indicates that when the injection of a new batch j is completed, its upper volumetric distance is equal to the volume of the batch j minus the portion of its volume discharged during its injection.

Constraint set (13) states that at the time x_j^H, the remaining volume of the batch j in the pipeline equals its volumetric distance. Constraint sets (14) and (15) update the remaining volume of the batches inside the pipeline when a new batch is injected. Constraint set (16) indicates that if the batch j is not empty, then during its injection, discharging occurs at some DCs. Constraint set (17) expresses that during the injection of the batch $j = NEW_1$, the total discharged volume of the batch i is at most equal to the remaining volume of the batch i at the beginning of the planning horizon. However, constraint set (18) ensures that during the injection of the batch $j > NEW_1$, the total discharged volume of the batch i is at most equal to the remaining volume of the batch i at the time x_{j-1}^H. Constraint sets (19) and (20) handle the interface with respect to Assumption A4 and indicate that from the total quantity of any batch i, at least $\left(WASTE \sum_{p \in \mathbb{P}} \delta_{i,p}\right)$ units are reserved for DC N, where $\left(WASTE \sum_{p \in \mathbb{P}} \delta_{i,p}\right)$ denotes the contaminated volume of batch i. Constraint set (21) ensures that the volume delivered from a batch to a DC lies in a specified interval. Constraint sets (22) and (23) satisfy Assumption A4 and indicate that during the injection of the batch j, if a fraction of the batch i is discharged at the nth DC (i.e., $\gamma_{i,j,n} = 1$), then the upper volumetric distance of the batch i should be greater than or equal to either $DCD_n + \left(WASTE \sum_{p \in \mathbb{P}} \delta_{i,p}\right)$ (if $n < N$) or DCD_N (if $n = N$). Additionally, at the time x_{j-1}^H, the lower volumetric distance of the batch i equals $y_{i+1,j-1}^D$. However, after injecting the batch j, it increases by the volume of batches $i' \leq i$ which are discharged during the injection of the batch j. Thus, the term $y_{i+1,j-1}^D + \sum_{i' \in \mathbb{I}: i' \leq i} \sum_{n' \in \mathbb{N}: n' \geq n} u_{i',j,n'}^D$ denotes the lower volumetric distance of the batch i at the time x_j^H. Clearly, if this term is greater than DCD_n, then the batch i cannot be discharged at nth DC. This is ensured by constraint set (24). Note that constraint set (24) observes the discharging priority rule stated in Assumption A7, as well.

Constraint set (25) calculates the time spent to inject the total volume of the batch j into the pipeline, and constraint set (26) implies that this time should be equal to the summation of time spent to discharge the same quantity at DCs during the injection of the batch j. Constraint set (27) expresses lower and upper bounds on the variable $u_{i,j,n}^L$. Constraint set (28) indicates that the injection of the batch j starts after the completion of the injection of the batch $j-1$ and is completed after x_j^L hours. Additionally, constraint set (29) ensures that the injection of all non-empty batches is terminated before H_{MAX}.

Constraint sets (30) and (31) ensure the satisfaction of the point addressed in Remark 2. Constraint sets (32) and (33) ensure that the binary variable $\eta_{i,t,n}$ is equal to 1 if and only if the batch i is available at the nth DC before the end of the day t. To calculate the daily inventory level of the product p at each DC, the total discharged volume until the day t is added to the initial inventory, and then, the total demand until the day t is subtracted. Inventory level should not exceed the available storage capacity and it should be more than the specified minimum level. This is ensured by constraint set (34). The value of the variable $w_{i,p,t,n}$ is determined by constraint sets (35)–(38) where $M = \max_{p \in \mathbb{P}} \left(\overline{VOL}_p\right)$. Finally, constraint sets (39)–(47) define the domain of variables.

Remark 3 For the ease of modeling, it is assumed that only two levels are possible for the pumping rate; however, considering $x_j^{\text{MIN}*}$ and $x_j^{\text{MAX}*}$ as the values of variables x_j^{MIN} and x_j^{MAX} in the optimal solution of the model, and with respect to constraints (25)–(27), it can be concluded that in real life when implementing the optimal solution of the proposed model, the actual values of pumping rates used during the injection of the batch j can be set on any value in the continuous range $\left[\underline{PUMP}, \overline{PUMP}\right]$ provided that the total time of injecting batch j is equal to $\frac{x_j^{\text{MIN}*}}{\underline{PUMP}} + \frac{x_j^{\text{MAX}*}}{\overline{PUMP}}$, as demonstrated by constraint (25).

Remark 4 It is clear that α_p, with the following definition, introduces a lower bound on the number of required batches of the product p in any feasible solution of the problem LMPSP.

$$\alpha_p = \left\lceil \frac{\sum_{t \in \mathbb{T}} \sum_{n \in \mathbb{N}} DEM_{p,t,n} - \sum_{n \in \mathbb{N}} \left(INIT_{p,n} - \underline{INV}_{p,n}\right)}{\overline{VOL}_p} \right\rceil \quad \forall p \in \mathbb{P} \tag{48}$$

Thus, constraint set (49) is a valid inequality for the LMPSP, strengthening the LP relaxation bound and hence improving the running time of LP relaxation-based methods such as branch and bound.

$$\sum_{i\in\mathbb{I}} \delta_{i,p} \geq \alpha_p \quad \forall p \in \mathbb{P} \tag{49}$$

3 Decomposition-based algorithm

In this section, we make an attempt to take advantage of the special properties of the problem and propose an efficient decomposition-based heuristic. Our preliminary experiments indicate that by fixing the variable $\delta_{i,p}$ in the LMPSP (i.e., determining the sequence of products), we obtain a reduced model which can be solved quickly. Therefore, we present a decomposition-based heuristic that iteratively solves two problems, namely MP and SP. The MP contains the variable $\delta_{i,p}$ and is initially constructed from constraint sets (2)–(5), (39) and (49). Let $\mathbb{L}$ (indexed by ℓ) be a set containing iteration indices. During the algorithm, two sets of cuts (51) and (52) (which are called feasibility and improvement cuts, respectively) are added to MP. Later, we explain the role these cuts in more detail, but note that at each iteration of the algorithm, either a feasibility cut or an improvement cut is generated. Thus, we define an indicator parameter IND_ℓ equal to 1 if at the ℓth iteration, a feasibility cut is generated, and 0 if at the ℓth iteration, an improvement cut is generated. The generated cuts remain in MP until the end of algorithm, and at the beginning of the algorithm, MP starts with $\mathbb{L} = \emptyset$.

MP:

$$\min c_1 \sum_{i\in\mathbb{I}} \sum_{p\in\mathbb{P}} \delta_{i,p} \tag{50}$$

s.t. (2)–(5), (39) and (49)

$$\sum_{i\in\mathbb{I}^{(\ell)}_{p,after}} \sum_{p\in\mathbb{P}:\, \tilde{\delta}^{(\ell)}_{i,p}=0} \delta_{i,p} \geq 1 \quad \forall \ell \in \mathbb{L} : IND_\ell = 1 \quad \text{(Feasibility cut)} \tag{51}$$

$$\sum_{i\in\mathbb{I}^{new}:\, i\geq j^{(\ell)}} \sum_{p\in\mathbb{P}:\, \tilde{\delta}^{(\ell)}_{i,p}=0} \delta_{i,p} \geq 1 \quad \forall \ell \in \mathbb{L} : IND_\ell = 0 \quad \text{(Improvement cut)} \tag{52}$$

Let $\tilde{\delta}^{(\ell)}$ be the sequence obtained by the MP at the ℓth iteration of the algorithm. Once the MP is solved, the feasibility of $\tilde{\delta}^{(\ell)}$ needs to be verified. For this purpose, the corresponding subproblem, denoted by SP($\tilde{\delta}^{(\ell)}$), is solved. The problem SP($\tilde{\delta}^{(\ell)}$) contains constraints (6)–(47) together with constraint set (56), fixing the variable $\delta_{i,p}$ to $\tilde{\delta}^{(\ell)}_{i,p}$.

SP ($\tilde{\delta}^{(\ell)}$):

$$\min c_2 \sum_{j\in\mathbb{I}^{new}} x^{\mathrm{MIN}}_j + c_3 \sum_{j\in\mathbb{I}^{new}} x^{\mathrm{MAX}}_j + c_M \sum_{p\in\mathbb{P}, t\in\mathbb{T}, n\in\mathbb{N}} s_{p,t,n} \tag{53}$$

s.t. (6)–(33), (35)–(47)

$$\underline{INV}_{p,n} - s_{p,t,n} \leq INIT_{p,n} + \sum_{i\in\mathbb{I}} w_{i,p,t,n} - \sum_{t'\in\mathbb{T}:\, t'\leq t} DEM_{p,t',n} \leq \overline{INV}_{p,n}$$
$$\forall t \in \mathbb{T},\ p \in \mathbb{P},\ n \in \mathbb{N} \tag{54}$$

$$s_{p,t,n} \geq 0 \quad \forall p \in \mathbb{P},\ t \in \mathbb{T},\ n \in \mathbb{N} \tag{55}$$

$$\delta_{i,p} = \tilde{\delta}^{(\ell)}_{i,p} \quad \forall i \in \mathbb{I},\ p \in \mathbb{P} \tag{56}$$

Note that the sequence $\tilde{\delta}^{(\ell)}$ obtained by the MP may violate constraint set (34). Therefore, in the SP, we use constraint set (54) instead of (34) where $s_{p,t,n}$ is a new nonnegative variable that indicates the shortage volume of the product p on the day t at the nth DC and is penalized in the objective function by a sufficiently large coefficient, namely c_M. Due to the existence of the variable $s_{p,t,n}$ in constraint set (54), the SP will not be infeasible. However, since demands should be satisfied with no delay, if in the optimal solution of SP($\tilde{\delta}^{(\ell)}$), we have $\sum_{p\in\mathbb{P},t\in\mathbb{T},n\in\mathbb{N}} s_{p,t,n} > 0$, then the sequence $\tilde{\delta}^{(\ell)}$ is considered infeasible and a feasibility cut is added to the MP to remove the sequence $\tilde{\delta}^{(\ell)}$ from its feasible region. However, if $\sum_{p\in\mathbb{P},t\in\mathbb{T},n\in\mathbb{N}} s_{p,t,n} = 0$, the sequence is feasible and its optimality should be investigated. In this regard, if the total pumping cost is greater than a given threshold, $\tilde{\delta}^{(\ell)}$ is considered as a non-optimal sequence and a cut referred to as an improvement cut is added to the MP to remove the sequence $\tilde{\delta}^{(\ell)}$ from its feasible region. By adding feasibility and improvement cuts to the MP, the feasible region of the MP is narrowed down, and finally, the best found solution is returned. In the next subsections, we describe how the feasibility and improvement cuts are derived.

3.1 Feasibility cut

Timely delivery of products is very important, and the capacities of pipelines and storage tanks are usually enough to meet the demands. Thus, planners prefer that the batches, even if using the high pumping rate, reach DCs so that $\sum_{p\in\mathbb{P},t\in\mathbb{T},n\in\mathbb{N}} s_{p,t,n} = 0$. In this problem, we assume that the system is able to supply the demands without any delay (see Assumption A9). Thus, a sequence is infeasible if $\sum_{p\in\mathbb{P},t\in\mathbb{T},n\in\mathbb{N}} s_{p,t,n} > 0$.

The output of the MP defines a sequence of products denoted by $\tilde{\delta}^{(\ell)}$. In the optimal solution of the problem SP($\tilde{\delta}^{(\ell)}$), if $\sum_{p\in\mathbb{P},t\in\mathbb{T},n\in\mathbb{N}} s_{p,t,n} > 0$, we conclude that $\tilde{\delta}^{(\ell)}$ is an infeasible sequence to the LMPSP, and a feasibility cut should be added to the MP to remove this sequence from its feasible region. The general form of this cut is as follows:

$$\sum_{i\in\mathbb{I},p\in\mathbb{P}:\, \tilde{\delta}^{(\ell)}_{i,p}=0} \delta_{i,p} + \sum_{i\in\mathbb{I},p\in\mathbb{P}:\, \tilde{\delta}^{(\ell)}_{i,p}=1} \left(1 - \delta_{i,p}\right) \geq 1 \tag{57}$$

where $\tilde{\delta}_{i,p}^{(\ell)}$ denotes the value of the variable $\delta_{i,p}$ achieved from the MP at the ℓth iteration of the algorithm. The above cut is usually poor and can eliminate just one infeasible sequence (i.e., the current sequence) per iteration. Since the current solution could not be improved by reducing the number of batches, cut (57) can be replaced by a little stronger cut as follows:

$$\sum_{i\in\mathbb{I},p\in\mathbb{P}:\ \tilde{\delta}_{i,p}^{(\ell)}=0} \delta_{i,p} \geq 1 \tag{58}$$

Note that the efficiency and speed of the algorithm significantly depend on the strength of cuts. Cut (58) is not the most effective one and should be strengthened by utilizing the underlying problem structure. In this regard, we make an attempt to propose an approach to generate a stronger cut.

Suppose that the current sequence $\tilde{\delta}^{(\ell)}$ is infeasible; therefore, at least, one DC is faced with shortage of one or more products. Assume that the product $\bar{p}$ is not supplied on time (i.e., $\sum_{t\in\mathbb{T},n\in\mathbb{N}} s_{\bar{p},t,n} > 0$), and let $FB_{\bar{p}}^{(\ell)} \in \mathbb{I}^{new}$ be the index of the first batch allocated to the product $\bar{p}$ in the sequence $\tilde{\delta}^{(\ell)}$, i.e., $FB_{\bar{p}}^{(\ell)} = \min\left\{ i \in \mathbb{I}^{new} : \tilde{\delta}_{i,\bar{p}}^{(\ell)} = 1 \right\}$. We divide the current sequence into the following two subsequences:

$$\mathbb{I}_{\bar{p},before}^{(\ell)} = \left\{ i \in \mathbb{I} : i < FB_{\bar{p}}^{(\ell)} \right\}, \tag{59}$$

$$\mathbb{I}_{\bar{p},after}^{(\ell)} = \left\{ i \in \mathbb{I} : i \geq FB_{\bar{p}}^{(\ell)} \right\}, \tag{60}$$

where $\mathbb{I}_{\bar{p},before}^{(\ell)}$ contains the batches injected before the batch $FB_{\bar{p}}^{(\ell)}$ and $\mathbb{I}_{\bar{p},after}^{(\ell)}$ contains the other batches. As an example, Fig. 2 shows a sequence containing nine batches of four products (8 non-empty batches and 1 fictitious batch). Assume that this sequence leads to the lack of product P4, then the sets $\mathbb{I}_{\mathrm{P4},before}^{(\ell)}$ and $\mathbb{I}_{\mathrm{P4},after}^{(\ell)}$ are determined as follows:

$$\mathbb{I}_{\mathrm{P4},before}^{(\ell)} = \{1, 2, 3, 4\},$$

$$\mathbb{I}_{\mathrm{P4},after}^{(\ell)} = \{5, 6, 7, 8, 9\}$$

We claim that the feasibility cut can be stated as (51) imposing that the component of at least one batch of the part $\mathbb{I}_{\bar{p},after}^{(\ell)}$

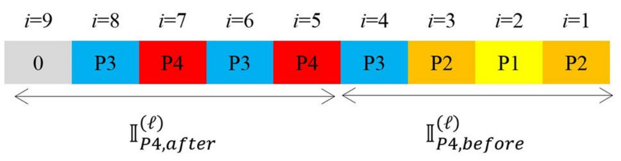
i=9
i=8
i=7
i=6
i=5
i=4
i=3
i=2
i=1
0
P3
P4
P3
P4
P3
P2
P1
P2
$\mathbb{I}_{P4,after}^{(\ell)}$
$\mathbb{I}_{P4,before}^{(\ell)}$

Fig. 2 Illustration of dividing a sequence based on shortage of product P4

should be changed. The following proposition proves the validity of this cut.

Proposition 1 *Let $\tilde{\delta}^{(\ell)}$ be an infeasible sequence achieved from the MP at the ℓth iteration of the algorithm, and assume that by implementing this sequence, the demand of product $\bar{p}$ cannot be satisfied on time. Then, (51) is a valid cut, i.e., it removes the infeasible sequence $\tilde{\delta}^{(\ell)}$ but is does not eliminate any feasible solution of the LMPSP.*

Proof It is clear that (51) removes $\tilde{\delta}^{(\ell)}$ from the feasible region of MP. It is enough to show that other sequences removed by (51) are also infeasible to LMPSP. Let $\mathbb{S}^{(\ell)}$ be the set of all sequences at which the part $\mathbb{I}_{\bar{p},after}^{(\ell)}$ remains the same as that of $\tilde{\delta}^{(\ell)}$ and only the part $\mathbb{I}_{\bar{p},before}^{(\ell)}$ changes. In other words, $\mathbb{S}^{(\ell)}$ contains all sequences removed by cut (51). We show that none of the elements of $\mathbb{S}^{(\ell)}$ is feasible to LMPSP. Since the product $\bar{p}$ is not supplied on time, it can be concluded that either the number of batches containing $\bar{p}$ is not sufficient, or at least one batch of $\bar{p}$ is injected late. For every sequence $\bar{\delta}$ in the set $\mathbb{S}^{(\ell)}$, at least one of the following cases may occur:*Case 1* Compared with $\tilde{\delta}^{(\ell)}$, in the sequence $\bar{\delta}$, only the order of the batches of the part $\mathbb{I}_{\bar{p},before}^{(\ell)}$ is changed, but their product type remains unchanged. In this case, it is clear that the volume injected before $\bar{p}$ is not reduced, and hence, the product $\bar{p}$ is not injected earlier. Thus, $\bar{\delta}$ encounters a lack of $\bar{p}$ as well, and hence, it is infeasible to LMPSP.*Case 2* Compared with $\tilde{\delta}^{(\ell)}$, in the sequence $\bar{\delta}$, the product type of at least one batch in the part $\mathbb{I}_{\bar{p},before}^{(\ell)}$ is changed to $\bar{p}$. Let $i' \in \mathbb{I}_{\bar{p},before}^{(\ell)}$ be a batch containing the product p' in the sequence $\tilde{\delta}^{(\ell)}$. If in the sequence $\bar{\delta}$, the product type of the batch i' is changed to $\bar{p}$, then $\bar{\delta}$ encounters a lack of the product p'. This is because of two facts. First, the MP minimizes the number of injected batches. Second, for every fixed sequence $\tilde{\delta}^{(\ell)}$, the SP determines the batch volumes so that the total delay is minimized.Therefore, it can be concluded that to obtain a feasible solution, the part $\mathbb{I}_{\bar{p},after}^{(\ell)}$ should be changed. (Note that if the part $\mathbb{I}_{\bar{p},after}^{(\ell)}$ is changed, the part $\mathbb{I}_{\bar{p},before}^{(\ell)}$ may change or remain unchanged.) By adding cut (51) to the MP, in addition to the current infeasible sequence, $\tilde{\delta}^{(\ell)}$, many other infeasible sequences are also omitted; however, this cut does not violate any feasible solution to the LMPSP. Hence, (51) is a valid feasibility cut.

Feasibility cut (51) not only removes the current infeasible sequence $\tilde{\delta}^{(\ell)}$ from the MP feasible region, but also may eliminate some other solutions to the MP that happen to be infeasible to the LMPSP. Thus, (51) is stronger than cuts (57) and (58).

As an example, assuming that the sequence depicted in Fig. 2 leads to the lack of product P4, the sequence 0→P3

$\rightarrow$P4$\rightarrow$P3$\rightarrow$P4$\rightarrow$P2$\rightarrow$P3$\rightarrow$P1$\rightarrow$P2 is clearly infeasible, but cut (58) is not able to remove it from the MP feasible region. However, if the cut $\sum_{i\in\mathbb{I},p\in\mathbb{P}|i\geq 5\text{and}\tilde{\delta}^{(\ell)}_{i,p}=0}\delta_{i,p}\geq 1$ [generated with respect to (51)] is added to the MP, some infeasible sequences such as 0$\rightarrow$P3$\rightarrow$P4$\rightarrow$P3$\rightarrow$P4$\rightarrow$P2$\rightarrow$P3$\rightarrow$P1$\rightarrow$P2 and 0$\rightarrow$P3$\rightarrow$P4$\rightarrow$P3$\rightarrow$P4$\rightarrow$P2$\rightarrow$P1$\rightarrow$P3$\rightarrow$P2 are also removed from the MP feasible region.

Remark 5 Assume that the sequence $\tilde{\delta}^{(\ell)}$ obtained by MP leads to the shortage of more than one product, and let $\mathbb{Q}$ contain such products. Then, feasibility cut (51) is written based on the product $\bar{p}\in\mathbb{Q}$ whose corresponding set $\mathbb{I}^{(\ell)}_{p,after}$ has the smallest cardinality, i.e.,

$$\bar{p}=\underset{p\in\mathbb{Q}}{\operatorname{argmin}}\left(\left|\mathbb{I}^{(\ell)}_{p,after}\right|\right)\tag{61}$$

For example, assuming that the sequence depicted in Fig. 2 leads to the lack of the products P3 and P4, we have:

$$\mathbb{I}^{(\ell)}_{\text{P3},after}=\{4,5,6,7,8,9\},\quad \mathbb{I}^{(\ell)}_{\text{P4},after}=\{5,6,7,8,9\}$$

Since $\left|\mathbb{I}^{(\ell)}_{\text{P4},after}\right|<\left|\mathbb{I}^{(\ell)}_{\text{P3},after}\right|$, we set $\bar{p}=\text{P4}$, and hence, feasibility cut (51) is written as $\sum_{i\in\mathbb{I},p\in\mathbb{P}|i\geq 5\text{and}\tilde{\delta}^{(\ell)}_{i,p}=0}\delta_{i,p}\geq 1$.

3.2 Improvement cut

In order to obtain good quality solutions with minimum pumping rate costs, the utilization of improvement cuts is necessary. For this purpose, when a feasible solution is obtained, it is checked whether or not it satisfies a given fitness condition. If yes, the algorithm is terminated and the best solution found so far is returned. Otherwise, an improvement cut is generated to remove this solution from the feasible region of MP. As a fitness criterion, we define $\overline{elb}$ as an estimated lower bound on the total volume that can be injected at the maximum pumping rate. For a given feasible solution, if $\sum_{j\in\mathbb{I}^{new}}x^{\text{MAX}}_j\leq\overline{elb}$, we say that the fitness condition is established and the best solution found so far is reported as a near-optimal solution. However, if $\sum_{j\in\mathbb{I}^{new}}x^{\text{MAX}}_j>\overline{elb}$, the fitness condition is not established and an improvement cut is generated and added to the MP to remove the current solution from the MP feasible region. Let $j^{(\ell)}$ be the first batch injected at the maximum pumping rate, i.e., $x^{\text{MAX}}_{j^{(\ell)}}>0$ and $\sum_{j\in\mathbb{I}^{new}:j<j^{(\ell)}}x^{\text{MAX}}_j=0$. Our preliminary experiments indicate that the inequality (52) may treat as an improvement cut imposing that the part $\{i\in\mathbb{I}|i\geq j^{(\ell)}\}$ of the current sequence should be changed. As a result of the addition of this cut, some sequences are omitted. In all such deleted sequences, compared with the current sequence, only the part $\{i\in\mathbb{I}|i<j^{(\ell)}\}$ changes; however, the injected volume before the batch $j^{(\ell)}$ usually does not decrease. Therefore, typically, the cost of the using maximum pumping rate for eliminated sequences is not less than the best observed solution.

Note that the accuracy of estimation of $\overline{elb}$ is very important. If the estimated value for $\overline{elb}$ is more than the optimal value of $\sum_{j\in\mathbb{I}^{new}}x^{\text{MAX}}_j$, the first sequence that satisfies $\sum_{j\in\mathbb{I}^{new}}x^{\text{MAX}}_j\leq\overline{elb}$ is introduced as a near-optimal solution. However, if the estimated value for $\overline{elb}$ is less than the optimal value for $\sum_{j\in\mathbb{I}^{new}}x^{\text{MAX}}_j$, the algorithm is repeated until the MP becomes infeasible. $\overline{elb}$ can be obtained as follows:

$$\overline{elb}\geq\left[\sum_{p\in\mathbb{P},n\in\mathbb{N},t'\in\mathbb{T}:t'\leq t}DEM_{p,t',n}-\sum_{p\in\mathbb{P},n\in\mathbb{N}}\left(INIT_{p,n}-\underline{INV}_{p,n}\right)-H_t\underline{PUMP}\right]\left(\frac{\overline{PUMP}}{\overline{PUMP}-\underline{PUMP}}\right)\quad\forall t\in\mathbb{T}\tag{62}$$

To justify the above inequality, note that $\sum_{p\in\mathbb{P},n\in\mathbb{N},t'\in\mathbb{T}:t'\leq t}DEM_{p,t',n}-\sum_{p\in\mathbb{P},n\in\mathbb{N}}\left(INIT_{p,n}-\underline{INV}_{p,n}\right)$ can be interpreted as the total net volume that should be injected to the pipeline within the interval $[0,H_t]$. We denote this value by φ_t. Additionally, let $\tau^{\min}_t$ and $\tau^{\max}_t$ represent the duration of injecting petroleum products by minimum and maximum pumping rates within the interval $[0,H_t]$, respectively. Clearly, we have:

$$\begin{cases}\tau^{\min}_t\underline{PUMP}+\tau^{\max}_t\overline{PUMP}=\varphi_t\\ \tau^{\min}_t+\tau^{\max}_t=H_t\end{cases}$$

By solving the above system, we get

$$\tau^{\max}_t=\frac{\varphi_t-H_t\underline{PUMP}}{\overline{PUMP}-\underline{PUMP}}$$

Thus, the total volume injected at the maximum pumping rate within the interval $[0,H_t]$ can be estimated as follows:

$$\frac{\varphi_t-H_t\underline{PUMP}}{\overline{PUMP}-\underline{PUMP}}\times\overline{PUMP}$$

Accordingly, the result is established. The value of $\overline{elb}$ can be estimated from the historical data, as well. In other words, if the demand pattern of the current planning horizon is somewhat similar to that of a given previous planning horizon, $\overline{elb}$ can be estimated with respect to the optimal plan of that horizon. However, it is a rare condition in practice.

3.3 Overall framework of the proposed algorithm

Regarding the previous subsections, the main steps of the algorithm are summarized as follows:

Decomposition-based heuristic method

```
• Let $\ell_0$ be the iteration counter and set $\ell_0 \coloneqq 1$ and initialize $\mathbb{L} := \emptyset$. Let $UB^*$ represent the best upper bound found so far and initialize it to $+\infty$; moreover, denoting the sequence corresponding to $UB^*$ with $\delta^*$. Furthermore, considering the parameter $Ctrl$ as an indicator showing whether the stopping criterion is met and set $Ctrl \coloneqq 0$.
• While $Ctrl = 0$ do
    • Solve the MP.
    • If the MP is infeasible, then
        • Check whether $UB^* < \infty$ . If $UB^* < \infty$, return $\delta^*$ and $UB^*$ as the best found solution and set $Ctrl := 1$; otherwise, conclude that the LMPSP is infeasible and set $Ctrl := 1$.
    • Else
        • Denote the MP optimal solution by $\tilde{\delta}^{(\ell_0)}$ and solve the SP($\tilde{\delta}^{(\ell_0)}$).
        • If $\sum_{p\in\mathbb{P},t\in\mathbb{T},n\in\mathbb{N}} s_{p,t,n} > 0$ (in the optimal solution of the SP($\tilde{\delta}^{(\ell_0)}$)) then
            • Create a feasibility cut in the form of (51) and set $\mathbb{L} \coloneqq \mathbb{L}\cup\{\ell_0\}$, and $IND_{\ell_0} \coloneqq 1$.
        • Else if $\sum_{p\in\mathbb{P},t\in\mathbb{T},n\in\mathbb{N}} s_{p,t,n} = 0$ (in the optimal solution of the SP($\tilde{\delta}^{(\ell_0)}$)) then
            • Denote the objective value corresponding to the current feasible solution by $UB \coloneqq c_1 \sum_{i\in\mathbb{I},p\in\mathbb{P}} \tilde{\delta}_{i,p}^{(\ell)} + c_2 \sum_{j\in\mathbb{I}^{new}} x_j^{\text{MIN}} + c_3 \sum_{j\in\mathbb{I}^{new}} x_j^{\text{MAX}}$ . If $UB < UB^*$, update $UB^*$ to $UB$ and set $\delta^* = \tilde{\delta}^{(\ell_0)}$.
            • If the fitness condition $\sum_{j\in\mathbb{I}^{new}:j<j^{(\ell)}} x_j^{\text{MAX}} \le \overline{elb}$ is satisfied (with respect to the optimal solution of SP($\tilde{\delta}^{(\ell_0)}$)), then
                • Return $\delta^*$ and $UB^*$ as the best found solution and set $Ctrl := 1$.
            • Else
                • Create an improvement cut in the form of (52) and set $\mathbb{L} \coloneqq \mathbb{L}\cup\{\ell_0\}$, and $IND_{\ell_0} \coloneqq 0$.
            • Endif.
        • Endif.
    • Endif.
    • Set $\ell_0 \coloneqq \ell_0 + 1$.
• Endwhile.
```

Note that since the sets of batches and products are bounded, the total number of sequences is finite; moreover, since at each iteration of the proposed algorithm, at least one sequence is eliminated, the algorithm terminates within a finite number of iterations.

4 Computational results

In this section, first, the algorithm is evaluated on a real-world data case. Then, some randomly generated instances are considered to further analyze the proposed algorithm in terms of the runtime and solution quality. Experiments are carried out on a PC with Intel 2 GHz processor and 2 GB

Table 3 Minimum/maximum batch sizes and allowed sequences

p	Min batch size $\underline{VOL}$, vu	Max batch size $\overline{VOL}_p$, vu	Allowed/forbidden sequences			
			P1	P2	P3	P4
P1	500	12,000	×	✓	✓	×
P2	500	15,000	✓	×	✓	×
P3	500	18,000	✓	✓	×	✓
P4	500	22,000	×	×	✓	×

Table 4 Permissible storage level in each DC (vu)

p	Level	DC		
		*n*1	*n*2	*n*3
P1	Max	7425	7425	9870
	Min	400	400	800
P2	Max	8330	16,970	16,550
	Min	800	1000	1200
P3	Max	19,980	23,050	50,215
	Min	1800	2000	5000
P4	Max	26,540	23,660	30,968
	Min	2500	3900	6200

RAM. The algorithm is coded in the AIMMS mathematical modeling language 3.12 (Bisschop 2012), and optimization models are solved using ILOG CPLEX 12.4 solver (ILOG 2011) included in the AIMMS software.

4.1 Implementation of the proposed algorithm on a real-world data case

In this section, the proposed algorithm is evaluated on a real-world system, partially taken from MirHassani et al. (2011). In this system, four oil derivatives are produced in a refinery and delivered to three DCs by a direct pipeline. General information about this system is given in Tables 3 and 4. Moreover, maximum and minimum batch sizes and forbidden sequences are presented in Table 3, and permissible storage levels for each product are shown in Table 4. The minimum and maximum pumping rates are equal to 400 and 720 volumetric units (vu) per hour, respectively.

The volumetric coordinates of DCs from the refinery are 4963 vu, 7242 vu and 14,181 vu, respectively. Initial inventory and the total demand of 11 days for each DC and various products are summarized in Table 5.

With respect to Eq. (48), the lower bound of the number of required batches that should be injected is equal to $\alpha_{\mathrm{P1}} = 1$, $\alpha_{\mathrm{P2}} = 3$, $\alpha_{\mathrm{P3}} = 1$ and $\alpha_{\mathrm{P4}} = 3$. Moreover, regarding (62), we obtain $elb = 0$.

In the first iteration of the algorithm, the MP is solved with no cut (i.e., $\mathbb{L} = \emptyset$ and $\mathbb{L}' = \emptyset$) and the sequence $0 \to \mathrm{P2} \to \mathrm{P1} \to \mathrm{P2} \to \mathrm{P3} \to \mathrm{P4} \to \mathrm{P3} \to \mathrm{P4} \to \mathrm{P3} \to \mathrm{P2} \to \mathrm{P3} \to \mathrm{P4}$ (denoted by $\tilde{\delta}^{(1)}$) is obtained. The optimal solution of the $\mathrm{SP}(\tilde{\delta}^{(1)})$ indicates that product P1 faces shortage. We have $\mathbb{I}^{(1)}_{\mathrm{P1},after} = \{i10, i11, i12\}$, and accordingly, a feasibility cut is generated and added to the MP. Until this moment, we have $\mathbb{L} = \{1\}$ and $IND_1 = 1$.

In the second iteration, the MP is solved and the sequence $0 \to \mathrm{P4} \to \mathrm{P3} \to \mathrm{P2} \to \mathrm{P1} \to \mathrm{P2} \to \mathrm{P3} \to \mathrm{P4} \to \mathrm{P3} \to \mathrm{P2} \to \mathrm{P3}$

Table 5 Initial inventory and the total demand

	Initial inventory, vu			Total demand, vu		
	*n*1	*n*2	*n*3	*n*1	*n*2	*n*3
P1	500	660	876	1032	1050	1695
P2	2064	1527	2748	5965	6497	24,237
P3	3152	3021	7898	2972	4678	7338
P4	2583	5181	8000	7211	7131	35,392

Table 6 Description of each iteration of the algorithm

Iteration	Running time, s	Total shortage volume, vu	Products faced with shortage	Total volume injected at $\overline{PUMP}$, vu	Feasibility	Fitness
1	25	2341	P1	–	No	No
2	25	1540	P1, P4	–	No	No
3	32	13,699	P1, P2, P4	–	No	No
4	11	0	–	27,497	Yes	No
5	47	205	P1, P2	–	No	No
6	16	0	–	34,160	Yes	No
7	98	20,500	P1, P2, P4	–	No	No
8	31	32	P1		No	No
9	23	0	–	27,497	Yes	No
10	17	0	–	0	Yes	Yes

Table 7 Sequences obtained at each iteration of the proposed algorithm

Iteration	Sequence												Status of sequence
	*i*12	*i*11	*i*10	*i*9	*i*8	*i*7	*i*6	*i*5	*i*4	*i*3	*i*2	*i*1	
1	0	P2	P1	P2	P3	P4	P3	P4	P3	P2	P3	P4	Infeasible
2	0	P4	P3	P2	P1	P2	P3	P4	P3	P2	P3	P4	Infeasible
3	0	P2	P3	P2	P1	P2	P3	P4	P3	P4	P3	P4	Infeasible
4	0	P4	P3	P4	P3	P2	P3	P2	P1	P2	P3	P4	Feasible
5	0	P2	P3	P4	P3	P2	P1	P2	P3	P4	P3	P4	Infeasible
6	0	P2	P3	P4	P3	P4	P3	P2	P1	P2	P3	P4	Feasible
7	0	P4	P3	P2	P3	P2	P1	P2	P3	P4	P3	P4	Infeasible
8	0	P4	P3	P4	P3	P2	P1	P2	P3	P2	P3	P4	Infeasible
9	0	P4	P3	P2	P3	P4	P3	P2	P1	P2	P3	P4	Feasible
10	P2	P3	P4	P3	P4	P3	P2	P1	P2	P1	P3	P4	Optimal

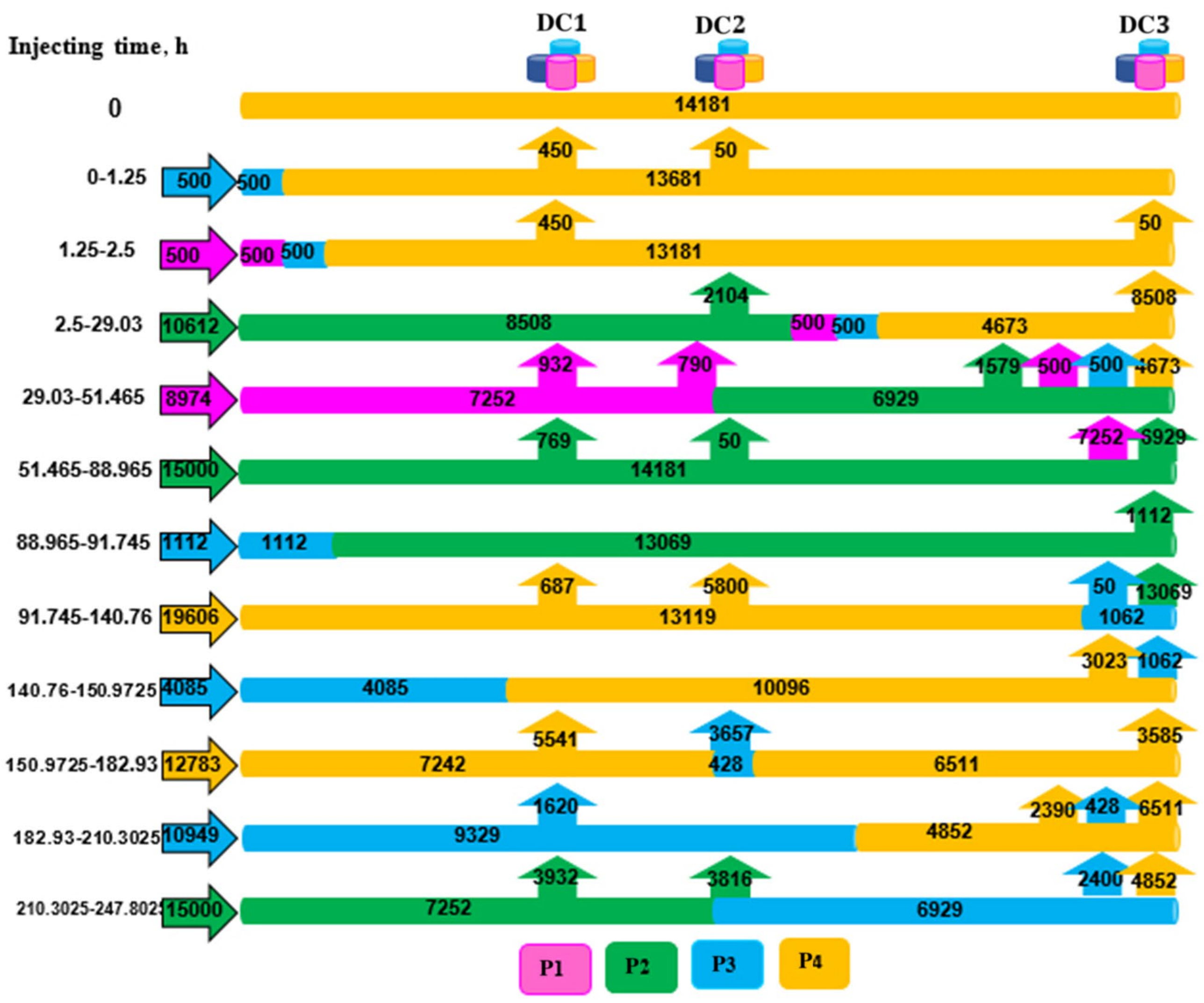

Fig. 3 Optimal scheduling for the real-world data case

Table 8 Comparing the improved cuts with conventional cuts and direct method

Approach	Running time, s	Number of iterations	Relative gap, %
Proposed algorithm with cuts (51) and (52)	325	10	0
Proposed algorithm with conventional cut (58)	6226	26	0
Direct method (CPLEX)	7200	–	54

$\rightarrow$P4 (denoted by $\tilde{\delta}^{(2)}$) is obtained. The optimal solution of the SP ($\tilde{\delta}^{(2)}$) implies that the products P1 and P4 face shortage. Since $\left|\mathbb{I}^{(2)}_{\mathrm{P1},after}\right| \leq \left|\mathbb{I}^{(2)}_{\mathrm{P4},after}\right|$, a feasibility cut is generated and added to the MP with respect to $\mathbb{I}^{(2)}_{\mathrm{P1},after}$. By this moment, we have $\mathbb{L} = \{1, 2\}$ and $IND_1 = IND_2 = 1$.

After four iterations, the first feasible sequence 0$\rightarrow$P4$\rightarrow$P3$\rightarrow$P4$\rightarrow$P3$\rightarrow$P2$\rightarrow$P3$\rightarrow$P2$\rightarrow$P1$\rightarrow$P2$\rightarrow$P3$\rightarrow$P4 (denoted by $\tilde{\delta}^{(4)}$) is obtained. The optimal solution of the SP($\tilde{\delta}^{(4)}$) indicates that a volume equal to 27,497 vu of batches is injected at the maximum pumping rate and batch 3 is the first batch injected at the maximum pumping rate). Since this volume is greater than $\overline{elb}$, the fitness condition is not established and an improvement cut is generated over the last ten batches and added to the MP. By this moment, we have $\mathbb{L} = \{1, 2, 3, 4\}$, $IND_1 = IND_2 = IND_3 = 1$ and $IND_4 = 0$. The same process is repeated, and in the tenth iteration, the fitness condition is established. Table 6 summarizes the information in each iteration and indicates that the algorithm could find the optimal solution within 325 s.

The sequence obtained in each iteration is shown in Table 7. The feasibility cuts are generated at iterations 1, 2, 3, 5, 7 and 8. However, the improvement cuts are produced at iterations 4, 6 and 9. The sequence P2$\rightarrow$P3$\rightarrow$P4$\rightarrow$P3$\rightarrow$P4$\rightarrow$P3$\rightarrow$P2$\rightarrow$P1$\rightarrow$P2$\rightarrow$P1$\rightarrow$P3$\rightarrow$P4 is the optimal sequence having a further non-empty batch, compared with the previous sequences. The schedule corresponding to the optimal sequence is shown in Fig. 3.

As mentioned earlier, conventional cut (58) omits only the current sequence from the MP solution space, while improved cut (51) may remove several infeasible sequences, in addition to the current sequence. Thus, the number of iterations as well as the running time of the algorithm is reduced. For example, by using conventional cut (58) instead of cuts (51) and (52) in the above instance, the number of iterations and the running time increase to 26 and 6226 s, respectively. In addition, if the model LMPSP is solved directly by the solver CPLEX, after 2 h, a solution is achieved with 54% relative gap. Therefore, as shown in Table 8, the performance of the proposed algorithm with improved cuts will be much better than that of the conventional cuts and direct method.

4.2 Implementation of the proposed algorithm on randomly generated instances

In the previous real-world data case, assume that the initial inventory and total demands of each DC are given in Table 9. For this sample, the pipeline system has to inject a fraction of total volume at the maximum pumping rate and the parameter $\overline{elb}$ is calculated as 3155 vu. Moreover, the lower bounds of the number of batches that should be injected are equal to $\alpha_{\mathrm{P1}} = 2$, $\alpha_{\mathrm{P2}} = 3$, $\alpha_{\mathrm{P3}} = 2$ and $\alpha_{\mathrm{P4}} = 4$.

For this instance, the proposed algorithm reaches an optimal solution by considering 16 batches after 21 iterations and 832 s. As shown in Table 10, the first fifteen sequences are infeasible, and the first feasible sequence is sequence 16. The volume injected at the maximum pumping rate for this sequence is more than the expected value $\overline{elb}$, and therefore, the fitness condition is not established. However, after five more iterations, the fitness condition is established and the best found solution (which is optimal as well) is returned.

In this instance, if conventional combinatorial cuts are used instead of the proposed feasibility and improvement cuts (51) and (52), after 2 h, the algorithm provides a solution with 23% relative gap. If the parameter $\overline{elb}$ is considered as 5000 vu, a suboptimal solution is obtained at the iteration 16. However, if the parameter $\overline{elb}$ is considered less than 3155 vu (e.g., 2500 vu), the fitness condition is not established at the iteration 21; thus, the algorithm continues for up to 2 h, and then, sequence 21 (similar to the obtained optimal solution for $\overline{elb} = 3155$) is introduced as the best found solution.

To further evaluate the performance of the proposed algorithm, a number of random samples are generated, and the running time and quality of the obtained solutions are compared with previous approaches in the literature. Several parameters remain unchanged in the pipeline scheduling problem at different time horizons; however, other parameters change at every time horizon. For example, the capacity of storage tanks is constant in any time period; nevertheless, daily demands and initial inventories change at different time horizons. Similar to a previous study (MirHassani et al. 2011), 100 samples are randomly generated, and the algorithm is applied for each one. In all the cases, the algorithm reaches the optimal solution in less than ten minutes. The

Table 9 A random sample for the initial inventory and the total demand

	Initial inventory, vu			Total demand, vu		
	*n*1	*n*2	*n*3	*n*1	*n*2	*n*3
P1	538	541	902	3650	3103	6126
P2	1345	1432	3273	7617	8244	31,696
P3	3152	3021	6898	6686	11,929	18,815
P4	3678	4333	7130	12,650	13,803	58,382

Table 10 Sequences obtained in throughout the algorithm for a random instance

Iteration	Sequence																Status of sequence
	*i*16	*i*15	*i*14	*i*13	*i*12	*i*11	*i*10	*i*9	*i*8	*i*7	*i*6	*i*5	*i*4	*i*3	*i*2	*i*1	
1	P2	P3	P4	P3	P4	P3	P4	P3	P1	P2	P1	P3	P1	P2	P3	P4	Infeasible
2	P1	P3	P4	P3	P4	P3	P4	P3	P2	P1	P2	P3	P2	P1	P3	P4	Infeasible
3	P3	P2	P3	P1	P2	P3	P4	P3	P4	P3	P4	P3	P1	P2	P3	P4	Infeasible
4	P1	P2	P3	P1	P3	P2	P3	P4	P3	P2	P3	P4	P3	P4	P3	P4	Infeasible
5	P3	P4	P3	P4	P3	P2	P1	P3	P4	P3	P1	P2	P3	P2	P3	P4	Infeasible
6	P3	P4	P3	P4	P3	P4	P3	P1	P2	P3	P2	P3	P2	P1	P3	P4	Infeasible
7	P3	P4	P3	P4	P3	P2	P3	P1	P2	P3	P4	P3	P2	P1	P3	P4	Infeasible
8	P2	P1	P3	P4	P3	P4	P3	P2	P3	P1	P2	P1	P3	P4	P3	P4	Infeasible
9	P3	P4	P3	P4	P3	P2	P3	P4	P3	P2	P1	P2	P3	P1	P3	P4	Infeasible
10	P4	P3	P4	P3	P4	P3	P4	P3	P1	P2	P1	P2	P1	P2	P3	P4	Infeasible
11	P4	P3	P4	P3	P4	P3	P1	P2	P1	P2	P1	P2	P3	P1	P3	P4	Infeasible
12	P4	P3	P1	P3	P4	P3	P4	P3	P1	P2	P1	P2	P1	P2	P3	P4	Infeasible
13	P4	P3	P2	P3	P4	P3	P1	P2	P1	P2	P1	P2	P3	P4	P3	P4	Infeasible
14	P1	P3	P4	P3	P4	P3	P1	P2	P3	P4	P3	P2	P1	P2	P3	P4	Infeasible
15	P2	P3	P4	P3	P4	P3	P2	P3	P4	P3	P2	P1	P3	P1	P3	P4	Infeasible
16	P4	P3	P2	P1	P2	P3	P4	P3	P4	P3	P4	P3	P2	P1	P3	P4	Feasible
17	P2	P3	P4	P3	P4	P3	P1	P2	P1	P3	P2	P1	P3	P4	P3	P4	Infeasible
18	P1	P3	P4	P3	P4	P3	P2	P3	P4	P3	P2	P3	P2	P1	P3	P4	Infeasible
19	P4	P3	P4	P3	P2	P1	P2	P3	P4	P3	P4	P3	P1	P2	P3	P4	Feasible
20	P3	P4	P3	P2	P3	P4	P3	P4	P3	P2	P3	P2	P1	P2	P3	P4	Infeasible
21	P4	P3	P4	P3	P2	P1	P2	P3	P4	P3	P4	P3	P2	P1	P3	P4	Optimal

Table 11 Comparing the proposed algorithm with other approaches

Approach	Average running time, s	Average gap, %
Proposed algorithm with cuts (51) and (52)	413	0
Proposed algorithm with conventional cut (58)	5878	7
Heuristic algorithm proposed by MirHassani et al. (2011)	6236	16
Direct method (CPLEX)	> 7200	87

average number of iterations over 100 samples is 13. In addition, the average and standard deviation of the running time are 485 and 96 s, respectively. However, achieving the optimal schedule in a short time is impossible in all previous heuristics methods for long-term scheduling. For example, the algorithm proposed by MirHassani et al. (2011) spends an average of 6236 s to reach a suboptimal solution in 30-day planning. The performance of the proposed algorithm with improved combinatorial cuts, the algorithm with conventional combinatorial cuts, the heuristic algorithm proposed by MirHassani et al. (2011) and the direct method for 15 samples of random data is presented in Table 11. The results show that the proposed decomposition-based algorithm is significantly better than the previous methods presented in the literature in terms of the solution quality and computation time.

5 Conclusions

In this paper, the scheduling and inventory management of a straight pipeline system connecting a single refinery to multiple DCs is addressed. The model proposed by MirHassani et al. (2011) is considered as a base, and a novel decomposition-based heuristic method is developed to solve it. The idea behind our cut generation method is novel and is based on the knowledge of the underlying problem structure. The proposed algorithm is tested on a real-world system and different randomly generated instances. The results confirm that the proposed decomposition-based algorithm is significantly better than the previous methods presented in the literature in terms of the solution quality and computation time.

Timely supply of demands is the priority of planners, and adequate resources are usually considered to achieve this purpose. In other words, a program without any delay is preferred to a plan with less pumping cost. However, in rare cases, delay is inevitable and every sequence leads to a shortage of some products. In such cases, the system is unable to meet demands on time and one or more DCs would face a shortage of products. The extension of the proposed algorithm to deal with this situation is suggested for future work. Additionally, it would be valuable to extend the proposed algorithm to deal with more complex pipeline systems with, for example, tree or network structures.

References

Akpinar S, Elmi A, Bektaş T. Combinatorial Benders cuts for assembly line balancing problems with setups. Eur J Oper Res. 2017;259(2):527–37. https://doi.org/10.1016/j.ejor.2016.11.001.

Bai L, Rubin PA. Combinatorial Benders cuts for the minimum tollbooth problem. Oper Res. 2009;57(6):1510–22. https://doi.org/10.1287/opre.1090.0694.

Beheshti Asl N, MirHassani SA. Accelerating Benders decomposition: multiple cuts via multiple solutions. J Comb Optim. 2018;37(3):806–26. https://doi.org/10.1007/s10878-018-0320-8.

Benders JF. Partitioning procedures for solving mixed-variables programming problems. Numer Math. 1962;4(1):238–52. https://doi.org/10.1007/BF01386316.

Bisschop J. AIMMS-optimization modeling. Harlem: Paragon Decision Technology. 2012. http://www.aimms.com.

Cafaro DC, Cerdá J. Optimal scheduling of refined products pipelines with multiple sources. Ind Eng Chem Res. 2009;48(14):6675–89. https://doi.org/10.1021/ie900015b.

Cafaro DC, Cerda J. Rigorous formulation for the scheduling of reversible-flow multiproduct pipelines. Comput Chem Eng. 2014;61:59–76. https://doi.org/10.1016/j.compchemeng.2013.10.006.

Cafaro VG, Cafaro DC, Mendes CA, Cerda J. Optimization model for the detailed scheduling of multi-source pipelines. Comput Ind Eng. 2015;88:395–409. https://doi.org/10.1016/j.cie.2015.07.022.

Chen JH, Lee DH, Cao JX. A combinatorial Benders' cuts algorithm for the quayside operation problem at container terminals. Transp Res E. 2012;48(1):266–75. https://doi.org/10.1016/j.tre.2011.06.004.

Codato G, Fischetti M. Combinatorial Benders' cuts for mixed-integer linear programming. Oper Res. 2006;54(4):756–66. https://doi.org/10.1287/opre.1060.0286.

Fabro JA, Stebel SL, Rossato D, Polli HL, Arruda LV, Neves FJ, et al. A MILP decomposition solution to the scheduling of heavy oil derivatives in a real-world pipeline. Comput Chem Eng. 2014;66:124–38. https://doi.org/10.1016/j.compchemeng.2014.01.004.

Hooker JN. Planning and scheduling by logic-based Benders decomposition. Oper Res. 2007;55(3):588–602. https://doi.org/10.1287/opre.1060.0371.

Hooshmand F, MirHassani SA. An effective bilevel programming approach for the evasive flow capturing location problem. Netw Spat Econom. 2018. https://doi.org/10.1007/s11067-018-9415-0.

ILOG. ILOG CPLEX 12.4 User's manual. IBM. http://www.ilog.com/products/cplex; 2011.

Kirschstein T. Planning of multi-product pipelines by economic lot scheduling models. Eur J Oper Res. 2018;264(1):327–39. https://doi.org/10.1016/j.ejor.2017.06.014.

Magatão L, Arruda LV, Neves-Jr F. A combined CLP-MILP approach for scheduling commodities in a pipeline. J Sched. 2011;14(1):57–87. https://doi.org/10.1007/s10951-010-0186-9.

Magatão SN, Magatão L, Neves-Jr F, Arruda LV. Novel MILP decomposition approach for scheduling product distribution through a pipeline network. Ind Eng Chem Res. 2015;54(18):5077–95. https://doi.org/10.1021/ie5046796.

Maravelias CT. A decomposition framework for the scheduling of single- and multi-stage processes. Comput Chem Eng. 2006;30(3):407–20. https://doi.org/10.1016/j.compchemeng.2005.09.011.

MirHassani SA, BeheshtiAsl N. A heuristic batch sequencing for multiproduct pipelines. Comput Chem Eng. 2013;56:58–67. https://doi.org/10.1016/j.compchemeng.2013.05.007.

MirHassani SA, Fani Jahromi H. Scheduling multi-product tree-structure pipelines. Comput Chem Eng. 2011;35(1):165–76. https://doi.org/10.1016/j.compchemeng.2010.03.018.

MirHassani SA, Ghorbanalizadeh M. The multi-product pipeline scheduling system. Comput Math Appl. 2008;56(4):891–7. https://doi.org/10.1016/j.camwa.2008.01.035.

MirHassani SA, Moradi S, Taghinezhad N. Algorithm for long-term scheduling of multi-product pipelines. Ind Eng Chem Res. 2011;50(24):13899–910. https://doi.org/10.1021/ie200101a.

MirHassani SA, Abbasi M, Moradi S. Operational scheduling of refined product pipeline with dual purpose depots. Appl Math Model. 2013;37(8):5723–42. https://doi.org/10.1016/j.apm.2012.11.009.

Moradi S, MirHassani SA. Transportation planning of petroleum products and integrated inventory management. Appl Math Model. 2015;39(23–24):7630–42. https://doi.org/10.1016/j.apm.2015.04.023.

Moradi S, MirHassani SA. Robust scheduling for multi-product pipelines under demand uncertainty. Int J Adv Manuf Technol. 2016;87(9–12):2541–9. https://doi.org/10.1007/s00170-016-8561-0.

Mostafaei H, Castro PM, Ghaffari-Hadigheh A. A novel monolithic MILP framework for lot-sizing and scheduling of multiproduct treelike pipeline networks. Ind Eng Chem Res. 2015;54(37):9202–21. https://doi.org/10.1021/acs.iecr.5b01440.

Mostafaei H, Castro PM, Ghaffari-Hadigheh A. Short-term scheduling of multiple source pipelines with simultaneous injections and deliveries. Comput Oper Res. 2016;73:27–42. https://doi.org/10.1016/j.cor.2016.03.006.

Rahmaniani R, Crainic TG, Gendreau M, Rei W. The Benders decomposition algorithm: a literature review. Eur J Oper Res. 2017;259(3):801–17. https://doi.org/10.1016/j.ejor.2016.12.005.

Rejowski RJ, Pinto JM. Efficient MILP formulations and valid cuts for multiproduct pipeline scheduling. Comput Chem Eng. 2004;28(8):1511–28. https://doi.org/10.1016/j.compchemeng.2003.12.001.

Relvas S, Barbosa-Póvoa AP, Matos HA. Heuristic batch sequencing on a multiproduct oil distribution system. Comput Chem Eng. 2009;33(3):712–30. https://doi.org/10.1016/j.compchemeng.2008.10.012.

Relvas S, Magatão SN, Barbosa-Póvoa AP, Neves F. Integrated scheduling and inventory management of an oil products distribution system. Omega. 2013;41(6):955–68. https://doi.org/10.1016/j.omega.2013.01.001.

Verstichel J, Kinable J, De Causmaecker P, Vanden Berghe G. A combinatorial Benders' decomposition for the lock scheduling problem. Comput Oper Res. 2015;54:117–28. https://doi.org/10.1016/j.cor.2014.09.007.

Zhang HR, Liang YT, Xiao Q, Wu MY, Shao Q. Supply-based optimal scheduling of oil product pipelines. Pet Sci. 2016;13(2):355–67. https://doi.org/10.1007/s12182-016-0081-x.