Computers and Chemical Engineering 34 (2010) 1687–1704

Contents lists available at ScienceDirect

Computers and Chemical Engineering

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

Operational scheduling of refined products pipeline networks with simultaneous batch injections

Diego C. Cafaro, Jaime Cerdá*

INTEC (UNL – CONICET), Güemes 3450, 3000 Santa Fe, Argentina

ARTICLE INFO

Article history:
Received 31 August 2009
Received in revised form 23 February 2010
Accepted 4 March 2010
Available online 16 March 2010

Keywords:
Pipeline network
Operational planning
Continuous approach
Simultaneous injections
Multiple sources

ABSTRACT

Petroleum refined products are mostly sent from oil refineries to distribution depots by trunk pipelines. Pipeline networks usually involve multiple input and exit terminals, and even dual-purpose stations. Several pumping operations can be simultaneously performed at different sources. Most of the computational burden on the scheduling of multi-source pipeline networks comes from three operational tasks: pump sequencing, batch sizing, and batch allocation. Previous contributions applied discrete decomposition approaches performing such tasks through heuristic-based decisions. This paper introduces an MILP continuous formulation for the operational scheduling of unidirectional pipeline networks that allows simultaneous batch injections. The problem goal is to satisfy depot requirements at minimum total cost. The optimal schedule of pumping and delivery operations is established all at once. Results show that simultaneous batch injections lead to a better use of the pipeline transport capacity and a substantial reduction on the overall time needed to meet depot demands.

1. Introduction

Between oil refineries, where a variety of refined products such as heating oil, motor gasoline, jet fuel, and liquefied gas are produced, and the end consumers there is a distribution network consisting of trunk pipelines, rail, road tankers and coastal vessels carrying such finished products to several distribution centers. In these facilities, large storage tanks with significant amounts of refined products and several loading bays for road tankers are available to deliver them to consumer markets. About two-thirds of all petroleum products in the US are carried by pipelines because they represent the safest and least expensive mode of transportation. Pipeline networks have several entry and exit points and lots of products usually travel through several pipelines before reaching their final destinations. The aim of pipeline logistics is to ensure that the right product will be available by the customer at the right time, at the right depot and at the lowest cost, under the best possible conditions of safety, security and respect for the environment. Planning the injection of new batches in pipelines and the simultaneous product deliveries to depots is a very difficult task with many operational constraints to be satisfied. Efficient supporting tools are then required for an optimal planning of pipeline operations.

Most contributions on short-term operational planning of refined products pipelines have been focused on rather simple pipeline systems. They usually considered single-source, unidirectional pipelines connecting a single origin to multiple distribution depots. Moreover, it is generally assumed a one-period planning horizon with product demands due before the horizon end, i.e. a unique delivery due-date. Different types of approaches were proposed to study single-source pipeline scheduling problems, including rigorous optimization models, knowledge-based heuristic techniques (Sasikumar, Prakash, Patil, & Ramani, 1997), discrete-event simulation tools (García-Sánchez, Arreche, & Ortega-Mier, 2008; Mori et al., 2007), and decomposition frameworks (Hane & Ratliff, 1995). Rigorous optimization methods generally consist of solving a single MILP or MINLP mathematical model and are usually grouped into two classes: discrete and continuous, depending on the way volume and time domains are handled. Discrete MILP-formulations divide both the pipeline volume into a significant number of single-product packs, and the planning horizon into time intervals of equal and fixed duration (Magatão, Arruda, & Neves-Jr, 2004; Rejowski & Pinto, 2003, 2004; Zyngier & Kelly, 2009). As a result, flow-rate variations due to changes in pipeline diameter cannot be handled. Since discrete models indeed stand for approximate problem representations, they will not provide feasible schedules unless a high discretization level is adopted. Consequently, large-size discrete formulations are to be tackled even for rather short time horizons. Rejowski and Pinto (2008) introduced an improved continuous-time MINLP formulation that accounts for flow-rate variations originated by the smaller diameter of downstream pipeline segments. In addition, it considers different booster configurations with distinct number of

* Corresponding author. Tel.: +54 342 4559175; fax: +54 342 4550944.
E-mail address: jcerda@intec.unl.edu.ar (J. Cerdá).

0098-1354/$ – see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compchemeng.2010.03.005

Nomenclature

(a) Sets

K	chronologically ordered blocks of pumping runs
I	product batches ($I = I^{old} \cup I^{new}$)
I^{new}	new batches to be injected during the planning horizon
I^{old}	old batches in the initial linefill
P	refined petroleum products
P_i	refined petroleum product contained in old batch $i \in I^{old}$
S	pipeline input terminals
J	pipeline output terminals
J_p	distribution depots demanding product p
$J_{p,s}$	distribution depots demanding product p to be supplied by an upstream source s

(b) Parameters

$cb_{p,j}$	unit backorder penalty cost for tardy orders of product p at output terminal j
$cif_{p,p'}$	total reprocessing cost of interface material involving products p and p'
$cin_{p,s}$	pumping cost per unit of product p at input terminal s
D_{min}, D_{max}	minimum/maximum delivery size to output terminals
$DL_{p,j}$	minimum request of product p at the output terminal j
$DU_{p,j}$	maximum amount of product p that can be delivered to output terminal j
h_{max}	planning horizon length
PV	total pipeline volume from the origin to the farthest depot
Q_{min}, Q_{max}	minimum/maximum batch injection size
$Q_{max,p}$	maximum batch injection size for product p
$SL_{p,s}$	lowest amount of product p to be shipped from input terminal s
$SU_{p,s}$	maximum amount of product p that can be shipped from input terminal s
$vb_{min,s}$, $vb_{max,s}$	minimum/maximum product flow-rates at source s
W_i^o	initial volume of old batch i
σ_j	volume coordinate of output terminal j from the pipeline origin
τ_s	volume coordinate of input terminal s from the pipeline origin

(c) Continuous variables

BC	total backorder penalty cost for tardily meet product demands
$B_{p,j}$	backorder of product p at the output terminal j
C_k, L_k	completion time/length of run block k (measured in time units)
$D_{i,j}^{(k)}$	volume of batch i diverted to output terminal j during run block k
$DP_{i,j,p}^{(k)}$	volume of product p diverted from batch i to terminal j during run block k
$F_{i,k}$	upper coordinate of batch i from the origin at time C_k
L_k	length of run block k
$L_{k,s}$	length of pumping run k taking place at source s
PC_k	total pumping cost during block k
$Q_{i,s}^{(k)}$	size of batch i shipped from input terminal s during run k
$QP_{i,s,p}^{(k)}$	size of batch i containing product p inputted from input terminal s through run k
TC_i	total reprocessing cost for the interface between batch i and the following $(i+1)$
$W_{i,k}$	size of batch i at the end of run k

(d) Binary variables

$v_{i,s}^{(k)}$	variable denoting that a portion of batch i is injected from source s through block k
$x_{i,j}^{(k)}$	variable denoting that a portion of batch i is diverted to the depot j during block k
$y_{i,p}$	variable denoting that batch i contains product p

pumping stages, unit pumping costs and yield-rate curves to choose the one minimizing operating costs. However, the pipeline volume is still partitioned into many packs with each one containing only one product. When compared with the discrete-time representation of Rejowski and Pinto (2004), it was found that the proposed MINLP model always achieved better pipeline schedules.

1.1. Continuous approaches for single-source pipelines

On the other hand, a continuous MILP-formulation in both time and volume was first proposed by Cafaro and Cerdá (2004) for the operational planning of pipeline systems with a single origin and multiple depots. The approach permits to determine the optimal sequence of batch injections, lot sizes, pump rates, start/end times of pumping runs, interface volumes to be reprocessed, as well as amounts and types of products delivered to distribution depots during every run. Flow-rate variations due to changes in pipeline diameter are easily handled. Moreover, short pumping operations injecting small plugs to separate batches of incompatible products, if necessary, are inserted by the model to reduce product contamination. In other words, the whole set of pumping and delivery operations is established all at once. In addition, the model is able to track the location and size of product lots over the planning horizon, maintain product inventory levels in refinery and depot tanks within allowable ranges and account for high-pumping cost intervals. Another MILP continuous representation is due to Relvas, Matos, Barbosa-Póvoa, Fialho, and Pinheiro (2006) who studied the scheduling of a single pipeline transporting a variety of oil derivatives from one refinery to a unique distribution center over a multiperiod, monthly horizon. This contribution has focused on the end of the pipe terminal to consider quality control procedures that may significantly affect the inventory management at the tank farm. Besides, the model is able to satisfy daily demands at the depot, i.e. multiple due dates for each product. However, the computational performance of the approach badly deteriorates when the complete sequence of batch injections is to be selected. A similar pipeline scheduling problem was studied by Cafaro and Cerdá (2008a) who developed a smaller MILP formulation providing better schedules in a much shorter computational time. Improved results were obtained by optimizing the complete sequence of pumping runs. Relvas, Matos, Barbosa-Póvoa, and Fialho (2007) extended their previous MILP model to account for variable flow-rates and pipeline stoppages and, in addition, they developed a reactive scheduling framework for updating the operational planning when unexpected events occur.

Another important feature of pipeline scheduling problems is the fact that pumping operations taking place at the end of the time horizon have the only purpose of pushing the prior injected lots towards the assigned destinations, keeping the pipeline com-

pletely filled. A large batch of a filler product is then inserted to sweep the batch sequence in the line and it occupies the entire pipeline. Model-based approaches usually choose to pump at last the most compatible fuel with the one previously injected as the filler product. As a result, the final linefill does not match at all the requested depot demands at earlier stages of the next planning horizon. To overcome such usual shortcomings of pipeline scheduling methods, Cafaro and Cerdá (2008b) extended the continuous MILP-formulation of Cafaro and Cerdá (2004) to develop the pipeline operational planning over a multiperiod rolling horizon, with product deliveries due at the end of each week. Pumping and delivery operations can overlap two or more time periods. As time goes on and the current period comes to an end, the planning horizon moves forward and a new period with further product demands is considered to keep a constant horizon length. Consequently, a rescheduling process based on updated demand data is triggered over the new time-horizon instance. Planned operations may be modified beyond the current first period due to the new demand scenario. Besides, late pumping runs are intended for both sweeping the pipeline content, and matching product requirements due at the last period. In this way, it was introduced a rescheduling framework for single-source pipeline systems with a unique origin and several destinations that can handle multiple delivery due dates.

1.2. Previous contributions on multiple-source pipeline network planning

Recently, more realistic approaches handling real-world pipeline networks with multiple origins and destinations have been developed. A new interesting feature of multiple-source pipeline networks is the fact that several pumping operations can be simultaneously performed at several input terminals. In addition, a source/receiving node can be directly connected to multiple distribution terminals through different pipelines, i.e. pipeline branching. Moreover, more than one pipeline can interconnect an input terminal with a single depot, and the flow direction in some pipelines can be reversed. As a result, there are several alternative routes (i.e. sequence of pipelines) to move a batch from a particular origin to a given destination. Despite their complex topology, simple principles are usually applied to scheduling pipeline networks. When a new lot is inserted at the inlet of a pipeline, another one with a similar volume at the other extreme is pushed either to a single adjacent downstream pipeline, or into the assigned tank of a receiving terminal. Similarly, a pipeline can receive material from at most a single source that may be an adjacent upstream pipeline or the tank farm at the pipeline inlet.

Neves et al. (2007) presented a computational framework that uses a decomposition approach for the planning of pipeline network operations over a monthly horizon. The decomposition relied on a heuristic pre-processing block that accounts for demand requirements, production planning, and typical lot sizes to determine a candidate set of pumping run sequences. In addition, the heuristic block provides time windows within which pump and strip operations should be performed. Afterwards, the pre-processed information is used by a continuous-time MILP formulation to determine the exact start/finish times of batch injections and deliveries at every pipeline terminal. Since seasonal costs of the electric energy are considered, the model includes binary variables just to avoid pumping operations during high-energy cost periods. Mori et al. (2007) developed a discrete-event simulation model for a detailed study of planned operational activities in real-world pipeline networks. The proposed simulator was used in combination with a short-term optimization package providing the pipeline schedule to be tested. The simulation tool allows visualizing conflicts in pipeline allocation to competing batches, movement of batches throughout the pipelines, start/end times of stripping operations during pumping runs, and the management of tank inventories at source and destination terminals. It also reports some performance measures such as pipeline utilization rate and throughput. Boschetto et al. (2008) reformulated the hybrid approach of Neves et al. (2007) using a different decomposition strategy now involving three blocks: (i) a resource-allocation block determining candidate sequences of batch injections, (ii) a pre-analysis block specifying the precise volumes to be either pumped from source nodes or received in destination nodes, and providing the earliest start/finish times for stripping operations in every destination node, and (iii) a continuous-time MILP model determining the exact timing of pump and delivery operations at each node. Another hybrid approach that combines a randomized constructive heuristic with novel constraint programming (CP) models was reported by Moura, de Souza, Cire, and Lopez (2008). It comprises two phases. The planning phase uses heuristics to create the set of batches to be injected (delivery orders), by specifying their volume, origin, destination depot, product type, assigned route and delivery due-date. The scheduling phase takes the set of delivery orders generated at the planning phase and determines the sequence and start times of pumping operations at every source node. This second phase was implemented through a pair of CP models. A first CP model provides the pump sequence of delivery orders at each input node, and the time intervals for the start of the corresponding pump operations. Afterwards, a simpler second CP model determines the set of pump operations for each delivery order as well as their exact start times. The pumping of a delivery order can be interrupted and resumed at a later time to either allow the shipping of more urgent products through a common pipeline segment, or to avoid high-energy-cost periods. The decomposition approaches previously described (Boschetto et al., 2008; Moura et al., 2008; Neves et al., 2007) were all applied to the scheduling of real-world pipeline networks transporting a substantial number of refined products from several sources to multiple destinations over a one-month horizon. Structural features like the interconnection of a pair of nodes by multiple pipelines, dual-purpose terminals receiving and/or sending products, and bi-directional pipelines were all considered. Most of the computational burden in multiproduct pipeline scheduling comes from three formidable tasks: pumping sequencing, batch sizing, and batch allocation to receiving terminals. By heuristically choosing them, the remaining operational decisions can be taken in a short CPU time. However, the final pipeline schedule is greatly influenced by those heuristic-based decisions previously taken (Boschetto et al., 2008).

Cafaro and Cerdá (2009) developed the first MILP continuous formulation, in both time and volume, for the scheduling of a restricted family of pipeline networks with multiple origins and destinations. It is a single-level approach that efficiently determines both input and delivery schedules all at once. Given the product requirements and delivery dates at distribution terminals, the proposed MILP chooses the size, origin and destinations for each batch to be injected, the pump sequence at every source, and the start/end times of pumping and extraction operations. The scheduling task in multi-source pipeline systems involves additional challenges efficiently handled by the model. Pumping runs at intermediate locations can either insert a new lot or increase the size of a batch in transit. As a result, batches traveling in a particular pipeline are no longer chronologically arranged. In fact, a batch can be preceded by another lot that has been later injected. To deal with this scheduling challenge, a batch to be inserted at a downstream source is booked by the model, traveling as an empty lot up to the assigned pumping station. However, the model assumes that a single pumping operation can at most be performed at any time. Moreover, the approach of Cafaro and Cerdá (2009) is restricted to pipeline networks involv-

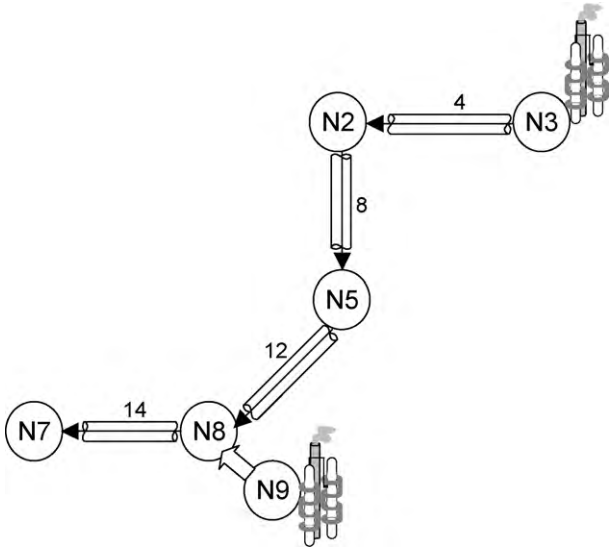

Fig. 1. A refined products pipeline network (Boschetto et al., 2008).

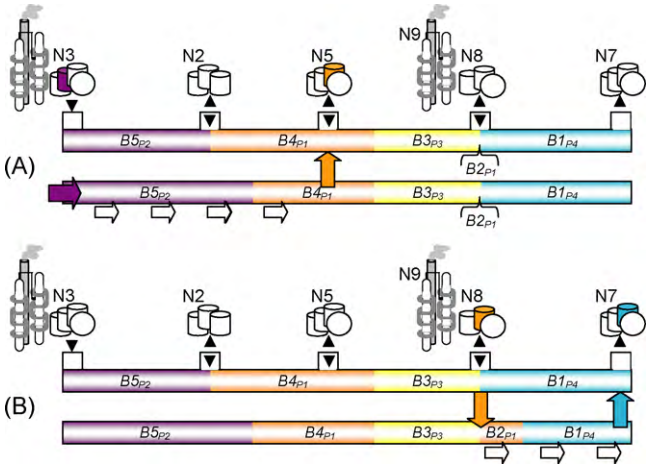

Fig. 3. An operational planning involving a sequence of two batch injections.

ing unidirectional pipes and a single pipeline between every pair of nodes in the network.

This paper introduces an improved MILP formulation for the operational planning of multi-source, unidirectional pipeline networks that allows simultaneous batch injections at several input terminals. In this manner, the overall time needed to meet the specified product demands at distribution depots is substantially decreased by making a better use of the pipeline transport capacity. As remarked by Moura et al. (2008), some conflicting pumping operations cannot be overlapped in time. By incorporating the so-called non-interacting pumping run conditions, the model is able to automatically disallow the overlapping of conflicting batch injections. Three illustrative examples have been solved to show that the new MILP formulation provides much better pipeline operational schedules.

2. Motivating example

Let us consider a motivating example involving a portion of the Brazilian pipeline network studied by Boschetto et al. (2008). As shown in Fig. 1, a set of four unidirectional pipelines identified by the code numbers *4*, *8*, *12*, and *14* are connecting the adjacent pairs of nodes *N3–N2*, *N2–N5*, *N5–N8* and *N8–N7*, respectively. Node *N3* stands for an oil refinery facility injecting several refined products into the line while *N2*,*N5* represent dual-purpose terminals that either pump batches to meet downstream product requirements or receive material into their storage tanks by stripping batches coming from upstream pipelines. A third alternative is the direct transfer of material (tightlining) between a pair of adjacent pipelines. In addition, terminal *N8* collects the production of oil derivatives from refinery *N9* to shipping them to the final destination *N7*. Node *N8* can also be the destination for product lots pumped at upstream sources.

The pipeline system depicted in Fig. 1 can be viewed as a trunk pipeline network with multiple input and output terminals (see Fig. 2). Some of the terminals have a dual purpose and both strip and inject operations can be executed. Let us assume that the initial linefill comprises a sequence of four lots (*B5*, *B4*, *B3*, *B1*) containing the refined products (*P2*, *P1*, *P3*, *P4*), respectively. In addition, there is an empty lot *B2* on the interface *B3–B1* that has been reserved for a future injection of product *P1* at the mid-line location *N8* (Cafaro & Cerdá, 2009).

The pipeline operator has been instructed to supply product *P1* to node *N5* by stripping lot *B4*, and product *P4* from lot *B1* to destination *N7*. To do that, he knows that some batch injections are to be performed at the input terminals *N3* and *N8*, respectively. Since there is an initial inventory of *P2* in refinery *N3*, the operator plans to inject more *P2* in the existing batch *B5* to accomplish the delivery of *P1* to terminal *N5*. Besides, he will pump a batch of *P1* initially available in the tank farm of node *N8* for delivering lot *B1* to node *N7*. It will be assumed a fixed pump rate for all batch injections and the delivery of similar volumes of products *P1* and *P4* to destinations *N5* and *N7*, respectively. If pumping operations should be performed one by one, the operator will first inject *P2* into lot *B5*, and simultaneously deliver *P1* to *N5* (Operation A in Fig. 3). Next, he will insert a batch of *P1* at terminal *N8* and deliver *P4* to *N7* (Operation B in Fig. 3).

In Fig. 4, both pumping runs (A) and (B), and the prescribed product deliveries, are simultaneously performed. The transportation activity is confined within two isolated line sections, pipelines # *4* and *8* and pipeline # *14*, both separated by an idle pipeline # *12*. Each active line section is receiving a lot of product from a single source: *N3–N5* from node *N3* and *N8–N7* from *N8*. No material from *N3–N5* reaches node *N8*. As a result, batch *B1* is just pushed forward towards its destination *N7* by the injection of *B2* from *N8*. It is said

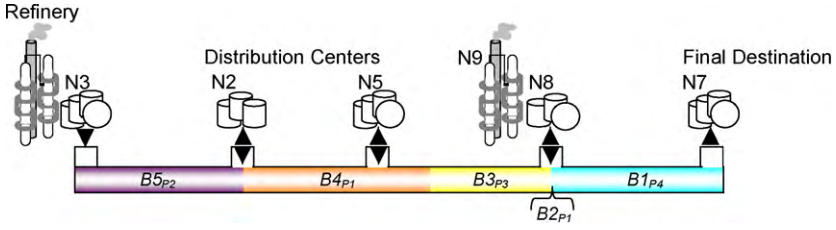

Fig. 2. Initial linefill at the motivating example.

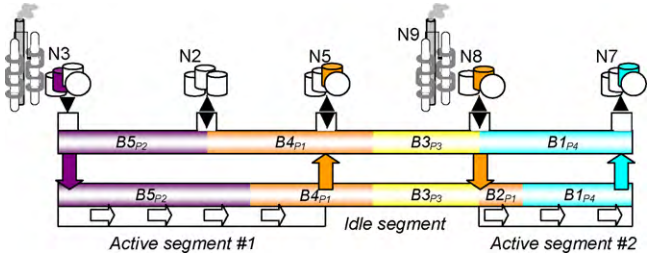

Fig. 4. Simultaneous pumping runs at multiple input terminals.

that there is no interaction between the two pumping runs, and the product flow-rate in pipeline # *14* is set by the injection rate of batch *B2* from *N8*. Then, such simultaneous batch injections constitute a feasible pipeline operation. Moreover, the amount of time required to perform them has been reduced by half by running both at the same time.

3. Model assumptions

The proposed approach is restricted to pipeline networks with unidirectional pipes and a single pipeline between each adjacent pair of terminals. However, dual-purpose terminals can be handled. To develop the problem formulation, the following assumptions have been made:

(1) A multi-source pipeline network with unidirectional pipelines is considered.
(2) There is at most a single pipeline between any pair of nodes in the network.
(3) The pipeline remains completely full of incompressible refined products at any time.
(4) Consecutive batches move along the pipelines with no physical barrier separating them.
(5) The "transmix" or interface loss between each pair of products is a known constant regardless of the pump rate, travel distance and stoppage time.
(6) Non-interacting batch injections from different pumping terminals can be simultaneously performed.
(7) During a pumping run, a pipeline can receive material from either an adjacent pipeline or the tank farm at the pipeline origin, but not both.
(8) Product demands at output terminals due before the horizon end are deterministic data.
(9) Product inventories available in input/output terminals and the initial linefill are also known.
(10) The injection rate may change with the source within the allowable range.
(11) There is no incoming product flow to the tank farm of any input terminal $s \in S$ over the planning horizon.

Though one can ignore assumption (11) by simply considering the tank farm inventory constraints at input terminals (see Cafaro & Cerdá, 2004), it was included to make the problem formulation simpler.

4. Mathematical formulation

Similar to the formulation of Cafaro and Cerdá (2009), the proposed mathematical model for the operational planning of multiple-source pipeline networks with parallel pumping runs involves five major sets: pumping run blocks (K), batches (I), oil derivatives (P), source or input nodes (S) and receiving terminals (J). In pipeline systems with a single source at the origin, the sequence of pumping runs and the sequence of new batches moving through the pipeline are strictly similar. Therefore, there is a one-to-one relationship between pumping runs and new batches, and just a single set I is to be defined. When multiple sources are considered, such a coincidence between pumping run and batch sequences no longer holds and both sets K and I are to be defined. Moreover, it is necessary to introduce the set S in order to identify the pumping station where the batch injection occurs. Therefore, the problem includes two additional sets with regards to the single-source case, i.e. K and S. In contrast to the approach of Cafaro and Cerdá (2009), the elements of K now represent blocks of parallel pumping runs at different sources rather than single batch injections. Parallel runs may not necessarily start or finish at the same time. However, batch injections in block k can start only if the previous block $(k-1)$ has ended. It is said that a block of parallel runs has finished only if all of them were completed. Then, the length of a block k is the time elapsed from the earliest start to the latest completion of pumping runs belonging to k. Since the required numbers of pumping runs and batches are not precisely known before solving the problem, the values of $|K|$ and $|I|$ should be arbitrarily adopted. They should be as low as possible to decrease the problem size, but large enough to be at least equal to the ones included in the optimal pipeline schedule. A simple expression for the estimation of $|K|$ is given in Section 4.1. The best choices usually depend on both the number of refining products to be transported and the extent of the scheduling horizon.

4.1. Pumping run constraints

The set of pumping run constraints pursues the following goals: (1) sequencing blocks of parallel pumping runs; (2) allocating batches to individual pumping runs; (3) sizing batch injections; and (4) choosing lengths for blocks of parallel runs.

4.1.1. Sequencing blocks of parallel pumping runs

A block of pumping runs $k \in K$ must be started after the completion of the preceding block $(k-1)$. Let C_k denote the completion time and L_k the duration (measured in time units) of pumping-run block k. Then,

$$C_k - L_k \geq C_{k-1} \quad \forall k \in K(k > 1) \tag{1}$$

Because simultaneous batch injections are permitted, the subscript k now stands for a generic block of pumping runs that may be performed after executing $(k-1)$ prior blocks over the planning horizon. Time C_k can be regarded as the earliest time at which all parallel runs within block k have ended. For simplicity, transition times between consecutive blocks of pumping runs have been neglected.

The number of individual runs in each block is chosen by the model accounting for the non-interacting condition among parallel batch injections. To get the best operational planning, the cardinality of the set K should be at least as large as the number of active blocks in the optimal pipeline schedule. In the simplest case, every active block just contains a single element and, consequently, the number of blocks and pumping runs are equal. For this instance, Cafaro and Cerdá (2009) proposed a simple expression to guess the value of $|K|$ given by,

$$|K| = \sum_{p \in P} \left(\frac{2}{Q_{\min,p} + Q_{\max,p}} \sum_{j \in J} DL_{p,j} \right)$$

where $(Q_{\min,p}, Q_{\max,p})$ represents the admissible lot-size range for product p, and $DL_{p,j}$ stands for the total demand of product p at the output terminal j. The above equation assumes a mean lot-size equal to $0.5 \times (Q_{\min,p} + Q_{\max,p})$ for any product p to guess the

number of batch injections. By allowing simultaneous batch injections, the value of $|K|$ usually declines because some active blocks in the optimal solution comprise several parallel pumping runs. The decreasing factor rises with the number of input terminals $|S|$ because the chance of simultaneous injections grows. By postulating a reduction factor of $[2/(|S|+1)]$, a good initial value for $|K|$ can be obtained through the following expression:

$$|K| = \frac{2}{|S|+1}\sum_{p\in P}\left(\frac{2}{Q_{\min,p}+Q_{\max,p}}\sum_{j\in J}DL_{p,j}\right)$$

Besides, the completion time for any block of parallel runs should not exceed the overall length of the time horizon $h_{\max}$.

$$C_k \le h_{\max} \quad \forall k\in K \tag{2}$$

4.1.2. Allocating batches to individual pumping runs

An individual pumping run is characterized by the block $k\in K$ to which it belongs and the input terminal where it takes place. Though a block of simultaneous pumping runs is now permitted, batch injections from a particular input terminal $s\in S$ should be executed one by one, i.e. they still happen in sequence. Let us define the binary variable $v_{i,s}^{(k)}$ to denote that a pumping run belonging to block k and accomplished at the input terminal s inserts a new batch $i\in I^{new}$ (or some amount of product to an existing batch $i\in I$) in the line whenever $v_{i,s}^{(k)}=1$. Since batches can be injected from a particular source s one at a time, then

$$\sum_{i\in I} v_{i,s}^{(k)} \le 1 \quad \forall k\in K, s\in S \tag{3}$$

In case the summation $(\sum_{s\in S}\sum_{i\in I} v_{i,s}^{(k)})$ is null, then the blocks of parallel runs k, $(k+1)$, $(k+2)\ldots$ are never performed, that is they are fictitious blocks. Such a condition is imposed by Eq. (4).

$$\sum_{s\in S}\sum_{i\in I} v_{i,s}^{(k)} \le M_B\left(\sum_{s'\in S}\sum_{i'\in I} v_{i',s'}^{(k-1)}\right) \quad \forall k\in K(k>1) \tag{4}$$

M_B is an upper bound on the number of elements of any pumping-run block k, i.e. $|S|$.

4.1.3. Sizing batch injections

A batch injection may be either a new batch or an additional portion of an existing batch. Let $Q_{i,s}^{(k)}$ denote the size of a new (or a portion of an existent) batch i injected in the pipeline by the pumping run (k,s). $Q_{i,s}^{(k)}$ will be positive only if the block of pumping runs k is really performed and a new (or an additional portion to an existent) batch i is inserted from the input terminal s ($v_{i,s}^{(k)}=1$) while performing block k. Therefore,

$$Q_{\min,s}v_{i,s}^{(k)} \le Q_{i,s}^{(k)} \le Q_{\max,s}v_{i,s}^{(k)} \quad \forall i\in I, s\in S, k\in K \tag{5}$$

where $(Q_{\min,s}, Q_{\max,s})$ stand for the minimum and maximum batch sizes that can be injected through a pumping run from the input terminal s. Since a single batch can be injected from source s during the block k, the resulting volume of refined product inserted in the line by run (k,s) will be given by $\sum_{i\in I}Q_{i,s}^{(k)}$.

4.1.4. Choosing lengths for blocks of pumping runs

Let $L_{k,s}$ be the length of the pumping run (k,s). Then,

$$vb_{\min,s}L_{k,s} \le \sum_{i\in I} Q_{i,s}^{(k)} \le vb_{\max,s}L_{k,s} \quad \forall k\in K, s\in S \tag{6}$$

The interval $[vb_{\min,s};\ vb_{\max,s}]$ represents the feasible pump rate range at source s. If no pumping run from block k is executed at input terminal s, then $v_{i,s}^{(k)}=0$ and, from Eq. (6), $L_{k,s}=0$. The block of pumping runs $(k+1)$ cannot start before completing the execution of block k. Then, the length of block k will be given by: $L_k=\max_{s\in S}(L_{k,s})$ or:

$$L_k \ge L_{k,s} \quad \forall k\in K, s\in S \tag{7}$$

In other words, the longest element determines the duration of block k.

4.2. Batch tracking constraints

Batch tracking constraints permit to trace the size and location of in-transit batches over the planning horizon.

4.2.1. Tracking the batch size over time

Let $W_{i,k}$ be the size of a batch i at the completion time of the block of parallel pumping runs k, i.e. at time C_k. During the execution of block k, the volume of batch i can change for two reasons: (a) it may receive an additional amount of product from an intermediate input location s, or (b) it may divert some volume of product to depots $j\in J$.

$$W_{i,k} = W_{i,k-1} + \sum_{s\in S} Q_{i,s}^{(k)} - \sum_{j\in J} D_{i,j}^{(k)} \quad \forall i\in I, k\in K \tag{8}$$

Since unidirectional pipelines are considered, no volume of product can be delivered to depot j while executing a block of pumping runs at downstream input terminals ($\sigma_j < \tau_s$). Then,

$$D_{i,j}^{(k)} \le \sum_{s/\tau_s<\sigma_j}\sum_{i'\in I} Q_{i',s}^{(k)} \quad \forall i\in I, j\in J, k\in K \tag{9}$$

If batch i is a new lot inserted in the line while executing the block of parallel runs k, then $W_{i,k-1}$ will be zero. Otherwise, $W_{i,k-1}$ is the size of batch i at the end of the prior block $(k-1)$. In case batch i is already in the pipeline at time $t=0$, then $W_{i,k-1}=W_i^o$ for $k=1$, with W_i^o denoting the content of batch i in the initial linefill.

4.2.2. Tracking the batch location over time

Let $F_{i,k}$ denote the upper volumetric coordinate of batch i at the completion time of block k (C_k). The coordinate $F_{i,k}$ is the total volume between the pipeline system origin and the farthest extreme of batch i after running block k. In turn, $F_{i+1,k}$ represents the upper coordinate of batch $(i+1)$ immediately chasing batch i in the pipeline at the completion of block k. Then,

$$F_{i,k} - W_{i,k} = F_{i+1,k} \quad \forall i\in I, k\in K \tag{10}$$

Since a batch can only move forward when the pipeline is active, the upper coordinate of batch i at the end of two consecutive blocks $(k-1)$ and k must satisfy the following condition,

$$F_{i,k-1} \le F_{i,k} \quad \forall i\in I, k\in K \tag{11}$$

If PV stands for the total pipeline content, then the upper coordinate of any batch i traveling along the pipeline at the end of any block k must never be greater than PV.

$$F_{i,k} \le PV \quad \forall i\in I, k\in K \tag{12}$$

Besides, the lower coordinate of any batch i in transit must be non-negative.

$$F_{i,k} - W_{i,k} \ge 0 \quad \forall i\in I, k\in K \tag{13}$$

4.3. Pipeline volumetric balance

At any time, refined products pipelines remain full of products. Therefore, the overall volume of batches traveling inside the

pipeline must be equal to *PV* at the end time of any pumping-run block *k*.

$$\sum_{i\in I} W_{i,k} = PV \quad \forall k \in K \tag{14}$$

Because all products have constant densities, the total volume diverted from in-transit batches to output terminal tanks must be equal to the overall amount of material injected from one or several sources during the execution of any block *k*.

$$\sum_{i\in I}\sum_{s\in S} Q_{i,s}^{(k)} = \sum_{i\in I}\sum_{j\in J} D_{i,j}^{(k)} \quad \forall k \in K \tag{15}$$

More than a single variable $Q_{i,s}^{(k)}$ may take a positive value if block *k* includes several parallel batch injections.

4.4. Feasibility constraints for batch injections and product deliveries

When a pumping run belonging to an active block *k* takes place at some downstream source *s* to either inject a new batch *i* or enlarge a batch *i* already in the line, the proposed problem representation assumes in both cases that the material from source *s* is supplied to an existing batch *i*. If batch *i* is a new one, the batch size $W_{i,k-1}$ at the start of block *k* will be zero. If instead batch *i* has already been injected at an upstream terminal, $W_{i,k-1}$ will be positive.

4.4.1. Supplying material from an input node to an existing batch

An existing batch *i* can receive material during pumping run (*k*, *s*) only if the following two conditions are satisfied:

(a) Before starting the block of runs *k*, batch *i* has already reached the location of the input facility *s* (τ_s). Then, $F_{i,k-1}$ should never be lower than τ_s.

$$F_{i,k-1} \geq \tau_s v_{i,s}^{(k)} \quad \forall i \in I, s \in S, k \in K \tag{16}$$

(b) Before starting block *k*, the lower coordinate of batch *i* ($F_{i,k-1} - W_{i,k-1}$) has not surpassed the location of source *s* (τ_s). Then, ($F_{i,k-1} - W_{i,k-1}$) must never be greater than τ_s.

$$F_{i,k-1} - W_{i,k-1} \leq \tau_s + (PV - \tau_s)(1 - v_{i,s}^{(k)}) \quad \forall i \in I, s \in S, k \in K \tag{17}$$

From Eqs. (16) and (17), it follows that $F_{i,k-1} - W_{i,k-1} \leq \tau_s \leq F_{i,k-1}$ whenever $v_{i,s}^{(k)} = 1$. Therefore, some product is supplied from input node *s* to the existing batch *i* when executing block *k*. In case batch *i* is a new one, $W_{i,k-1} = 0$ and consequently $F_{i,k-1} \leq \tau_s \leq F_{i,k-1}$, i.e. $\tau_s = F_{i,k-1}$. To reduce the computational cost, some very small tolerance $\varepsilon > 0$ is allowed and the feasibility condition for the insertion of a new batch at an intermediate source is relaxed as follows: $F_{i,k-1} - \varepsilon \leq \tau_s \leq F_{i,k-1} + \varepsilon$.

4.4.2. Diverting material from in-transit batches to output terminals

The delivery of material from batch $i \in I$ to depot $j \in J$ during run block $k \in K$ is feasible only if the physical connection to depot *j* is reachable from batch *i*. This implies the fulfillment of the following two feasible conditions:

(a) The upper coordinate of batch *i* at the end of run block *k*($F_{i,k}$) should never be lower than the *j*th-depot coordinate σ_j, i.e. $F_{i,k} \geq \sigma_j$.
(b) The lower coordinate of batch *i* at the completion of run block ($k-1$) must never exceed the depot coordinate σ_j, i.e. $F_{i,k-1} - W_{i,k-1} \leq \sigma_j$.

Let $x_{i,j}^{(k)}$ be a binary variable denoting that the *j*th-terminal tankage is reachable from batch *i* during run block *k* ($x_{i,j}^{(k)} = 1$). Otherwise, $x_{i,j}^{(k)} = 0$ and no material can be transferred from batch *i* to depot *j* ($D_{i,j}^{(k)} = 0$) as imposed by Eq. (18).

$$D_{\min} x_{i,j}^{(k)} \leq D_{i,j}^{(k)} \leq D_{\max} x_{i,j}^{(k)} \quad \forall i \in I, j \in J, k \in K \tag{18}$$

where $D_{\max}$ is an upper bound on the amount of product that can be stripped from batch *i* and sent to depot *j*. Constraints (19)–(20) represent the feasible conditions for diverting material from in-transit lots to depots, respectively.

$$F_{i,k} \geq \sigma_j x_{i,j}^{(k)} \quad \forall i \in I, j \in J, k \in K \tag{19}$$

$$F_{i,k-1} - W_{i,k-1} \leq \sigma_j + (PV - \sigma_j)(1 - x_{i,j}^{(k)}) \quad \forall i \in I, j \in J, k \in K \tag{20}$$

Though the delivery of some product from batch *i* to an output terminal *j* is feasible, it may happen that batch *i* has been destined for other depots. In such a case, the variable $x_{i,j}^{(k)}$ is driven to zero to fulfill constraint (18).

Let us assume that a new batch *i'* is injected in the line while executing the parallel-run block *k*, and the coordinates of batch *i* (<*i'*) at the completion of blocks ($k-1$) and *k*, satisfy the following condition: $F_{i,k-1} - W_{i,k-1} < \sigma_j \leq F_{i,k}$. Then, an upper bound on the volume of product that can be delivered from batch *i* to the output terminal *j* during block *k* is given by $[\sigma_j - (F_{i,k-1} - W_{i,k-1})]$. However, a different situation may arise if the execution of block *k* enlarges the size of a batch *i* already in the line. It may occur that a pumping run (*k*, *s*) injects an additional amount of material to batch *i* ($Q_{i,s}^{(k)} > 0$) and simultaneously some portion of batch *i* is diverted to accessible downstream depots. As a result, the maximum volume that can be delivered from batch *i* to accessible terminals up to depot *j* is given by,

$$\sum_{j'=1}^{j} D_{i,j'}^{(k)} \leq \sigma_j - (F_{i,k-1} - W_{i,k-1}) + \sum_{\substack{s\in S \\ \tau_s < \sigma_j}} Q_{i,s}^{(k)} + (PV - \sigma_j)(1 - x_{i,j}^{(k)})$$

$$\forall i \in I, j \in J, k \in K \tag{21}$$

4.5. Non-interacting pumping run constraint

When parallel pumping runs are permitted, an additional important condition should be incorporated in the problem formulation. It aims to prevent from interferences among pipeline flows caused by simultaneous injections. If there is a product movement in a particular pipeline during the execution of block *k*, it should be caused by either running a batch injection (*k*, *s*) from source *s* at the pipe inlet extreme, or receiving material from an adjacent, upstream pipeline. If run (*k*, *s*) is performed, no product flow must enter the pipeline starting at source *s* because of upstream pumping runs. Reciprocally, no batch can be inserted in the pipeline from an intermediate source *s* during block *k* if that line is receiving a finite product flow caused by upstream pumping operations. Combined pushing effects are forbidden. The effect of parallel pumping runs should be confined to a similar number of isolated pipeline sections with no transfer of products between them during their execution.

Fig. 5 illustrates a simple example involving a pair of interacting parallel runs. The pipeline input planning includes the following simultaneous injections: a portion of batch *B5* containing product *P2* from source *N3*, and a volume of *P1* enlarging the size of batch *B4* pumped at input terminal *N5*. During the execution of such a parallel-run block, the pipeline output planning specifies a pair of

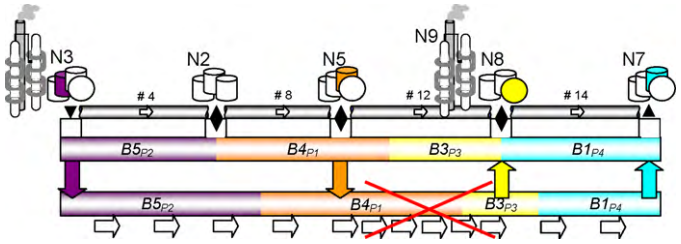

Fig. 5. Incompatible parallel batch injections.

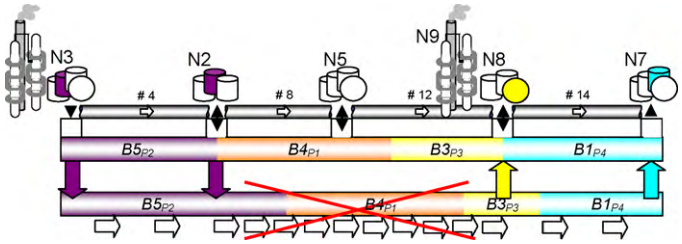

Fig. 6. Interactive parallel injections to the same lot.

product deliveries: *P3* from batch *B3* to terminal *N8*, and *P4* from lot *B1* to destination *N7*. The incompatibility of those batch injections arises because source *N5* is inserting material in the existing batch *B4*, and simultaneously *B4* is being pushed by the succeeding batch *B5* coming from an adjacent, upstream terminal. A combined pushing effect is observed in the pipeline starting at *N5*.

To avoid the generation of a pipeline planning involving incompatible parallel runs, the constraint (22) has been included in the problem model.

$$\sum_{i\in I}\sum_{\substack{j\in J\\ \sigma_j\le\tau_s}} D_{i,j}^{(k)} \ge \sum_{i\in I}\sum_{\substack{s'\in S\\ \tau_{s'}<\tau_s}} Q_{i,s'}^{(k)} - Q_{\max}\left(1-\sum_{i\in I} v_{i,s}^{(k)}\right) \quad \forall s\in S,\ k\in K \tag{22}$$

Constraint (22) states that pumping run (k, s) can be executed $(\sum_{i\in I} v_{i,s}^{(k)} = 1)$ only if no product flow is coming from the upstream pipeline section while performing block k. In other words, the total volume diverted from in-transit batches to output terminals featuring $\sigma_j \le \tau_s$ should be exactly equal to the total volume injected from input sources s' located at upstream pipelines. This condition is forced by constraint (22) only if $\sum_{i\in I} v_{i,s}^{(k)} = 1$.

In many cases, an alternative condition to prohibit interacting pumping runs can be written. Since no product flow is coming from the upstream pipeline section, the lower coordinate of an empty/non-empty batch i receiving material from an intermediate input terminal s (i.e. $v_{i,s}^{(k)} = 1$) remains unchanged during the execution of run block k. Since $F_{i+1,k}$ represents both the upper coordinate of batch $i+1$ and simultaneously the lower coordinate of batch i at the end of block k, the compatible pumping-run condition is given by,

$$F_{i+1,k} \le F_{i+1,k-1} + PV(1 - v_{i,s}^{(k)}) \quad \forall i\in I,\ s\in S(s>1),\ k\in K \tag{23}$$

This condition is not satisfied by the parallel runs shown in Fig. 5 where the lower coordinate of the receiving batch *B4* moves forward while executing the planned batch injections. However, Eq. (23) is less restricted than constraint (22). There are some cases where interacting parallel runs are erroneously considered feasible by constraint (23). In other words, such a constraint may allow blocks of incompatible parallel runs within the problem feasible space.

Let us consider a simple example involving a pair of simultaneous injections of product *P2* to the same receiving batch *B5* from both input nodes *N3* and *N2* (see Fig. 6). The lower coordinate of the receiving batch *B5* remains unchanged while accomplishing both pumping runs, i.e. it is always located at the pipeline system origin. However, there is a positive flow from the pipeline segment *N3–N2* coming to the adjacent pipeline also receiving *P2* from source *N2*. In contrast, constraint (22) is not satisfied because the volume of *P2* injected from *N3* moves forward beyond node *N2*.

To make condition (23) equivalent to Eq. (22), it is necessary to consider a complementary constraint that confines the effect of simultaneous pumping runs to isolated pipeline sections even when both runs inject material to the same receiving batch.

$$\sum_{\substack{j\in J\\ \sigma_j\le\tau_s}} D_{i,j}^{(k)} \ge \sum_{\substack{s'\in S\\ \tau_{s'}<\tau_s}} Q_{i,s'}^{(k)} - Q_{\max}(1 - v_{i,s}^{(k)}) \quad \forall i\in I,\ s\in S,\ k\in K \tag{24}$$

Though similar to constraint (22), Eq. (24) just applies to upstream product injections to (and deliveries from) the same batch. Using a combination of constraints (23) and (24) instead of Eq. (22), the proposed formulation shows a much better computational performance.

4.6. Product supply and demand constraints

4.6.1. Assigning products to batches

Every batch can at most contain a single product. Let $y_{i,p}$ be a binary variable denoting that batch i contains product p whenever $y_{i,p} = 1$. Then

$$\sum_{p\in P} y_{i,p} \le 1 \quad \forall i\in I \tag{25}$$

If a pre-defined batch i does not contain any product ($y_{i,p} = 0$, $\forall p\in P$), it has never been injected into the pipeline. In other words, $\sum_{p\in P} y_{i,p} = 0$ implies that there is no pumping run inserting batch i in the line. Therefore, batch i is a fictitious lot and: $\sum_{s\in S}\sum_{k\in K} v_{i,s}^{(k)} = 0$. If instead product p has been allocated to batch i, then there is at least one pumping run inserting batch i carrying some volume of product p in the line. This condition can be mathematically written as follows,

$$\sum_{p\in P} y_{i,p} \le \sum_{s\in S}\sum_{k\in K} v_{i,s}^{(k)} \le |S||K|\sum_{p\in P} y_{i,p} \quad \forall i\in I^{new} \tag{26}$$

To reduce the size of the problem feasible region without cutting off the optimal solution, fictitious batches featuring $y_{i,p} = 0$ for all $p\in P$ are confined to the end of the batch sequence through the following constraint,

$$\sum_{p\in P} y_{i,p} \le \sum_{p\in P} y_{i-1,p} \quad \forall i\in I^{new}(i>1) \tag{27}$$

Because of product contamination, some product sequences are forbidden. Let us assume that (p, p') stands for a forbidden sequence of products. Then,

$$y_{i-1,p'} + y_{i,p} \le 1 \quad \forall i\in I^{new}(i>1) \tag{28}$$

4.6.2. Amount of product p injected in the line through parallel-run block k

If $y_{i,p} = 0$, the amount of product p in batch i inserted in the line from some input terminal will be equal to zero. Otherwise, $y_{i,p} = 1$ and the volume of product p contained in batch i pumped in the pipeline during the run (k, s) will be equal to $Q_{i,s}^{(k)}$. Both conditions

are stated through Eqs. (29) and (30).

$$QP_{i,s,p}^{(k)} \le Q_{\max,p} y_{i,p} \quad \forall i \in I, s \in S, k \in K, p \in P \tag{29}$$

$$\sum_{p \in P} QP_{i,s,p}^{(k)} = Q_{i,s}^{(k)} \quad \forall i \in I, s \in S, k \in K \tag{30}$$

$Q_{\max,p}$ stands for the maximum batch injection size for product p.

4.6.3. Volume of product p stripped from in-transit batches and delivered to depots

No product p can be delivered from batch i to output terminal j through any run in block k if lot i does not contain p. Otherwise, $y_{i,p} = 1$ and the amount of product p supplied by batch i to depot j during block k is given by $D_{i,j}^{(k)}$. Both conditions are imposed by Eqs. (31) and (32), respectively.

$$DP_{i,j,p}^{(k)} \le D_{\max} y_{i,p} \quad \forall i \in I, j \in J, k \in K, p \in P \tag{31}$$

$$\sum_{p \in P} DP_{i,j,p}^{(k)} = D_{i,j}^{(k)} \quad \forall i \in I, j \in J, k \in K \tag{32}$$

4.6.4. Feasible range for the amount of product p shipped from source s

Let us assume that $SU_{p,s}$ stands for the total amount of product p initially available in source s. Besides, $SL_{p,s}$ stands for a lower bound on the amount of product p that should be pumped into the line from source s during the time horizon. Hence,

$$SL_{p,s} \le \sum_{k \in K} \sum_{i \in I} QP_{i,s,p}^{(k)} \le SU_{p,s} \quad \forall p \in P, s \in S \tag{33}$$

$SL_{p,s}$ is usually large enough to fulfill, in combination with other sources, the specified demands of product p at distribution depots not covered by the initial linefill. Moreover, it may also include the "sweeping" lots pushing the overall pipeline content to the assigned depots. Such filler lots are selected to either getting a suitable final linefill to meet future product demands or providing free capacity to receive new production runs from source s.

4.6.5. Fulfilling product demands at every output terminal

Let $DL_{p,j}$ be the demand of product p at the output terminal j to be satisfied before the horizon end. Since the storage capacity at any output terminal is finite, the total amount of product p diverted from the pipeline to depot j during the planning horizon should be bounded. Let us define the model parameter $DU_{p,j}$ as the maximum amount of product p that can be delivered and stored in depot j. Then, the total amount of product p delivered to terminal j from any batch containing p is constrained as follows,

$$DL_{p,j} - B_{p,j} \le \sum_{k \in K} \sum_{i \in I} DP_{i,j,p}^{(k)} \le DU_{p,j} \quad \forall p \in P, j \in J \tag{34}$$

When the demand of product p at depot j cannot be satisfied before the end of the planning period, a non-zero backorder $B_{p,j} > 0$ will arise. By including the continuous variable $B_{p,j}$ in Eq. (34), the pipeline scheduling problem will remain feasible even though some demands are unsatisfied at the horizon end.

4.7. Initial linefill

Let W_i^o be the volume of the old batch $i \in I^{old}$ already in the pipeline at the start of the scheduling horizon. Then, the upper coordinate of batch $i \in I^{old}$ can be obtained by summing the volume of every old batch $i' \in I^{old}$ succeeding batch i, plus the initial volume of batch i.

$$F_{i,k-1} = \sum_{\substack{i' \ge i \\ i' \in I^{old}}} W_{i'}^o \quad \forall i \in I^{old}, k = 1 \tag{35}$$

Moreover, it is also known the product P_i contained in every old batch $i \in I^{old}$.

$$y_{i,p} = 1 \quad \text{for } p = P_i, \forall i \in I^{old} \tag{36}$$

4.8. Objective function

Planning the execution of parallel operations aims to better utilizing the pipeline transportation capacity and making product deliveries at receiving locations on schedule. As a result, the required pipeline operations can be completed in a shorter time. This is why two alternative problem goals have been chosen. They are:

(1) *The minimum makespan*, assuming a non-fixed horizon length and non-specified delivery due dates. Pipeline operations should be completed as soon as possible.

$$Min \quad z = H, \quad subject\ to\ H \ge C_k \quad \forall k \in K \tag{37}$$

(2) *The minimum total cost*, including transition, pumping and backorder costs, and assuming that the horizon length and delivery due dates at output terminals are given.

$$Min \quad z = \sum_{i \in I} TC_i + \sum_{k \in K} PC_k + BC \tag{38}$$

where,

$$TC_i \ge cif_{p,p'}(y_{i,p} + y_{i+1,p'} - 1) \quad \forall i \in I; p, p' \in P \tag{39}$$

$$PC_k = \sum_{i \in I} \sum_{s \in S} \sum_{p \in P} cin_{p,s} QP_{i,s,p}^{(k)} \quad \forall k \in K \tag{40}$$

$$BC = \sum_{p \in P} \sum_{j \in J} cb_{p,j} B_{p,j} \tag{41}$$

The parameter $cif_{p,p'}$ is the cost for reprocessing the interface volume $p - p'$, the coefficient $cin_{p,s}$ stands for the cost of pumping a unit volume of product p from the input terminal s and $cb_{p,j}$ stands for the cost of failing to provide a single unit of product p at depot j on schedule.

5. Results and discussion

5.1. Example 1

Example 1 first introduced by Jittamai (2004) and recently reformulated by Cafaro and Cerdá (2009) is concerned with the short-term operational planning of a pipeline network involving four unidirectional pipelines transporting three refined products *A*–*B*–*C* from a pair of oil refineries *S1*–*S2* to three distribution depots *D1*–*D2*–*D3*. Pipeline # *1* goes from source *S1* to depot *D1*, pipeline # *2* connects depot *D1* to source *S2* and so on (see Fig. 7). Any refined product can be injected from both the source *S1* at the origin of pipeline #1 and the downstream source *S2* at the inlet of pipeline # *3*. Available supplies of refined products in refinery storage tanks and product demands to be satisfied at receiving terminals are shown in Table 1. Moreover, Table 2 provides the transition costs between lots of different refined products. For instance, the interface volume between batches of products *A* and *B* has a reprocessing

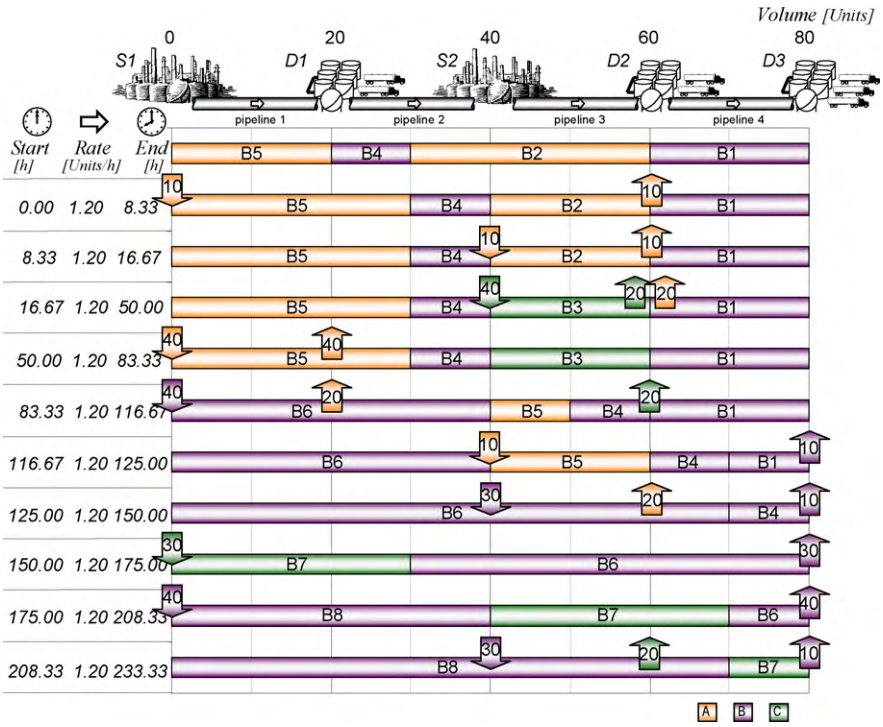

Fig. 7. Best sequential pipeline schedule for Example 1.

Table 1
Product supplies and demands for Example 1.

	Supplies (in units)		Demands (in units)		
	Input nodes		Destinations		
	Source 1	Source 2	Depot 1	Depot 2	Depot 3
A	50	20	60	60	–
B	80	60	–	–	100
C	30	40	–	60	–

Table 2
Interface costs (in 10^2 $).

Predecessor	Successors		
	A	B	C
A	–	22.0	35.0
B	24.0	–	21.0
C	30.0	32.0	–

cost of 22.0 (10^2 $) if the lot of B is preceded by the lot of *A*. Product-dependent pumping costs are given in Table 3. A maximum pump rate of 1.20 (units/h) for any product is adopted.

The total pipeline content amounts to 80 volume units, while the capacity of every individual pipeline is equal to 20 units. The initial linefill includes a sequence of four finite lots of refined products *B5–B4–B2–B1* and is shown at the top of Fig. 7. To account for the possibility of inserting a new lot between existing batches *B4* and *B2*, a fifth empty lot *B3* has been considered. The injection of lot *B3* and the refined product assigned to it are both model decisions. The pipeline network is operated on fungible mode, and the product injection at any input terminal can have a maximum size of 40 units and a minimum size of 10 units. Nonetheless, a batch can increase its volume while traveling through the network by receiving additional amount of product from downstream sources.

Table 3
Product-dependent pumping costs at every refinery.

Pumping costs (10^2 $ per unit volume)		
Product (p)	Refinery	
	Source 1	Source 2
A	29.0	14.5
B	34.0	17.0
C	49.0	24.5

As remarked before, one of the major advantages of executing a block of non-interacting parallel runs is the better utilization of the pipeline capacity and, consequently, a significant reduction in the amount of time required for completing all the specified product deliveries, i.e. the makespan. For this reason, the minimum total cost has been selected as the primary objective and the minimum makespan as the secondary target. To show the improvements that are obtained by moving from sequential to parallel pipeline schedules, the best solutions to Example 1 with and without simultaneous batch injections have been found. When pumping runs are to be accomplished one after another, the continuous approach of Cafaro and Cerdá (2009) provides the sequential pipeline schedule shown in Fig. 7. Model size and computational requirements for the sequential instance of Example 1 are reported in Table 4. The best sequential pipeline schedule has been found in 51.7 s on a 64 bits 4-processors (3.0 GHz) Pentium IV PC with GAMS/GUROBI 1.0 MILP solver (Brooke, Kendrick, Meeraus, & Raman, 2006) and a relative tolerance gap of 1e−03.

Ten pumping runs *k1–k10*, five at node *S1* and the remaining five at the intermediate source *S2*, are performed to satisfy product needs at distribution depots (see Fig. 7). The sequential schedule has a makespan of 233.33 h. However, runs *k3* at *S2* and *k4* at *S1* could both be accomplished simultaneously. During run *k3*, batch *B3* containing 40 units of product *C* is pumped in the line at source *S3* from time 16.67 h to 50.00 h, and 20 units of *A* from batch *B2* and 20 units of *C* from *B3* are extracted to depot *D2*. Therefore, only pipeline # *3* remains active. Afterwards, 40 units of product *A* are added to batch *B5* by performing run *k4* at the origin of pipeline # *1*, and a similar amount of *A* from *B5* is delivered to *D1*. During run *k4* taking place from time 50.00 h to 83.33 h, only pipeline # *1* is active.

Table 4
Model statistics and computational results for Example 1.

Case	Eqs.	Cont. variables	Binary variables	CPU time (s)	Number of iterations	Optimal cost (10^2 $)	Makespan	Interface cost (10^2 $)
Sequential schedule	4091	2038	477	51.73	372,538	8120.0	233.33	190.0
Parallel schedule	3812	1856	432	275.32	5,292,702	8120.0	183.33	190.0

Runs *k3* and *k4* have a similar length of 33.33 h. Since pipeline # *2* between pipes # *1* and *3* remains idle while performing *k3* and *k4*, then both runs are non-interactive and can be accomplished in parallel. As a result, the makespan could be diminished by at least 33.33 h, i.e. from 233.33 h to 200.00 h. But the new MILP mathematical formulation allowing parallel pumping runs provides a pipeline schedule even better featuring a makespan lower than 200.00 h.

The best parallel pipeline schedule that was obtained by solving the new MILP mathematical formulation is shown in Fig. 8. It includes 7 single batch injections (*k1–k2*, *k5–k9*) and a pair of blocks with two parallel pumping runs (*k3–k4*).

As expected, runs *k3* and *k4* in the sequential schedule are now executed simultaneously in a parallel block, now called *k3*, from time 25.00 h to 58.33 h. Furthermore, the optimal parallel schedule comprises an additional block of simultaneous batch injections (*k4*). At the end of block *k3*, lot *B4* is properly located to receive 40 additional units of product *B* from the intermediate source *S2*. By doing so, 20 units of product *C* from *B3* and 20 units of product *B* from lot *B1* are extracted to depots *D2* and *D3*, respectively. During that pump operation, pipelines # *3* and *4* will be active. At the same time, it can be inserted a new lot *B6* containing 20 units of product *B* at node *S1* to extract 20 units of *A* from lot *B5* to depot *D1*. In this way, the product movement will be confined to pipeline # 1. Therefore, both batch injections from sources *S1* and *S2* do not interact between themselves because the corresponding active pipeline sections # *1* and # (*3* and *4*) remain separated by the idle pipeline # *2*. The block *k4* is executed from time 58.33 h to 91.67 h, though the shorter pumping run at *S1* may be completed earlier. Then, there are further time savings of 16.67 h. In this manner, the best parallel pipeline schedule features a makespan of 183.33, well below the 233.33-mark of the sequential schedule (see Table 4). As a result, the horizon length has been cut by more than two days.

Besides, the number of batch injections has increased to 11 though the active blocks of pumping runs drops to 9. When blocks of parallel runs are permitted at large-size pipeline scheduling problems, the required cardinality of set *K*, the model size and the CPU time all usually decrease. From Table 4, it is concluded that such a trend holds at Example 1 except for the CPU time that grows from 51.7 s to 275.3 s. This may happen mainly because Example 1 is a rather small problem. Pumping and transition costs take similar values in both cases.

Fig. 9 describes the input events taking place in every pipeline while executing the sequential schedule, i.e. the pipeline activities. An input event occurs whenever a pipeline receives a new lot coming from the adjacent pipeline, or from the source located at its origin (indicated with a frame in Fig. 9). When run *k1* is executed, several input events take place. An additional portion of lot *B5* is inputted to pipeline # *1*, a part of lot *B5* is received by pipeline # *2*, and a fraction of lot *B2* is tightlined to pipeline # *3*. Therefore, three pipelines are active. During run *k2*, only pipeline # *3* receives an additional part of lot *B2* from the input terminal *S2*. The pipeline system activity is just confined to pipeline # *3* because a similar amount of product is delivered from *B2* to depot *D2* at the outlet of pipe # *3*. The number of events significantly increases in later runs like *k8* during which a new lot *B7* is inserted at the inlet of pipeline # *1*. At the same time, lot *B6* moves forward through pipelines # *2*, *3* and *4*, and the front portion of *B7* also enters pipeline # *2*. Therefore, the activity is spread out over the whole pipeline system.

On the other hand, Fig. 10 shows the input events for the parallel schedule. It is quite clear that the pipeline activity has substantially increased and the available transport capacity is better utilized. The input events for the first two runs are similar to those taking place in the sequential case. Major differences arise when the parallel block *k3* is executed. This time an additional part of lot *B5* enters

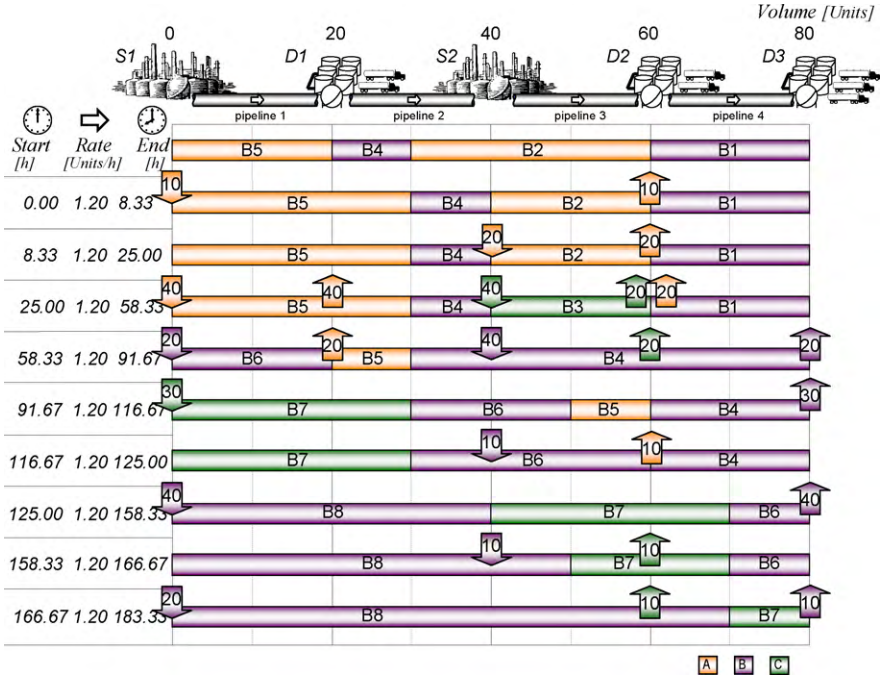

Fig. 8. Best parallel pipeline schedule for Example 1.

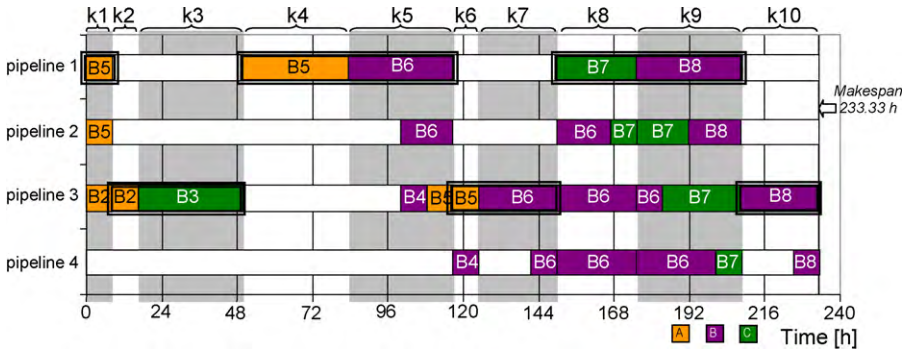

Fig. 9. Input events at the optimal sequential schedule for Example 1.

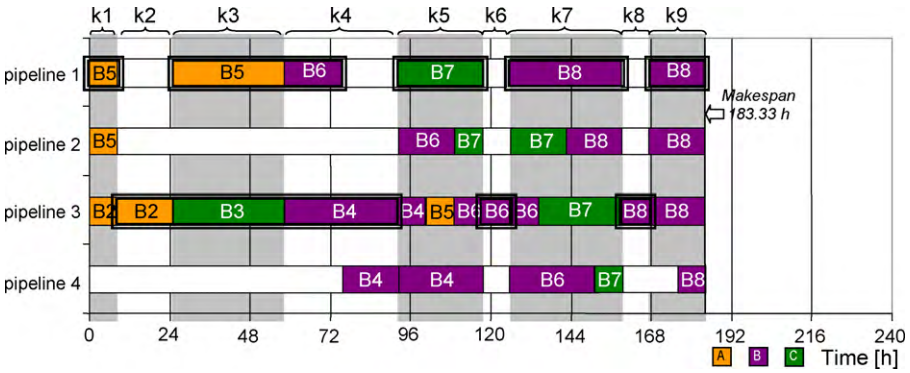

Fig. 10. Input events at the optimal parallel schedule for Example 1.

pipeline # *1* and a new batch *B3* is inserted in pipeline # *3*. However, activities are just confined to those pipelines because delivery operations simultaneously occur in depots *D1* and *D2* at their end extreme. During runs *k5*, *k7* and *k9*, the whole pipeline system is active, despite a single batch injection is carried out in all of them at the origin of pipeline # *1*.

5.2. Example 2

Example 2 deals with a real-world distribution network composed by three unidirectional trunk pipelines transporting five oil derivatives (*P1*–*P2*–*P3*–*P4*–*P5*) from two sources (*N1*, *N3*) to three receiving depots (*N2*, *N3*, *N4*). Moreover, the pipeline system shown in Fig. 11 includes a dual-purpose node *N3* where pumping and delivery operations can take place. In contrast, *N2* stands for an output terminal supplying products *P1*, *P2* and *P4* to neighboring markets through a lateral pipeline, and the distribution depot *N4* provides all products (*P1*–*P5*) to another important consumer zone. Products demands for the next ten days at depots *N2*, *N3* and *N4*, and product inventories at source nodes (*N1*, *N3*) are listed in Tables 5 and 6, respectively.

The overall length of the pipeline system from *N1* to *N4* is over 1000 km and the volumes of the three pipelines are 33,600, 23,300 and 27,700 m^3, respectively. Moreover, the pump rate at the two input terminals should be set within the range 310–580 m^3/h, while unit pumping costs and interface reprocessing costs are given in Tables 7 and 8. Interface costs are assumed to be independent of the pumping rate, stoppage times and traveled distances. Some product transitions indicated with an X in Table 8 are forbidden. For instance, fuel *P3* cannot travel immediately before or after neither *P2* nor *P4*.

Table 5
Product demands at receiving terminals (in m^3).

	N2	*N3*	*N4*	Total
P1	5110	–	12,180	17,290
P2	41,340	39,210	27,700	108,250
P3	0	–	17,820	17,820
P4	3090	2120	1100	6310
P5	0	–	37,170	37,170

Table 6
Product inventories at source nodes (in m^3).

	N1	*N3*	Total
P1	25,884	4548	30,432
P2	167,772	–	167,772
P3	5316	21,384	26,700
P4	9468	–	9468
P5	20,532	30,876	51,408

Table 7
Product-dependent unit pumping costs (in $/$m^3$).

	N1	*N3*
P1	8.136	3.042
P2	9.492	–
P3	10.170	4.056
P4	8.814	–
P5	8.814	3.380

Table 8
Interface costs (in $).

	Successors				
	P1	*P2*	*P3*	*P4*	*P5*
P1	0	3600	3600	4500	2700
P2	3600	0	X	2700	5000
P3	3600	X	0	X	3500
P4	4500	2100	X	0	4500
P5	2400	5100	3000	5400	0

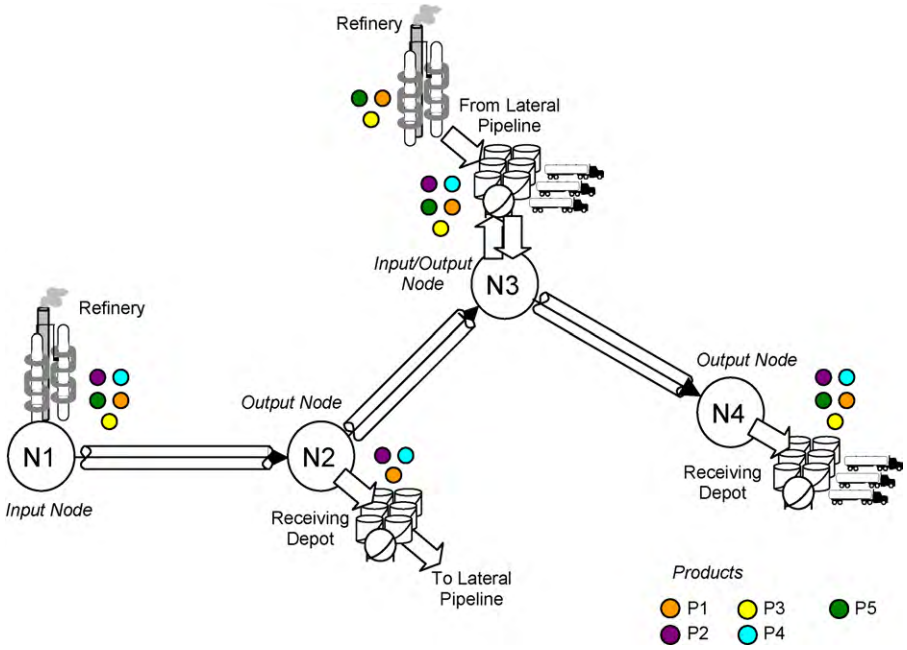

Fig. 11. The pipeline distribution network considered in Example 2.

Besides, there is an upper bound on the size of batch injections equal to 30,000 m^3, and the volume of product deliveries to receiving terminals can never be lower than 1000 m^3 to reduce the number of interfaces.

The initial linefill is shown in Fig. 12. It consists of a sequence of five batches distributed as follows: (i) pipeline # *1* contains batches *B6* with product *P4*, *B5* transporting product *P2*, and the back portion of *B4* involving product *P1*; (ii) the front portion of *B4* and the lot *B2* with product *P5* both reside in pipeline # *2*, and (iii) the volume of pipeline # *3* is fully occupied by batch *B1* with product *P2*. The model will assume the existence of an empty batch *B3* to consider the possibility of injecting a new lot just in the interface of the initial lots *B4* and *B2*. The pumping time and the size of the new batch *B3* are both model decisions. The problem goal is to develop the pipeline operational planning for the next ten days (240 h) in order to satisfy the specified depot needs at minimum total (pumping, transition and backorder) cost. Backorder costs may arise because of unsatisfied demands at the end of the ten-day period, and they are usually highly penalized. It has been assumed a unit backorder cost of 100 ($/$m^3$) for any product.

The best parallel pipeline schedule for Example 2 is shown in Fig. 13. It includes a total of 12 batch injections (6 from each source) that are grouped into 8 blocks of parallel pumping runs. Since some runs just add further amounts of products to existing batches, 10 lots are transported in the pipe over the ten-day horizon. At the top of Fig. 13, the pipeline system is described through a simplified representation where pipelines have been lined up along the volume axis. One of the major advantages of continuous approaches

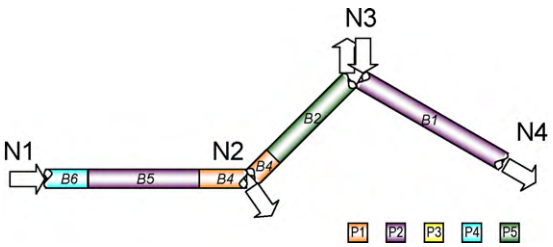

Fig. 12. Initial pipeline linefill for Example 2.

with regards to discrete formulations is the easy treatment of batch routes involving two or more pipelines between their input and receiving terminals. In such cases, batches flow directly from one pipeline to another through what are called tightlining operations. Tightlining is handled by continuous approaches in a very simple manner. The next line of Fig. 13 shows the initial state of the pipeline network, while the following ones depict the batch movements along the line and the delivery operations as new blocks of batch injections take place at source nodes *N1* and *N3*.

The first block *k1* running from time $t = 0.00$ h to 51.41 h involves the insertion of a large lot *B7* containing product *P2* at the origin *N1*. While doing so, the interface *B2–B4* moves forward just to reach the location of the intermediate source *N3*, and simultaneously three delivery operations are carried out. Some amounts of products *P1* and *P2* are stripped from lots *B4* and *B5*, and delivered to depot *N2*, while a certain quantity of *P2* is sent from *B1* to depot *N4*. The next parallel blocks *k2* and *k3* both involve simultaneous pumping runs at the two input terminals *N1* and *N3*. During block *k2* running from $t = 51.41$ h to 78.47 h, the refined product *P5* is injected in the pipeline from both sources: a new lot *B8* having a volume of 15,694 m^3 is inputted at node *N1*, and a further volume of 4626 m^3 of product *P5* is added to the existing lot *B2* at *N3*. A maximum pump rate of 580 m^3/h was adopted at *N1*, while the input rate at *N3* is significantly lower. Batch movements are just confined to pipelines # *1* and *3*, and the interface *B2–B4* remains steady at the location of *N3*. The parallel block *k3* comprises a pair of batch injections at the inlet of pipelines # *1* and *3*. A new lot *B9* containing 17,906 m^3 of product *P2* is inputted at *N1*, and the empty lot *B3* traveling between *B2* and *B4* in the initial linefill is injected at *N3* with 17,820 m^3 of product *P3*. Both operations are carried out almost at maximum pump rate (580 and 577.21 m^3/h, respectively). In each of the next four blocks *k4–k7*, it is performed a single batch injection. Two blocks (*k4*, *k6*) are executed at source *N3* to insert additional volumes of product *P1* to the existing batch *B4*. Besides, pumping runs (*k5*, *k7*) take place at source *N1* to add further amounts of *P2* to the recently inserted batch *B9*. Finally, the last block *k8* running from $t = 194.74$ h to 240.00 h comprises a pair of simultaneous batch injections from sources *N1* and *N3*, respectively. On one hand, the new batch *B10* containing 25,884 m^3 of product *P1* is inputted at terminal *N1* to deliver a similar amount of *P2* from batch *B9* to terminal *N3*. The

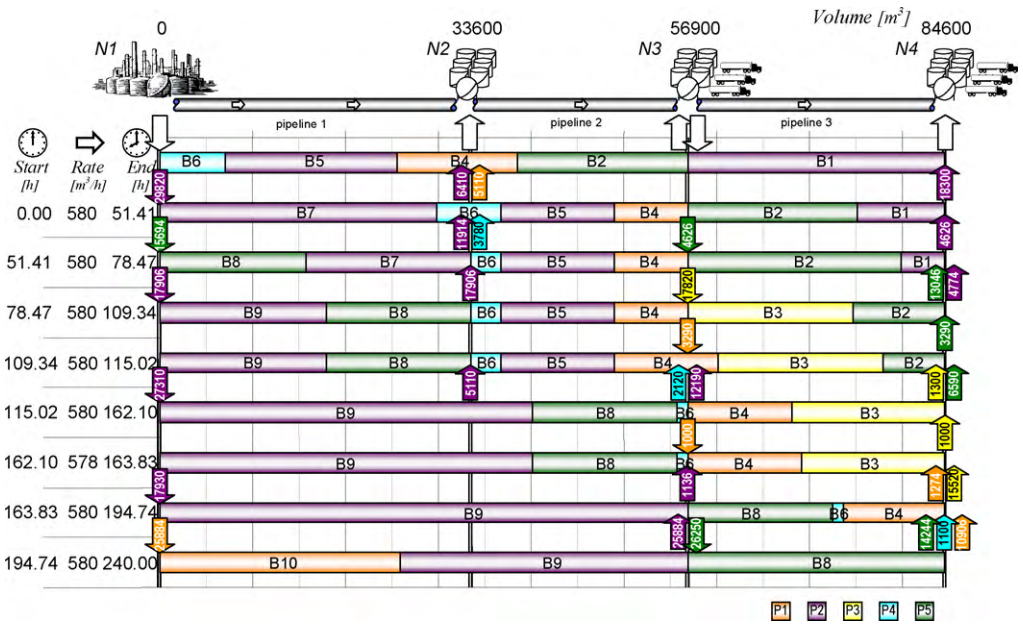

Fig. 13. Optimal parallel pipeline schedule for Example 2.

related batch movements are then confined to pipelines # *1* and *2* connecting *N1* to *N3*. As a result, the pumping run accomplished at source *N1* does not affect the location of batch *B8* in pipeline # *3*. On the other hand, a further amount of product *P5* (26,250 m^3) is simultaneously injected to the existing batch *B8* from source *N3* to strip a similar volume of products *P1*–*P4*–*P5* from batches *B4*–*B6*–*B8* to the last depot *N4*. In the latter case, the pipeline activity just occurs in pipeline # *3*. Therefore, block *k8* executes a pair of non-interacting pumping runs.

Through this careful coordination among pumping activities at input terminals provided by the proposed MILP model, product demands at distribution depots are fully satisfied within the ten-day horizon. Since the amount of *P4* contained in the initial lot *B6* exceeds by 690 units the overall demand of *P4* requested at depots *N2*–*N3*–*N4*, such a volume is rapidly withdrawn from *B6* at the closest depot *N2* demanding *P4*. It is stripped from lot *B6* to depot *N2* a volume of 3780 m^3, despite *N2* only requests 3090 m^3 of *P4*. In all other depots, batch deliveries exactly meet product requirements. As expected, available inventories of products *P1*, *P3* and *P5* at source *N3* are mostly running out to satisfy demands at the near depot *N4*. Initial inventories of such products in *N3* are depleted by 94.33%, 83.33% and 100%, respectively, at the end of the planning horizon. In this way, pumping costs are significantly reduced.

In order to compare the proposed formulation allowing simultaneous batch injections at multiple input terminals with the sequential approach, based on both operational performance and computational cost, Example 2 has also been solved using the MILP model recently introduced by Cafaro and Cerdá (2009). To make a fair comparison, it has been adopted $|K| = 8$ and $|I| = 10$ in both cases. The best sequential pipeline schedule depicted in Fig. 14 involves 8 pumping runs against 12 performed at the parallel schedule. Though the model sizes are quite similar, the required CPU time to

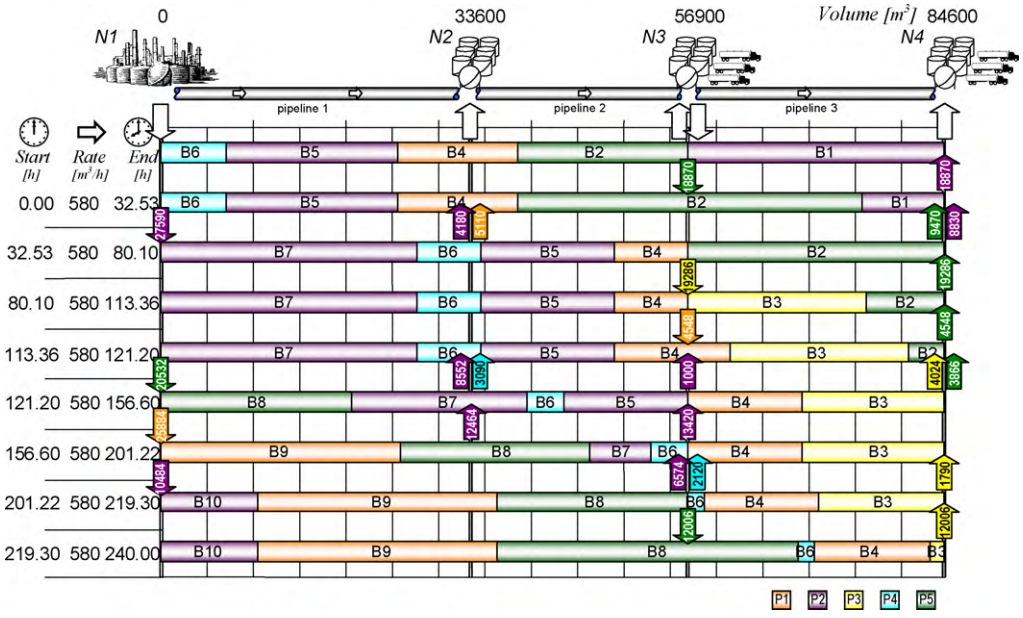

Fig. 14. The best sequential schedule for Example 2 (8 pumping runs).

Table 9
Model sizes and computational results for Example 2.

	Runs $\|K\|$	Lots $\|I\|$	Eqs.	Cont. var.	Bin. var.	CPU time (s)	Iter. (10^6)	Opt. gap (%)	Pump cost (10^3\$)	Interf. cost (10^3\$)	Back orders (%)
SIN_8	8	10	4605	2504	450	865.8	17.6	0.0	1421.0	33.7	0.0
NSI_8	8	10	4461	2504	450	4330.5	88.4	<0.1	949.4	31.0	25.5
NSI_9	9	10	4977	2808	500	20,000	361.5	0.2	946.8	32.6	25.5

SIN_8: Simultaneous Injections (8 parallel blocks). NSI_8: Non-Simultaneous Injections (8 runs). NSI_9: Non-Simultaneous Injections (9 runs).

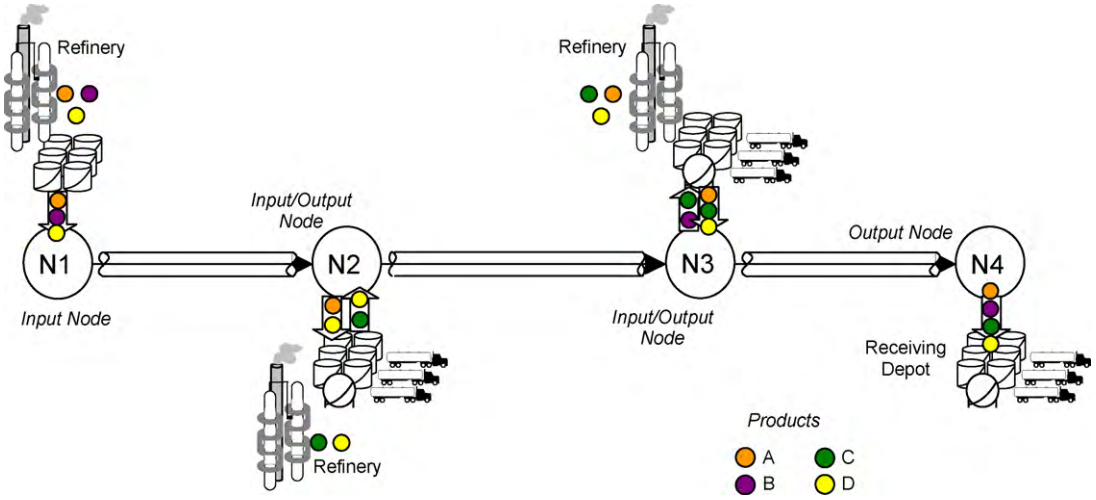

Fig. 15. Refined products pipeline network for Example 3.

find the best sequential schedule increases almost five times with respect to the simultaneous approach, i.e. from 865.8 s to 4330.5 s of CPU time (see Table 9). Therefore, the computational cost sharply grows despite the decrease in the number of batch injections performed at the optimal sequential schedule. As a result, it is observed a lower usage of the pipeline transport capacity that prevents from satisfying 25.5% of the requested demands within the ten-day horizon. Such a decrease in pipeline activity, especially at node *N1*, explains the lower pumping cost exhibited by the sequential schedule (see Table 9). By increasing the cardinality of the set *K* from 8 to 9, no improvement in the sequential schedule is achieved. Backorders still remain at 25.5% and the optimality is not guaranteed after 20,000 s of CPU time (see Table 9).

5.3. Example 3

Example 3 considers a refined products supply network composed by three unidirectional pipelines (*PL1*, *PL2*, *PL3*) that transport four liquid fuels (*A*, *B*, *C*, *D*) from three input terminals (*N1*, *N2*, *N3*) to three receiving depots (*N2*, *N3*, *N4*). Nodes *N2* and *N3* are dual-purpose terminals that can send/receive product batches to/from the pipelines. The pipeline network structure for Example 3 is depicted in Fig. 15. Pipelines *PL1*–*PL2*–*PL3* have a size of 30, 40 and 30 volumetric units (v.u.) respectively, and the product injection rate at source nodes is restrained to the range 0.80–1.20 v.u./h. The primary problem goal is to satisfy all terminal demands at minimum makespan, while the secondary target aims to minimizing pipeline operating costs including pumping and interface costs.

Table 10 provides the initial product availabilities at the source nodes and the product demands for the next ten days at the receiving depots, in volumetric units (v.u.). Each source node can ship batches of products through the pipeline system to downstream terminals and/or dispatch them by truck to neighbouring markets. The latter product demands arise as local requirements at the supplying sources in Table 10. For instance, the origin-node *N1* sends products *A*–*B* by pipeline to terminals *N2*–*N3*–*N4*, and delivers by road products *A* and *D* to near markets. Most products required at downstream terminals are available at two different upstream sources. Because of the pumping cost, closer sources should be preferred. Optimal product flows from sending to receiving terminals are determined by solving the proposed model. In Table 11, average product-dependent pumping costs from each source are given. Lower pumping costs correspond to input stations closer to the destined terminals.

Table 10
Product supplies and demands (in v.u.) at pipeline terminals.

		N1	*N2*	*N3*	*N4*
Product availability (in v.u.)	A	80	–	40	
	B	80	–	–	
	C	–	60	70	
	D	10	80	60	
Product demand (in v.u.)	A	20	30	–	40
	B	–	–	40	30
	C	–	–	30	40
	D	10	20	–	80

Table 12 presents the interface reprocessing cost for each ordered pair of products. Besides, two additional constraints are considered: (i) a maximum size of 50 v.u. for any shipment pumped from an input terminal, and (ii) a minimum size of 5 v.u. for any product delivery to a receiving terminal during a pumping run.

The initial linefill comprises a sequence of four batches: *B5*[*B*]–*B4*[*D*]–*B2*[*A*]–*B1*[*D*], containing the products arising between brackets (see Fig. 16). *B3* is an empty batch that has been reserved

Table 11
Unit pumping costs (in 10^2 \$/v.u.).

	N1	*N2*	*N3*
A	30.0	–	9.0
B	35.0	–	–
C	–	35.0	15.0
D	45.0	31.5	13.5

Table 12
Interface reprocessing costs (in $).

Predecessor	Successor			
	A	B	C	D
A	–	2200	3500	2600
B	2400	–	2100	2400
C	3000	3200	–	3200
D	2800	2200	3000	–

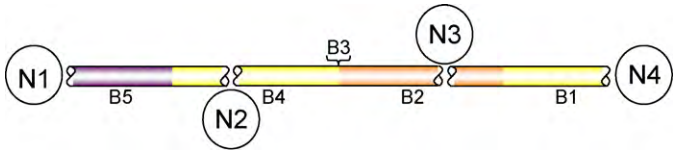

Fig. 16. Initial linefill for Example 3.

for the injection of a new lot at the downstream source *N3*. The lot size and the product allocated to batch *B3* are both model decisions.

To determine the best sequential pipeline schedule for Example 3, the maximum number of pumping runs $|K|$ is first guessed using the approximate equation proposed by Cafaro and Cerdá (2009).

$$|K| \approx \sum_{p \in P} \left(\frac{2}{Q_{\min,p} + Q_{\max,p}} \sum_{j \in J} DL_{p,j} \right) = \frac{2}{5+50} 260 = 9.45 \approx 9$$

By solving the MILP formulation of Cafaro and Cerdá (2009) to optimality, the best solution is found in 617.4 s of CPU time using an Intel Quad Core 2 GHz processor and the MILP solver GAMS/GUROBI 1.0. Since all proposed runs have been accomplished, $|K|$ was increased to 10 in order to confirm the solution optimality. Table 13 indicates that a pipeline schedule with the same makespan and the same total operating cost was determined for $|K| = 10$. However, the required CPU time increases by a factor of 2 because the model size in terms of binary variables and constraints grows 10%.

The effect of $|K|$ on both the solution quality and the CPU time is illustrated in Table 13. By reducing $|K|$ to 8, it is still found an optimal pipeline schedule in 587.1 s. Nonetheless, lower values of $|K|$ lead to higher operating costs ($|K| = 7$) or to infeasibility ($|K| = 6$).

The best sequential pipeline schedule for Example 3 with $|K| = 8$ is shown in Fig. 17. It involves the execution of 8 pumping runs. New batches *B3–B7–B6* containing products *C–D–A* are inserted at nodes *N3–N2–N3* through pumping runs *k2–k6–k8*, respectively. Batch *B8* with product *A* is injected at the origin *N1* through two non-consecutive pumping runs (*k4* and *k7*). Moreover, batch *B5* in the initial linefill receives two new shipments of product *B* from *N1* through runs *k1* and *k3*, and a further amount of product *D* is added to the existing batch *B4* from *N3* (run *k5*). In short, four injections are performed at the origin (*N1*), another one from node *N2*, and the remaining three from *N3*. Product demands at receiving terminals are fully satisfied in 216.67 h, i.e. the minimum makespan. Besides, the overall operating cost amounts to 637,000 (see Table 13).

The other option is to develop a pipeline schedule with simultaneous batch injections. To this end, the number of blocks of parallel runs $|K|$ is first guessed using the equation introduced in Section 4.1.

$$|K| \approx \frac{2}{|S|+1} \sum_{p \in P} \left(\frac{2}{Q_{\min,p} + Q_{\max,p}} \sum_{j \in J} DL_{p,j} \right)$$
$$= \frac{2}{3+1} \frac{2}{5+50} 260 = 4.73 \approx 5$$

However, the proposed problem formulation for $|K| = 5$ becomes infeasible. By increasing $|K|$ to 6 and solving the model again, it was found the parallel pipeline schedule shown in Fig. 18 that features a makespan equal to 166.67 h. Then, the length of time needed to fulfill all terminal demands is shortened by 50 h compared with the best sequential schedule. Moreover, the required CPU time is cut down by a factor of 13, i.e. from 587.1 s to 45.3 s (see Table 14).

To corroborate the solution optimality, the problem model with $|K| = 7$ has again been solved. From Table 14, it follows that the same optimum is found in 125.8 s of CPU time. The reason for the makespan reduction is quite simple. It comes from the execution of two parallel blocks each involving a pair of non-interactive pumping runs. They are: (i) the parallel block *k4* simultaneously injecting batch *B8* from source *N1* and batch *B4* from *N3*; and (ii) the parallel block *k6* inserting another portion of *B8* in the line from *N1*, and the new batch *B6* from node *N3* at the same time. In the sequential pipeline schedule, such injections are performed one after another. During block *k4*, the sending of a lot of product *A* from *N1* to deliver some amount of product *B* to terminal *N3* activates pipelines *PL1* and *PL2*. At the same time, the injection of 30 units of product *D* contained in lot *B4* to supply product *D* to terminal *N4* activates only pipeline *PL3*. Since the sets of pipelines activated by those simultaneous pumping runs show no intersection, then they are non-interactive and the parallel block *k4* is feasible.

Therefore, the proposed simultaneous approach presents significant advantages with respect to previous continuous contributions. Among them, it can be mentioned the following ones: (1) it is still based on a continuous mathematical representation in both time and volume domains; (2) while the model size remains nearly the same compared with the sequential case, the number of input and delivery operations significantly increases; (3) the computational cost drastically drops and good feasible solutions are more rapidly discovered; (4) optimal parallel schedules are less affected by variations on the number of pumping blocks $|K|$; and (5) the utilization of the network transport capacity substantially rises. Such improvements are obtained through a better coordination of input operations at the multiple sources of a pipeline network composed by several pipes arranged in series. Some further work is currently under way to extend the length of the planning horizon and to tackle pipeline systems with bi-directional lines and branching configurations.

Table 13
Results for Example 3 using the sequential model (Cafaro & Cerdá, 2009).

$\|K\|$	Sequential schedules						
	Makespan (h)	Opt. cost (10^2 $)	Eqs.	Cont. variables	Binary variables	Iterations (10^6)	CPU time (s)
6	Infeasible	–	2734	1350	320	1.4	55.0
7	216.67	7039	3159	1568	368	3.2	122.4
8	216.67	6370	3584	1786	416	13.6	587.1
9	216.67	6370	4009	2004	464	15.3	617.4
10	216.67	6370	4434	2222	512	25.3	1218.3

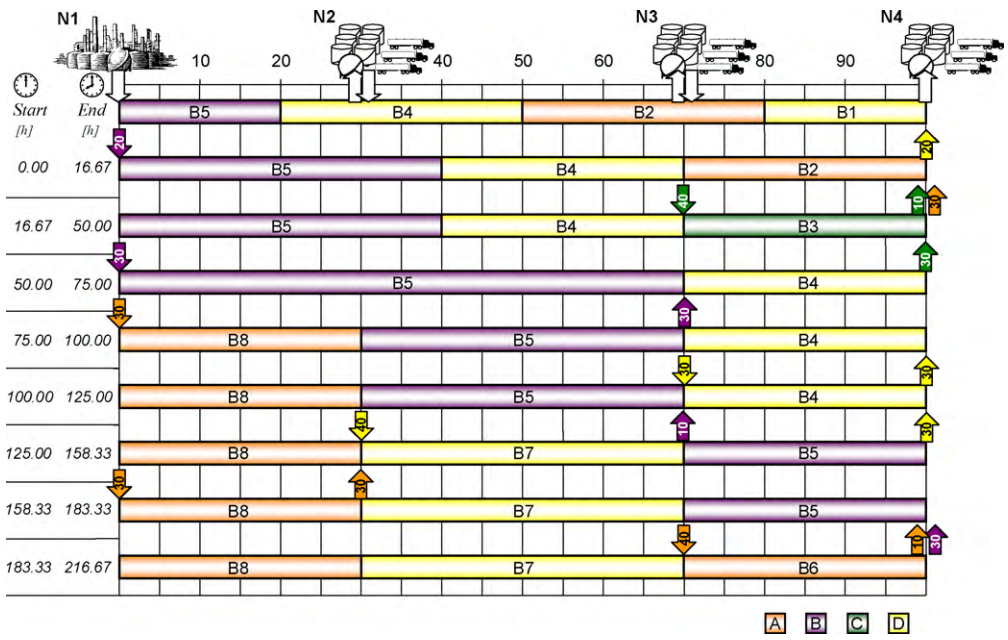

Fig. 17. Optimal sequential pipeline schedule for Example 3.

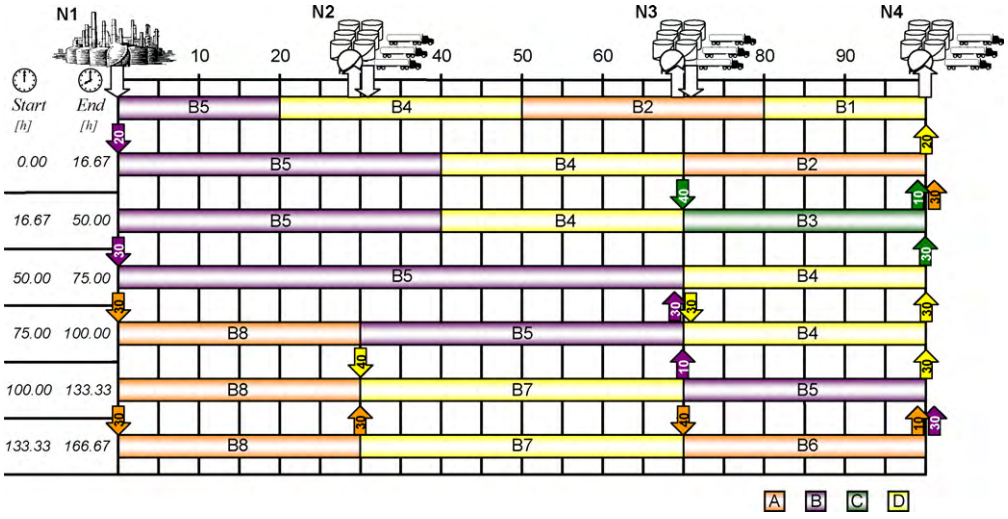

Fig. 18. Optimal parallel pipeline schedule for Example 3.

Table 14
Computational results for Example 3 using the parallel scheduling model.

$\|K\|$	Parallel schedules						
	Makespan (h)	Opt. cost (10^2\$)	Eqs.	Cont. variables	Binary variables	Iterations (10^6)	CPU time (s)
5	Infeasible	–	2239	1147	272	1.9	48.3
6	166.67	6370	2650	1368	320	1.2	45.3
7	166.67	6370	3061	1589	368	3.6	125.8
8	166.67	6370	3474	1810	416	6.9	267.6

6. Conclusions

A computationally efficient mixed-integer linear formulation for the operational scheduling of pipeline networks with simultaneous pumping runs at multiple input terminals has been presented. The proposed mathematical model is restricted to multiple-source networks with unidirectional pipelines and a single conduit between each pair of adjacent terminals. Dual-purpose stations performing pump and/or strip tasks are considered. This single-level approach permits to determine the optimal sequence of batch injections at every input terminal, lot sizes, allowable pump rates, start/end times of every block of pumping runs, and batch allocation to receiving terminals all at once. In addition, the model is able to track the location and size of product lots in different pipelines over the time horizon. Flow-rate variations due to changes in pipeline diameter are easily handled. The proposed formulation includes simple mathematical constraints for planning blocks of non-interacting, simultaneous batch injections. They

merely specify that a pipeline can receive material from at most a single source that may be an adjacent upstream pipeline or the tank farm at the pipeline inlet. One of the major advantages of executing blocks of non-interacting parallel runs is the better utilization of pipeline transport capacity, and much less time to complete all product deliveries (i.e. the schedule makespan). Reciprocally, it is most likely to meet every depot demand by running a larger number of pumping runs within the specified horizon length. For this reason, the minimum total cost including backorder expenses has been selected as a problem target. Moreover, good feasible schedules are more rapidly discovered. Three illustrative examples have been successfully solved. To show the improvements that are obtained by moving from sequential to parallel pipelines schedules, the best solutions with and without blocks of parallel batch injections have been found. By comparing results, several interesting conclusions can be drawn. When blocks of parallel runs are permitted at large-size problem instances, the run set K, the model size, and especially the required CPU time all usually decrease. The computational cost substantially diminishes despite a significant increase in the number of batch injections over the time horizon. Moreover, high backorder costs associated to sequential schedules can be removed by allowing blocks of parallel runs. Such improvements are provided by the proposed formulation through a better coordination among input operations at the pipeline network.

Acknowledgments

Financial support received from FONCyT-ANPCyT under Grant PICT 01837, from CONICET under Grant PIP-2221, and from "Universidad Nacional del Litoral" under PACT 66 is fully appreciated.

References

Boschetto, S. N., Felizari, L. C., Yamamoto, L., Magatão, L., Stebel, S. L., Neves, F., Jr, et al. (2008). An integrated framework for operational scheduling of a real-world pipeline network. *Computer Aided Chemical Engineering*, *25*, 259–264.

Brooke, A., Kendrick, D., Meeraus, A., & Raman, R. (2006). *GAMS—A user's guide*. Washington, DC: GAMS Development Corporation.

Cafaro, D. C., & Cerdá, J. (2004). Optimal scheduling of multiproduct pipeline systems using a non-discrete MILP formulation. *Computers & Chemical Engineering*, *28*, 2053–2068.

Cafaro, D. C., & Cerdá, J. (2008a). Efficient tool for the scheduling of multiproduct pipelines and terminal operations. *Industrial & Engineering Chemistry Research*, *47*, 9941–9956.

Cafaro, D. C., & Cerdá, J. (2008b). Dynamic scheduling of multiproduct pipelines with multiple delivery due dates. *Computers & Chemical Engineering*, *32*, 728–753.

Cafaro, D. C., & Cerdá, J. (2009). Optimal scheduling of refined products pipelines with multiple sources. *Industrial & Engineering Chemistry Research*, *48*, 6675–6689.

García-Sánchez, A., Arreche, L. M., & Ortega-Mier, M. (2008). Combining simulation and tabu search for oil-derivatives pipeline scheduling. *Studies in Computational Intelligence*, *128*, 301–325.

Hane, C. A., & Ratliff, H. D. (1995). Sequencing inputs to multi-commodity pipelines. *Annals of Operations Research*, *57*(1), 73–101.

Jittamai, P. (2004). *Analysis of oil-pipeline distribution of multiple products subject to delivery time-windows*. Ph.D. dissertation, Department of Industrial Engineering, Texas A&M University, College Station, Texas.

Magatão, L., Arruda, L. V. R., & Neves-Jr, F. (2004). A mixed integer programming approach for scheduling commodities in a pipeline. *Computers & Chemical Engineering*, *28*, 171–185.

Mori, F. M., Luders, R., Arruda, L. V. R., Yamamoto, L., Bonacin, M. V., Polli, H. L., et al. (2007). Simulating the operational scheduling of a realworld pipeline network. *Computer Aided Chemical Engineering*, *24*, 691–696.

Moura, A. V., de Souza, C. C., Cire, A. A., & Lopez, T. M. (2008). Planning and scheduling the operation of a very large oil pipeline network. *Lecture Notes in Computer Science*, *5202*, 36–51.

Neves, F., Jr, Magatão, L., Stebel, S. L., Boschetto, S. N., Felizari, L. C., Czaikowski, D. I., et al. (2007). An efficient approach to the operational scheduling of a real-world pipeline network. *Computer Aided Chemical Engineering*, *24*, 697–702.

Rejowski, R., & Pinto, J. M. (2003). Scheduling of a multiproduct pipeline system. *Computers & Chemical Engineering*, *27*, 1229–1246.

Rejowski, R., & Pinto, J. M. (2004). Efficient MILP formulations and valid cuts for multiproduct pipeline scheduling. *Computers & Chemical Engineering*, *28*, 1511–1528.

Rejowski, R., & Pinto, J. M. (2008). A novel continuous time representation for the scheduling of pipeline systems with pumping yield rate constraints. *Computers & Chemical Engineering*, *32*, 1042–1066.

Relvas, S., Matos, H. A., Barbosa-Póvoa, A. P. F. D., & Fialho, J. (2007). Reactive scheduling framework for a multiproduct pipeline with inventory management. *Industrial & Engineering Chemistry Research*, *46*, 5659–5672.

Relvas, S., Matos, H. A., Barbosa-Póvoa, A. P. F. D., Fialho, J., & Pinheiro, A. S. (2006). Pipeline scheduling and inventory management of a multiproduct distribution oil system. *Industrial & Engineering Chemistry Research*, *45*, 7841–7855.

Sasikumar, M., Prakash, P. R., Patil, S. M., & Ramani, S. (1997). PIPES: A heuristic search model for pipeline schedule generation. *Knowledge-Based Systems*, *10*, 169–175.

Zyngier, D., & Kelly, J. D. (2009). Multi-product inventory logistics modeling in the process industries. In W. Chaovalitwongse, K. C. Furman, & P. M. Pardalos (Eds.), *Optimization and logistics challenges in the enterprise. Springer optimization and its applications series* (pp. 61–95). N.Y: Springer.