I&EC research
Industrial & Engineering Chemistry Research

ARTICLE

pubs.acs.org/IECR

Detailed Scheduling of Operations in Single-Source Refined Products Pipelines

Vanina G. Cafaro, Diego C. Cafaro, Carlos A. Méndez, and Jaime Cerdá*

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

ABSTRACT: The short-term scheduling of refined products pipelines is a very complex problem with many operational constraints to be considered. Available nondiscrete planning approaches just provide the input schedule and the set of aggregate product deliveries to be performed from in-transit lots to depots at every batch injection. To determine the sequence of individual cuts on each batch to be accomplished by the pipeline operator, it is still necessary to refine such an aggregate plan. To do so, new computational tools for efficiently generating the detailed schedule of single-source pipelines with multiple distribution terminals are presented. Two types of methodologies are proposed. On one hand, it is developed a continuous-time mixed-integer linear programming (MILP) formulation that seeks to minimize both the total flow restart volume and the number of single-delivery pumping runs over the planning horizon. In this way, substantial savings in energy consumption and pump maintenance costs are achieved. Effective solution strategies for the MILP model are also designed to deal with large pipeline scheduling problems. On the other hand, three different heuristic rules for selecting the receiving terminal are introduced. By applying those rules in combination with a discrete-event simulation model, not only alternative detailed schedules can be generated in a very short CPU time but also some of them are near-optimal solutions. Three instances of a case study aimed at finding the detailed schedule of a real-world single-source pipeline system are solved through the proposed optimization and discrete-event simulation methods. Results are analyzed to assess the quality of the generated solutions and the required computational costs.

1. INTRODUCTION

Liquid pipelines transport a wide range of petroleum products with the fewest number of releases and the lowest energy requirement in comparison with other transportation modes. The short-term operational planning of pipeline systems is a very complex task involving two major steps: (a) the generation of the pipeline schedule at an aggregated level and (b) the refinement of the aggregate planning to develop the detailed pipeline schedule carried out by the pipeline operator. At the upper level (a), the sequence of batch injections and batch features (product, batch size, and mean pump rate) are all defined. In addition, the aggregate product deliveries to depots during every pumping run are also determined. Finding the best aggregate schedule is a combinatorial problem aimed at timely satisfying terminal demands and simultaneously minimizing interface, pumping, and inventory carrying costs. Several continuous-time approaches were proposed to find the optimal input schedule for multiproduct pipelines conveying batches of different commodities from a single refinery to several destinations.[1−3] Although those formulations also provide the set of aggregate batch stripping operations to be done at distribution terminals during every pumping run, the detailed sequence of individual "cuts" to be accomplished by the pipeline operator is not given. An individual cut is characterized by a single receiving depot, a unique giving batch, and the delivery size. If the pipeline is operated on fungible mode (i.e., a single batch can have multiple destinations), it should also be specified which part of the batch will be diverted to each depot. By assuming a unique distribution terminal, the continuous approaches of Relvas et al.[4] and Cafaro and Cerdá[5] can develop a detailed pipeline schedule at once.

Generating a feasible, detailed output schedule that efficiently accomplishes the aggregate delivery plan defined at level (a) using a continuous approach is a difficult problem that has not been solved yet in a rigorous way. Generally, there are many ways to distribute the inputted batches among the assigned destinations. Such a task is performed at step (b) by refining the aggregate schedule to obtain the detailed sequence of individual lots leaving the pipeline during every batch injection, their lot sizes, and the assigned depots. Moreover, stage (b) provides the times at which pumps should be turned on/off and valves at depot tanks must be open/closed to accomplish the detailed delivery schedule. Its major operational goal is to reduce the weighted number of flow restarts in idle segments and the frequency of on/off pump switching to get savings on energy consumption and pump maintenance costs. This work introduces new computational tools based on optimization and discrete-event simulation approaches to perform the refinement of the aggregate pipeline schedule in a reliable and cost-effective manner.

1.1. Literature Review. Most of the available methodologies for generating a detailed pipeline operational schedule are based on discrete representations. Such approaches divide both the pipeline volume into a number of single-product packs and the planning horizon into many time intervals.[6−9] Most of them generally use a uniform time and volume partitioning scheme. However, a later contribution by Rejowski and Pinto[10] assumes that the scheduling horizon comprises time intervals of adjustable

Received: January 3, 2011
Accepted: March 24, 2011
Revised: February 25, 2011
Published: April 15, 2011

dx.doi.org/10.1021/ie200007a | *Ind. Eng. Chem. Res.* 2011, 50, 6240–6259

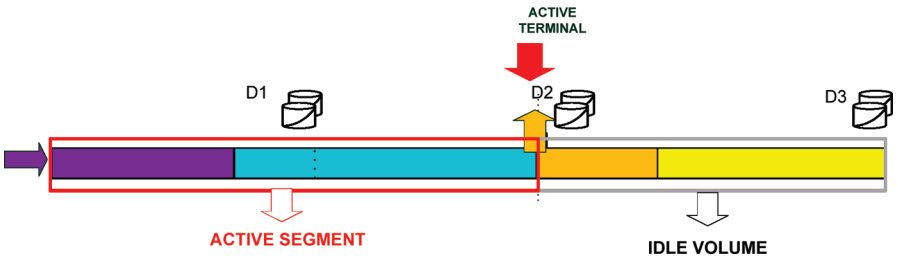

Figure 1. Active and idle pipeline segments while injecting a new batch.

duration to allow changes in the pump rate. Moreover, each pipeline segment is composed of packs with equal or different prespecified sizes to account for variations in the pipeline diameter. On the other hand, several hybrid methodologies decomposing the pipeline scheduling process into a number of stages and applying different techniques at every stage have been published. Neves et al.[11] presented a decomposition approach for the planning of pipeline operations over a monthly horizon. The decomposition relied on a heuristic-based preprocessing block that accounts for product requirements at distribution terminals, production planning at oil refineries, and a number of typical lot sizes to propose a set of candidate product strings. In addition, the heuristic block provides time windows for pump and delivery operations at every terminal. Afterward, the preprocessed information is used by a continuous-time MILP formulation to determine the exact start/finish times of batch input and delivery operations. 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.[12] developed a discrete-event simulation model for the scheduling of pumping and stripping operations in a real-world pipeline network. The network consists of a series of single lines that connect multiple refineries, harbors, and distribution centers among themselves and transport many oil derivatives.

Another hybrid approach that combines a randomized constructive heuristic with novel constraint programming (CP) models was reported by Moura et al.[13] It comprises the so-called planning and scheduling 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. Therefore, it assumes that the pipeline operates on batch mode, with every product lot having a single destination. The scheduling phase takes the proposed set of delivery orders and determines the sequence and start times of pumping operations at every source node to meet the promised delivery dates. This stage was implemented through a pair of CP models. García-Sánchez et al.[14] presented a hybrid methodology that combines tabu search and a discrete-event simulation model for addressing a real-world pipeline scheduling problem. The tabu search technique is used to improve nonoptimal pipeline schedules that are subsequently tested by the simulation model. Product batches are divided into a number of equal-sized packs whose destinations are predefined at the time they are injected into the pipeline network. Boschetto et al.[15] reformulated the hybrid approach of Neves et al.[11] using a different decomposition strategy now involving three blocks: (i) a resource-allocation block generating candidate sequences of batch injections, (ii) a preanalysis block specifying the precise volumes to be pumped from source nodes and received at destination nodes, and also providing the earliest start/finish times for stripping operations at every receiving terminal, and (iii) an MILP model determining the exact timing of pump and delivery operations at each depot. In contrast to continuous-time formulations, these approximate hybrid approaches provide very detailed output schedules.

1.2. Activated and Stopped Pipeline Volumes at Every Pumping Operation. As stated by Hane and Ratliff,[16] the pipeline energy consumption and the pump maintenance costs are mostly determined by the number of flow restarts in idle pipeline segments. Certainly, the cost incurred is not caused by the flow stoppage but for the restart of the fluid movement in some pipeline sections. Figure 1 shows a typical multiproduct pipeline with a single source at the origin and three distribution terminals: D1, D2, and D3.

When a product delivery occurs at depot D2 while pumping a new batch into the pipeline, the flow must be slowed down or completely stopped downstream of the receiving terminal D2. In the latter case, the active pipeline segments featuring a finite flow are those connecting the input station to terminal D2, while the section D2−D3 remains idle. If the next delivery is allocated to the downstream depot D3 (farther from the origin), the flow should be restarted in the segment connecting depots D2 and D3. Hence, the stoppage of pipeline segments can be extremely costly when the volume of fluid contained in such idle ducts must regain its lost momentum. The associated pumping costs, usually called flow restart costs, grow with the volume of product to put in motion. In case the next product delivery is allocated to the upstream depot D1 (closer to the origin) the flow in the segments joining D1 and D3 is stopped, and consequently there is no restart cost. This latter case clearly shows that the extra energy consumption is not linked to pipeline stoppages but to flow restarts. Moreover, only downstream deliveries become important at the detailed scheduling stage (b) because upstream ones never cause flow restarts (Hane and Ratliff).[16] Therefore, the decision on where to remove product from the line is an important economic issue. A practical criterion generally applied by pipeline operators to evaluate a detailed output schedule is the number of flow stoppages at pipeline segments that must resume operations later to perform downstream deliveries. Indeed, the cost of a flow stoppage is related to how much energy will be required to restart the fluid motion in the ducts, that is, the restart weighting cost. Such a weight factor is proportional to the volume of the pipeline section where fluid movement is to be restored. Then, the total restart volume over the time horizon is the goal to be minimized at the lower level (b) to cut down both the pipeline energy usage and the pump maintenance costs.

Two types of methodologies are introduced to develop a detailed pipeline schedule that accomplishes the aggregate delivery plan at low restart cost. One of them is based on a rigorous mixed-integer linear mathematical programming (MILP) formulation,

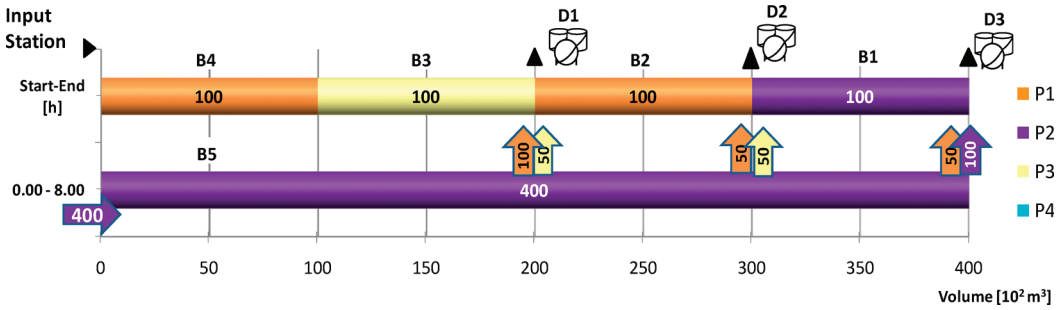

Figure 2. Aggregate pipeline schedule for the motivating example.

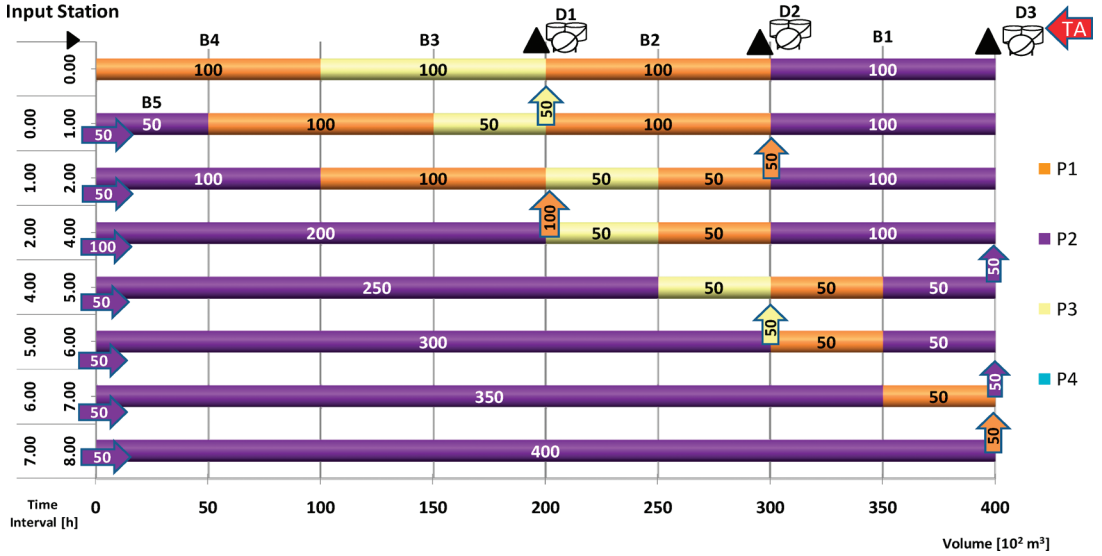

Figure 3. Detailed pipeline schedule favoring deliveries to the nearest-to-the-origin terminal.

while the other applies a discrete-event simulation model together with heuristic rules to choose the most convenient receiving terminal to be activated next. Because the size of the MILP model significantly grows with the length of the scheduling horizon, alternative solution techniques are presented to reduce the computational cost and still find a near-optimal detailed schedule. Three instances of a real-world case study have been tackled to find the detailed pipeline schedule of delivery operations by using both the optimization and discrete-event simulation approaches, and the alternative solution strategies for the MILP formulation. The obtained solutions are subsequently analyzed to compare their quality in terms of the total restart volume and the required CPU time.

2. MOTIVATING EXAMPLE

The pipeline system showed in Figure 2 transports three commodities from a single refinery to three distribution terminals. The first line in Figure 2 depicts the location of every batch inside the pipeline at the start of the time horizon. The initial linefill consists of a sequence of four batches [B4(P1)−B3(P3)−B2(P1)−B1(P2)], all having a volume of 100 units and containing the products indicated between parentheses. The following line in Figure 2 illustrates the pipeline state after completing the injection of batch B5 conveying 400 units of product P2, i.e., at time 8.00 h. This input operation is represented by a right arrow at the origin. It can also be observed a series of up arrows standing for the set of product deliveries to be accomplished while inserting B5. This is the type of results provided by the optimization model of Cafaro and Cerdá.[1,2] As shown in the second line of Figure 2, it not only generates an input schedule but also defines the set of aggregate stripping operations to be accomplished during the batch injection. Due to liquid incompressibility, the volume injected at the pipeline origin must be equal to the sum of product flows diverted to the receiving depots: 400 = 100 + 50 + 50 + 50 + 50 + 100. However, there are many ways to accomplish such aggregate product deliveries. For instance, one alternative is to prioritize deliveries to the nearest terminal to the origin, as depicted in Figure 3. In this way, the operator will choose a delivery sequence that hopefully removes some volumes of products from the line first at depot D1, then at D2, and finally at D3. This delivery sequence would tend to restore the flow in each pipeline segment only once to minimize the number of restarts. This "forward" delivery pattern cannot always be implemented by prioritizing the nearest depot to the origin because very often the giving batch does not already reach that terminal. As shown in Figure 3, this is the case arising in the second delivery operation.

We assume that the whole pipeline system was active just before the insertion of the new batch B5, and the last delivery occurred at depot D3. The aggregate pipeline schedule includes two deliveries to the nearest terminal D1: 50 units of P3 and 100 units of P1 (see the second line of Figure 2). Because batch B3

dx.doi.org/10.1021/ie200007a |Ind. Eng. Chem. Res. 2011, 50, 6240–6259

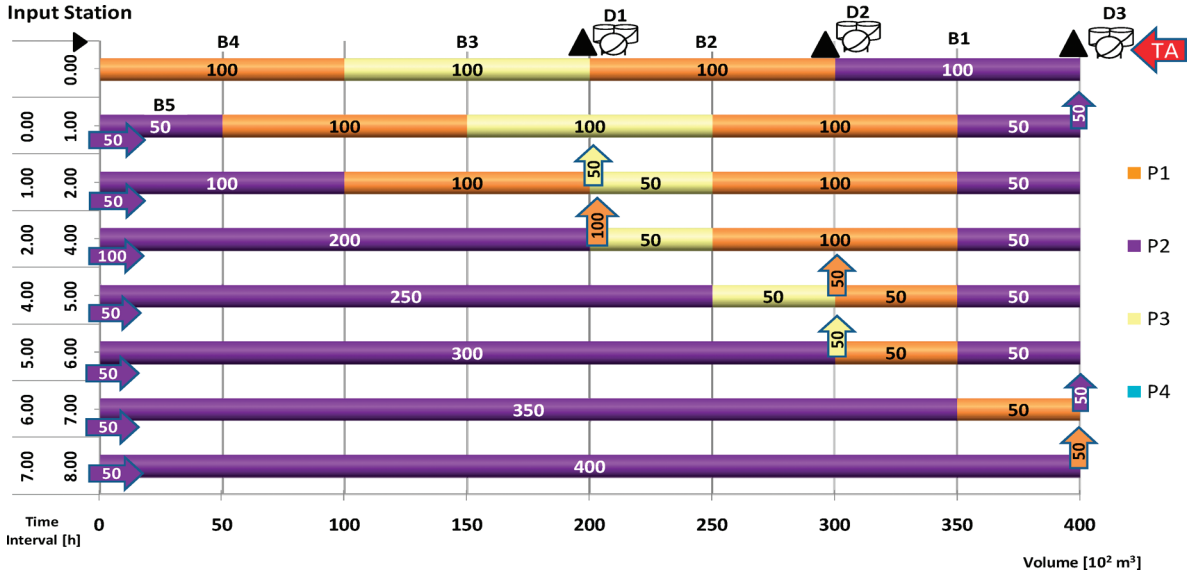

Figure 4. Optimal detailed pipeline schedule for the motivating example.

containing P3 has already arrived at D1, the delivery of product P3 to the nearest-to-the-origin depot (D1) is first performed by injecting 50 units of batch B5 (P2) at the input station. During that operation ending at time t = 1 h, the flow through the segments D1−D2 and D2−D3 is stopped. Next, the withdrawal of 100 units of P1 at depot D1 from batch B4 would be favored, but B4 does not still reach the location of D1 at t = 1 h (see the second line of Figure 3). Therefore, the next closest-to-origin terminal D2 is prioritized because the delivery of product P1 from batch B2 to depot D2 is a feasible operation. As a result, the flow in the segment D1−D2 should be restarted by pumping 50 additional units of batch B5 into the line.

After completing the removal of P1 at terminal D2 (time t = 2 h), batch B4 is in a proper location to supply product P1 to depot D1. Then, the flow in the pipeline segment D1−D2 is stopped and 100 further units of B5 are inputted at the origin. Once again, the second delivery to depot D2 cannot be next implemented because the giving batch B3 does not reach the location of D2. Consequently, terminal D3 is activated to receive the first 50 units of P2 from batch B1 by inserting a similar volume of B5 at the origin, and the flow in pipes D1−D2 and D2−D3 is restarted. When the partial delivery of P2 to D3 has ended (time t = 5 h), batch B3 has reached depot D2. Hence, the delivery of 50 units of P3 from B3 to D2 is next performed, and the flow in the section D2−D3 is stopped again. After that, the delivery of P2 from B1 to D3 is resumed by pumping 50 more units of B5 at the input station, and the flow in pipe D2−D3 is restored. Finally, 50 units of P1 are withdrawn at depot D3 from batch B2, and no flow restart is necessary.

Overall, there are two restarts in the segment D1−D2 and a similar number in D2−D3. Moreover, the aggregate delivery of 100 units of P2 from B1 to depot D3 requires a pair of nonconsecutive cut operations. Consequently, a string of seven short pumping runs (as many as the number of cut operations) is executed to inject the whole new batch B5 into the line. Both pipeline segments D1−D2 and D2−D3 have a volume of 100 units. Therefore, the total volume to be restarted is given by $2 \times 100 + 2 \times 100 = 400$ (10^2 m^3). Hane and Ratliff[16] stated that a lower bound on the overall activated volume is achieved if each of the pipelines connecting the initial active terminal (D3) to the farthest receiving terminal (D3) is restarted only once during the injection of batch B5, i.e., a volume equal to zero. A better solution for the motivating example is shown in Figure 4. It involves just one flow restart in the segment D1−D2 (line 5 of Figure 4) and one restart in the section D2−D3 (line 7 of Figure 4), that is, an overall activated volume of 200 units (in 10^2 m^3). It is a better detailed schedule from the restart cost viewpoint. The volume of fluid being activated to accomplish the specified product deliveries has been reduced by half.

In summary, every batch injection in the detailed pipeline schedule generally consists of a sequence of shorter pumping runs, each one associated to a different "cut" or stripping operation. Often, an aggregate delivery is accomplished by performing two or more nonconsecutive cut operations. As a result, the number of detailed pumping runs associated to a batch injection may exceed the number of related aggregate product deliveries.

To evaluate the quality of the detailed schedule, the notion of total activated volume (AV) has been introduced to measure the flow restart costs. AV is the accumulated volume of idle pipeline segments throughout the planning horizon where the fluid motion should be restored to carry out downstream cut operations. In other words, it is the accumulated increase of the active pipeline section. An equivalent criterion is the total stopped volume (SV) representing the accumulated volume of active segments where the flow is stopped. Therefore, SV is the accumulated reduction of the active pipeline section all over the time horizon. In Figure 4, there is just a single occasion when the length of the active pipeline section is reduced because of the flow stoppage in segments D1−D2 and D2−D3 (see line 3 of Figure 4). In the rest of the planning horizon, the active pipeline section never decreases, and SV = 200 (10^2 m^3). Note that depot D3 is not only the active terminal at t = 0 but also the final receiving terminal. It can be easily proven that the equality AV = SV always holds at the optimum if the initial and final active terminals are the same (for instance, depot D3 in Figure 4). Indeed, one of them takes the least value when the other criterion is minimized and vice versa. However, their minimum values differ in the content of the pipeline segments connecting the

dx.doi.org/10.1021/ie200007a |*Ind. Eng. Chem. Res.* 2011, 50, 6240–6259

initial and the latest receiving terminals; that is, $\Delta = \sigma_{final} - \sigma_{initial}$, with σ representing the volume coordinate of each terminal from the origin. More precisely, AV = SV + Δ at any detailed schedule.

After listing the problem assumptions in section 3, the following sections present a rigorous MILP optimization model providing the best detailed pipeline schedule, and decomposition-based strategies to efficiently find near-optimal or even optimal solutions in much lesser CPU times. Later, section 6 describes a discrete-event simulation approach and introduces the priority rules applied to select the next active terminal.

3. PROBLEM ASSUMPTIONS

Both the optimization and the discrete-event simulation approaches have been developed on the basis of the following assumptions:

(a) A trunk pipeline transporting several commodities from a single input station to several distribution terminals located along the line is considered.

(b) A feasible aggregate schedule of pipeline operations found through a continuous-time approach[2] is available.

(c) The detailed pipeline schedule must refine the input and output operations specified by the aggregate planning process.

(d) At the detailed level, whenever a pumping run is performed at the input station, there is at most one receiving terminal and a single giving batch.

(e) The flow in the pipeline section located downstream of the receiving depot is completely stopped while making the cut operation.

(f) Pumping energy consumption and maintenance costs are directly related to both the weighted number of flow restarts in idle pipeline sections and the number of on/off pump switching.

Assumption (d) is included because the discrete-event simulation model presented in section 6 does not consider simultaneous deliveries to multiple terminals during a batch injection. In line with assumption (d), the MILP approach described in section 4 also supposes a sequential pattern of deliveries to terminals during a pumping run to make a fair comparison between the two methodologies.

4. OPTIMIZATION APPROACH FOR THE DETAILED SCHEDULING OF SINGLE-SOURCE PIPELINES

We assume that the aggregate pipeline schedule has already been found. Hence, the following information is given: (a) the ordered set of batches $I = I^{old} \cup I^{new} = I^{old} \cup \{i_1, i_2, ..., i_n\}$ involving the initial linefill and the sequence of new lots to be inputted at the pipeline origin; (b) the ordered set $P = \{P_{i1}, P_{i2}, ..., P_{in}\}$ denoting the product $P_{i'}$ transported by every new lot $i' \in I^{new}$; (c) the subset of terminals $J_{i,i'}$ including the depots $j \in J$ receiving some amount of product from batch $i \in I$ during the injection of the new lot $i' \in I^{new}$ ($i' \geq i$); (d) the original size of the new batch $i' \in I^{new}$ represented by $qq_{i'}$; (e) the size $dd_{i,j}^{(i')}$ of the aggregate delivery of product from batch $i \in I$ to depot $j \in J_{i,i'}$ while injecting the lot $i' \in I^{new}$; (f) the volumetric coordinate σ_j of every terminal $j \in J$ from the origin; (g) the starting time ($st_{i'}$) and the completion time ($ft_{i'}$) for the injection of batch $i' \in I^{new}$; (h) the feasible pump rate range [$vb_{min}^{(i')}$; $vb_{max}^{(i')}$] for the injection of batch $i' \in I^{new}$ around the mean value adopted at the aggregate level.

4.1. Problem Variables. To formulate the detailed pipeline scheduling problem, it will be defined by the ordered set $K = K_{i1} \cup K_{i2} \cup ... \cup K_{in}$ representing the sequence of disaggregated ("detailed") pumping operations to be performed over the planning horizon. The elements $\{i_1, i_2, ..., i_n\} \in I^{new}$ are new batches to be injected in the pipeline. The subset K_{i1} stands for the string of input operations required to pump the whole new batch $i1$ into the pipeline. From the chronological viewpoint, the pumping operation $(k - 1) \in K$ immediately precedes run k. Moreover, it is said that the element $k \in K$ is a fictitious run if the injection is never executed. Though the cardinality of every subset $K_{i'}$ is not known beforehand, a minimum value for $|K_{i'}|$ is given by the number of aggregate deliveries to be accomplished during the injection of batch i'; i.e., $|K_{i'}| \geq \Sigma_{i \leq i'} |J_{i,i'}|$. Consequently, a lower bound on $|K|$ is $\Sigma_{i'}\Sigma_{i \leq i'} |J_{i,i'}|$. In addition, several sets of binary and continuous variables similar to those used for modeling the aggregate pipeline scheduling problem are to be defined. They are (i) the binary variable u_k standing for the existence of run k, (ii) the binary variable $x_{i,j}^{(k)}$ denoting the execution of a stripping operation on batch i to depot j during run k, if $x_{i,j}^{(k)} = 1$; (iii) the volume of fluid injected in the line during pumping run k (Q_k); (iv) the length (L_k) and the completion time (C_k) of every pumping run k; (v) the size of the cut on batch i to depot j during the execution of run k ($D_{i,j}^{(k)}$); (vi) the volume of batch i at the completion time C_k ($W_i^{(k)}$); and (vii) the front coordinate of batch i at time C_k ($F_i^{(k)}$). Besides, two other continuous variables are introduced to measure the quality of the detailed schedule: (viii) the activated volume (AV_k) representing the increase of the active pipeline section during run k; and (ix) the stopped volume (SV_k) denoting the reduction of the active pipeline section when performing run k.

4.2. Problem Constraints. Four types of problem constraints are considered: (1) pumping run constraints defining features of detailed input operations (length, timing, and injected volume); (2) batch constraints tracking the content and location of every lot moving along the pipeline; (3) product delivery restraints controlling the feasibility and the size of delivery operations; and (4) restrictions defining the values of the activated and stopped pipeline volumes at every run.

4.2.1. Pumping Run Constraints

Sequencing Constraints. The detailed pumping run k can start only if the previous one $(k - 1)$ has ended.

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

Starting Time for the First Element of the Set $K_{i'}$ Injecting the New Batch i'. The first pumping run of a new lot $i' \in I^{new}$ must not start before time $st_{i'}$ specified for the injection of batch i' at the aggregate level.

$$C_k - L_k \geq st_{i'} \quad \forall\ i' \in I^{new},\ k = first(K_{i'}) \tag{2}$$

Upper Bound on the Completion Time of Every Run $k \in K_{i'}$. Every run $k \in K_{i'}$ must never end after the completion time $ft_{i'}$ specified for the injection of batch $i' \in I^{new}$ at the aggregate operational schedule.

$$C_k \leq ft_{i'} \quad \forall\ i' \in I^{new},\ k \in K_{i'} \tag{3}$$

Volume of Product Injected by Run k and Its Length L_k. We assume that the input operation k is related to the injection of batch i' ($k \in K_{i'}$). If $u_k = 0$, run k is a fictitious element. Then, its length is null ($L_k = 0$), and no volume of product is injected in the line ($Q_k = 0$). Otherwise, the duration of the run must be within the allowed length range [l_{min}, $l_{max}^{(i')}$], and the pump rate should belong to the feasible interval [$vb_{min}^{(i')}$, $vb_{max}^{(i')}$] around

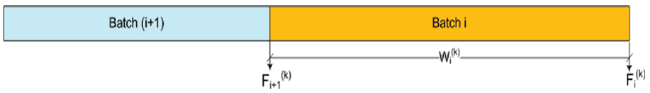

Figure 5. Relationship between the front coordinates of adjacent batches at the end of run k.

the mean value adopted for the injection of batch i' at the aggregate level. Such constraints on the values of L_k and Q_k are enforced through inequalities 4 and 5. The parameter l_{min} is a very small value while the upper bound $l_{max}^{(i')}$ is given by

$$l_{max}^{(i')} = \max_{\substack{i \le i' \\ j \in J_{i,i'}}} \{D_{i,j}^{(i')}\} / vb_{min}^{(i')}$$

Then,

$$vb_{min}^{(i')} L_k \le Q_k \le vb_{max}^{(i')} L_k \quad \forall\ i' \in I^{new},\ k \in K_{i'} \tag{4}$$

$$l_{min} u_k \le L_k \le l_{max}^{(i')} u_k \quad \forall\ i' \in I^{new},\ k \in K_{i'} \tag{5}$$

Fictitious Runs Placed at the End of Each Ordered Set $K_{i'}$. To avoid multiple equivalent solutions, fictitious pumping runs related to the injection of a new batch i' should always arise at the end of the ordered set $K_{i'}$.

$$u_k \le u_{k-1} \quad \forall\ i' \in I^{new},\ k \in K_{i'},\ k > \text{first}(K_{i'}) \tag{6}$$

Fulfillment of the Aggregate Input Schedule. The total volume pumped into the pipeline while performing the sequence of runs $k \in K_{i'}$ must be equal to the original size of lot i' at the aggregate input schedule; i.e., $qq_{i'}$.

$$\sum_{k \in K_{i'}} Q_k = qq_{i'} \quad \forall\ i' \in I^{new} \tag{7}$$

4.2.2. Batch Constraints

Relationship between the Front Coordinates of Two Consecutive Batches at the End of Run k. Because lot $(i+1)$ is introduced immediately after batch i in the line, it moves right behind batch i all along the pipeline (see Figure 5). Then, the front coordinates of both lots will be related by eq 8, where $W_i^{(k)}$ is the size of batch i at the completion time of run $k \in K_{i'}$ injecting a later lot $i' > i$.

$$F_{i+1}^{(k)} + W_i^{(k)} = F_i^{(k)} \quad \forall\ i \in I,\ i' \in I^{new},\ i' \ge i+1,\ k \in K_{i'} \tag{8}$$

Size of New Batch $i' \in I^{new}$ at the End of Run $k \in K_{i'}$ Inserting Lot i'. The content of lot i' at the end of run $k \in K_{i'}$ inserting a portion of batch i' in the line ($W_{i'}^{(k)}$) can be obtained from its old size $W_{i'}^{(k-1)}$ at the completion time C_{k-1} of the previous run $(k-1)$ by adding the volume injected through run k and subtracting the amount of product delivered from lot i' to terminals during the same run k. For $k = \text{first}(K_{i'})$, $W_{i'}^{(k-1)} = 0$.

$$W_{i'}^{(k)} = W_{i'}^{(k-1)} + Q_k - \sum_{j \in J_{i',i'}} D_{i',j}^{(k)} \quad \forall\ i' \in I^{new},\ k \in K_{i'} \tag{9}$$

Size of Batch $i \in I$ at the End of Run $k \in K_{i'}$ Inserting a Later Lot $i' > i$. In this case, batch i is not the one being injected by run $k \in K_{i'}$ (that is $i' > i$). Therefore, $W_i^{(k)}$ can be computed from $W_i^{(k-1)}$ by just subtracting the total volume transferred from batch i to terminals over run k. It may happen that run k is never accomplished ($u_k = 0$). As a result, $W_i^{(k)}$ will be equal to the content of lot i at the completion of the previously performed run, because Q_k and $D_{i,j}^{(k)}$ are both equal to zero for any fictitious input operation. For $k = 1$, lot i belongs to the starting linefill and $W_i^{(k-1)}$ is the initial content wo_i of the old batch i.

$$W_i^{(k)} = W_i^{(k-1)} - \sum_{j \in J_{i,i'}} D_{i,j}^{(k)} \quad \forall\ i \in I,\ i' \in I^{new},\ i' > i,\ k \in K_{i'} \tag{10}$$

4.2.3. Product Delivery Constraints

Bound on the Size of Product Deliveries. The volume of product stripped from batch $i \in I$ at some terminal $j \in J_{i,i'}$ during run $k \in K_{i'}$ ($D_{i,j}^{(k)}$) should belong to the feasible range $[d_{min}, dd_{i,j}^{(i')}]$. The parameter d_{min} is a very small value while the upper bound $dd_{i,j}^{(i')}$ is the size of the aggregate delivery to terminal $j \in J_{i,i'}$ from batch i during the injection of lot $i' \ge i$.

$$d_{min} x_{i,j}^{(k)} \le D_{i,j}^{(k)} \le dd_{i,j}^{(k)} x_{i,j}^{(k)} \quad \forall\ i \in I,\ i' \in I^{new},\ i' \ge i,\ j \in J_{i,i'},\ k \in K_{i'} \tag{11}$$

At Most a Single Product Delivery While Performing an Input Operation. Because of the liquid incompressibility, a volume of product similar to the one injected by pumping run k is sent out of the pipeline to terminals. However, simultaneous deliveries from the pipeline to multiple terminals cannot be made during a detailed input operation (see Problem Assumptions). Constraint 12 just permits a single batch stripping and a single receiving terminal at every run. In other words, the withdrawal of a refined product at a unique terminal from a single batch during a nonfictitious run is only allowed. Then, the cardinality of the set $K_{i'}$ can never be lower than the number of aggregate deliveries to be accomplished while injecting lot i'; i.e.,

$$\sum_{\substack{i \in I \\ i \le i'}} |J_{i,i'}|$$

If run k is never performed ($u_k = 0$), then no product delivery is carried out.

$$\sum_{\substack{i \in I \\ i \le i'}} \sum_{j \in J_{i,i'}} x_{i,j}^{(k)} = u_k \quad \forall\ i' \in I^{new},\ k \in K_{i'} \tag{12}$$

Feasibility Conditions for a Product Delivery. To remove some amount of product from lot i at terminal $j \in J_{i,i'}$ during a nonfictitious run $k \in K_{i'}$, the front coordinate of batch i at the end of run k must never be lower than the depot location σ_j; i.e., $F_i^{(k)} \ge \sigma_j$. Then,

$$F_i^{(k)} \ge \sigma_j\, x_{i,j}^{(k)} \quad \forall\ i \in I,\ i' \in I^{new},\ i' \ge i,\ j \in J_{i,i'},\ k \in K_{i'} \tag{13}$$

Moreover, the back-coordinate of batch i at the start of run k should be lower than σ_j by at least the volume $D_{i,j}^{(k)}$ ($\le dd_{i,j}^{(i')}$) delivered to terminal j during run $k \in K_{i'}$; i.e., $[F_i^{(k-1)} - W_i^{(k-1)}] \le \sigma_j - D_{i,j}^{(k)}$. If one of the feasibility conditions cannot be satisfied, then $x_{i,j}^{(k)} = 0$ and constraints 13 and 14 become redundant, and because of eq 11, the related product delivery is not accomplished ($D_{i,j}^{(k)} = 0$).

$$F_i^{(k-1)} - W_i^{(k-1)} + D_{i,j}^{(k)} \le \sigma_j + (pv - \sigma_j)(1 - x_{i,j}^{(k)}) \quad \forall\ i \in I,\ i' \in I^{new},\ i' \ge i,\ j \in J_{i,i'},\ k \in K_{i'} \tag{14}$$

dx.doi.org/10.1021/ie200007a |*Ind. Eng. Chem. Res.* 2011, 50, 6240–6259

Parameter pv is the total pipeline volume. Equation 14 does not account for product deliveries to upstream terminals j' (with $\sigma_{j'} < \sigma_j$) coming from the same batch i because just one delivery can be performed during run k.

Additional Upper Bound on the Size of a Product Delivery. So far, the volume $D_{i,j}^{(k)}$ transferred from batch i to terminal $j \in J_{i,i'}$ during run $k \in K_{i'}$ is bounded by constraints 11 and 14. These restrictions force the variable $D_{i,j}^{(k)}$ to never exceed $\mathrm{dd}_{i,j}^{(i')}$ or $[\sigma_j - (F_i^{(k-1)} - W_i^{(k-1)})]$, respectively. Besides, the volume $D_{i,j}^{(k)}$ must never be greater than the content of lot i at the end of the previous run $(k-1)$.

$$\sum_{j \in J_{i,i'}} D_{i,j}^{(k)} \le W_i^{(k-1)} \quad \forall\ i \in I,\ i' \in I^{\mathrm{new}},\ i' \ge i,\ k \in K_{i'} \tag{15}$$

Exact Balance between Input and Output Volumes during Run k. From the incompressibility property, the amount of product inputted by run k should be equal to the overall volume transferred from the pipeline to the selected terminal.

$$\sum_{\substack{i \in I \\ i \le i'}} \sum_{j \in J_{i,i'}} D_{i,j}^{(k)} = Q_k \quad \forall\ i' \in I^{\mathrm{new}},\ k \in K_{i'} \tag{16}$$

Fulfillment of Aggregate Product Deliveries. The total volume that is removed from batch i at terminal $j \in J_{i,i'}$ during the sequence of runs $k \in K_{i'}$ must be equal to the specified aggregate delivery $\mathrm{dd}_{i,j}^{(i')}$.

$$\sum_{k \in K_{i'}} D_{i,j}^{(k)} = \mathrm{dd}_{i,j}^{(i')} \quad \forall\ i \in I,\ i' \in I^{\mathrm{new}},\ i' \ge i,\ j \in J_{i,i'} \tag{17}$$

4.2.4. Activated and Stopped Pipeline Volumes. As already mentioned, the notion of "activated volume" has been introduced to measure the flow restart costs. The fluid motion in idle pipeline segments is restored when the next product delivery occurs at a downstream terminal, farther from the origin. Therefore, it is important to account for the coordinates of the receiving terminals j' and j during runs $(k-1)$ and k, respectively, to determine the pipeline volume activated by operation k (AV_k). If $\sigma_j > \sigma_{j'}$, a downstream delivery occurs and there will be an increase of the active pipeline volume caused by run k, equal to the following: $\mathrm{AV}_k = \sigma_j - \sigma_{j'}$. If instead the receiving terminal j is closer to the input station ($\sigma_j < \sigma_{j'}$), the flow in the segments connecting terminals j and j' is stopped. Then, there will be an increase of the idle pipeline volume caused by run k (SV_k), given by the following: $\mathrm{SV}_k = \sigma_{j'} - \sigma_j$. Note that σ_j is the volume between the pipeline origin and the location of depot j. When the receiving terminal changes at two consecutive runs, there will be a variation in either the pipeline activated or stopped volume. Moreover, no change in the destination with regard to run $(k-1)$ implies that the next run k will not produce an increase of either the activated or the stopped volume ($\mathrm{AV}_k = \mathrm{SV}_k = 0$).

If the farthest terminal is receiving product, the whole pipeline is active. On the contrary, the idle pipeline section takes a maximum length when the closest-to-the-origin depot is the selected destination. In this case, the rest of the pipeline starting from the nearest depot to the farthest terminal will remain idle. Then, it becomes clear that the selected series of receiving terminals strongly affects flow restarts and on/off pump switching costs.

Constraints 18, 19 and 20, 21 define lower bounds on the activated (AV_k) or stopped (SV_k) volumes caused by run k, respectively. Restrictions 18 and 20 assume $k > \mathrm{first}(K_{i'})$, while eqs 19 and 21 just apply to run $k = \mathrm{first}\ (K_{i'})$ injecting the first element of the new batch i'. In the latter case, the previous run $(k-1) \in K_{i'-1}$ is the last element associated with the injection of lot $(i'-1)$, and it may probably be fictitious. The current active terminal is the one actually receiving product when inserting the last element of batch $(i'-1)$ in the pipeline. As already mentioned, the cardinality of $K_{i'-1}$ should be at least equal to the number of aggregate deliveries to perform while injecting lot $(i'-1)$, i.e., $\sum_i |J_{i,i'-1}|$, with $i \le i'-1$. Therefore, the candidates for the last nonfictitious run of lot $(i'-1)$ are those elements of the set $K_{i'-1}$ occupying the final $[|K_{i'-1}| - \sum_i |J_{i,i'-1}| + 1]$ positions. In constraints 19 and 21, the subscript n screens all candidates for the last nonfictitious run of lot $(i'-1)$. Both restrictions become active for a particular run $(k-n)$ satisfying the following two conditions: (i) $u_{k-n} = 1$ and (ii) the intermediate runs $\{(k-n+1), (k-n+2), \ldots, (k-1)\}$ between runs $(k-n)$ and k are all fictitious; i.e., $u_{k-n+1} = u_{k-n+2} = \ldots = u_{k-1} = 0$. For instance, $n = 2$ accounts for the run $(k-2) \in K_{i'-1}$ right before $(k-1)$. If such a run $(k-2)$ is the last injecting lot $(i'-1)$, then $u_{k-2} = 1$ and $u_{k-1} = 0$. Therefore, constraints 19 and 21 both become active.

$$\mathrm{AV}_k \ge \sum_{\substack{i \in I \\ i \le i'}} \sum_{j \in J_{i,i'}} \sigma_j x_{i,j}^{(k)} - \sum_{\substack{i \in I \\ i \le i'}} \sum_{j' \in J_{i,i'}} \sigma_{j'} x_{i,j'}^{(k-1)}$$

$$\forall\ i' \in I^{\mathrm{new}},\ k \in K_{i'},\ k > \mathrm{first}(K_{i'}) \tag{18}$$

$$\mathrm{AV}_k \ge \sum_{\substack{i \in I \\ i \le i'}} \sum_{j \in J_{i,j'}} \sigma_j x_{i,j}^{(k)} - \sum_{\substack{i \in I \\ i \le i'-1}} \sum_{j' \in J_{i,i'-1}} \sigma_{j'} x_{i,j'}^{(k-1)} + \mathrm{pv}\Big(u_{k-n} - \sum_{l=k-n+1}^{k-1} u_l - 1\Big)$$

$$\forall\ i' \in I^{\mathrm{new}},\ k = \mathrm{first}(K_{i'}),\ n = 1, 2, \ldots, (|K_{i'-1}| - \sum_{\substack{i \in I \\ i \le i'-1}} |J_{i,i'-1}| + 1) \tag{19}$$

$$\mathrm{SV}_k \ge \sum_{\substack{i \in I \\ i \le i'}} \sum_{j' \in J_{i,i'}} \sigma_{j'} x_{i,j'}^{(k-1)} - \sum_{\substack{i \in I \\ i \le i'}} \sum_{j \in J_{i,i'}} \sigma_j x_{i,j}^{(k)}$$

$$\forall\ i' \in I^{\mathrm{new}},\ k \in K_{i'},\ k > \mathrm{first}(K_{i'}) \tag{20}$$

$$\mathrm{SV}_k \ge \sum_{\substack{i \in I \\ i \le i'-1}} \sum_{j' \in J_{i,i'-1}} \sigma_{j'} x_{i,j'}^{(k-1)} - \sum_{\substack{i \in I \\ i \le i'}} \sum_{j \in J_{i,i'}} \sigma_j x_{i,j}^{(k)} + \mathrm{pv}\Big(u_{k-n} - \sum_{l=k-n+1}^{k-1} u_l - 1\Big)$$

$$\forall\ i' \in I^{\mathrm{new}},\ k = \mathrm{first}(K_{i'}),\ n = 1, 2, \ldots, (|K_{i'-1}| - \sum_{\substack{i \in I \\ i \le i'-1}} |J_{i,i'-1}| + 1) \tag{21}$$

For $k = 1$ (the leading pumping run for the first batch injected in the line) the coordinate of the receiving terminal $\sigma_{j'}$ at the

dx.doi.org/10.1021/ie200007a |Ind. Eng. Chem. Res. 2011, 50, 6240–6259

previous run $k-1$ (the last one in the past horizon) is a problem datum. If instead the whole pipeline is idle at the initial time $t = 0$, then $\sigma_{j'} = 0$ for the run preceding $k = 1$.

4.3. Objective Function. The problem goal is to develop a detailed pipeline schedule that meets the aggregate input/output plan at minimum flow restart and on/off pump switching costs. This is achieved by minimizing the activated or stopped pipeline volume through the least number of single-delivery pumping runs. In the objective function 22, the parameter cs is the unit flow stoppage cost, ca is the flow restart cost per unit volume, and fco is the fixed cost associated with the execution of a single-delivery run.

$$\min z = \sum_{k \in K} ((\mathrm{cs})\mathrm{SV}_k + (\mathrm{ca})\mathrm{AV}_k + (\mathrm{fco})u_k) \qquad (22)$$

In summary, the proposed optimization approach is based on an MILP formulation comprising the set of constraints 1−21 and the objective function 22. When the aggregate operational plan includes a complete pipeline stoppage at some intermediate interval for maintenance or other purpose, the proposed approach should be sequentially applied to the aggregate schedule for every active period.

5. ALTERNATIVE STRATEGIES FOR SOLVING THE MILP FORMULATION

The size of the detailed scheduling formulation is directly related to the cardinality of the set K. The value of $|K|$ is strongly influenced by the length of the planning horizon ($h_{\max}$) because the number of batch injections may significantly grow. When $h_{\max}$ = 1 month and the problem formulation accounts for the whole set K at once, searching for the optimal solution can be extremely costly. For such cases, one can better apply some decomposition techniques to break the set K into a number of smaller subsets (K1, K2, ...), each one defining a different scheduling subproblem. In this way, one can tackle the large MILP formulation by solving a sequence of lower-size MILP subproblems.

Equation 7 imposes the so-called integrity condition on the definition of the subsets (K1, K2, ...). If some input operations related to the injection of the new batch i are included in the subset Kn, then all the runs inserting batch i should belong to Kn. Three alternative solution techniques are presented to solve the proposed MILP detailed scheduling formulation. They are as follows: (1) *the full decomposition (FD) algorithm*; (2) *the pair decomposition (PD) algorithm*; and (3) *the nondecomposition (ND) solution algorithm*.

A good detailed pipeline schedule can be developed by combining the optimal subproblem solutions for every batch injection. Though some coupling among subproblem decisions does exist, it is rather weak and has a short time range. In other words, the impact of the selected sequence of receiving terminals for the injection of batch i is mostly confined to the next batch injection $(i+1)$. This empirical finding is the fundamental basis of the so-called pair decomposition algorithm that usually yields the optimal solution to the original MILP detailed scheduling problem.

5.1. Full Decomposition Algorithm. To generate a suboptimal detailed pipeline schedule, the FD algorithm solves as many MILP subproblems as the number of batches to be pumped into the pipeline over the planning horizon. Each subproblem provides the refined pipeline schedule for a single batch injection

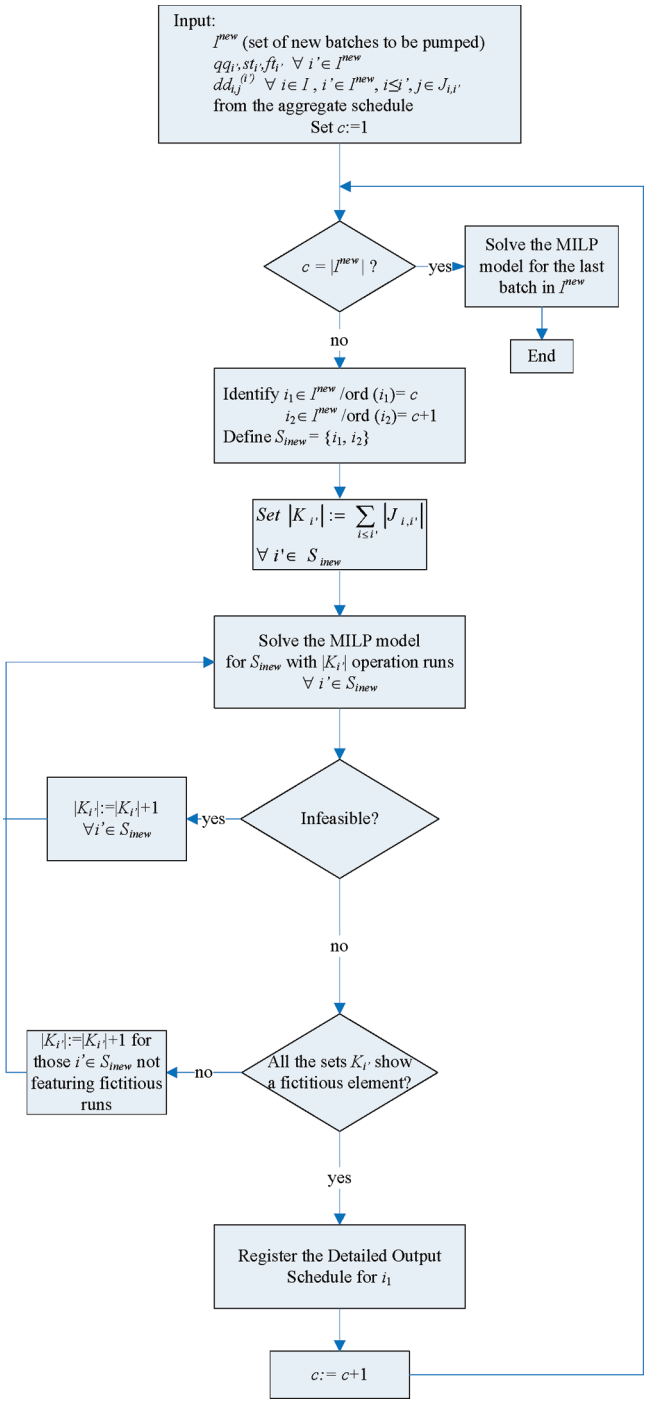

Figure 6. Schematic diagram of the PD solution algorithm.

($|I^{\mathrm{new}}| = 1$). The subproblem for the set K_{i1} should be solved before the one for K_{i2} (with $i1$ directly preceding $i2$) to know the active terminal at the last run of K_{i1}. A good solution to the original scheduling problem can be developed by recursively adding the detailed operational schedule for the injection of the new batch $(i+1)$ at the end of the one found for the pumping of batch i. Though the overall computational cost is drastically diminished, the development of the detailed schedule by considering the aggregate batch injections one-by-one has clear limitations. Its major drawback comes from the fact that the set of receiving

dx.doi.org/10.1021/ie200007a |*Ind. Eng. Chem. Res.* 2011, 50, 6240–6259

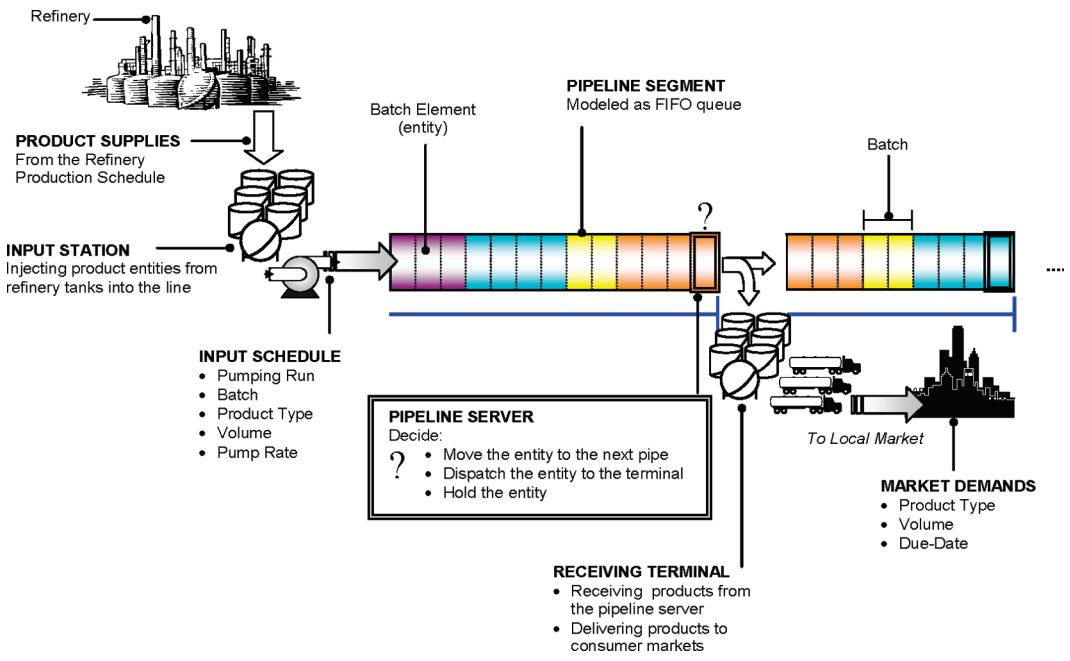

Figure 7. Major components of a discrete-event simulation model of pipeline operations.

terminals on the next pumping run $(i + 1)$ is completely ignored while determining the detailed schedule for the injection of batch *i*. Consequently, it usually leads to a suboptimal solution featuring a flow restart volume above the minimum value.

5.2. Pair Decomposition Algorithm. Similarly to the FD algorithm, the pair decomposition algorithm has to solve as many MILP subproblems as the number of aggregate batch injections. In this case, we deal with an ordered set of MILP subproblems, each one simultaneously determining the detailed input operations for two consecutive batch injections. It should first solve the scheduling subproblem for a run set K1 = $K_{i1} \cup K_{i2}$, then the one for K2 = $K_{i2} \cup K_{i3}$, and so on. In this manner, the sequence of active terminals during the injection of batch $(i + 1)$ is taken into account by the PD algorithm to develop the detailed schedule for the pumping of batch *i*. However, just the detailed output schedule for the insertion of batch *i* is adopted. The one for the injection of batch $(i + 1)$ can be modified when solving the next subproblem. In other words, the detailed pipeline schedule is gradually developed and updated as the time horizon rolls. For the case study presented in section 7, the PD solution strategy provides the optimal solution. In Figure 6, a schematic diagram of the PD algorithm is presented. The number of individual runs considered at any MILP subproblem is increased until no further improvement on the objective function is achieved. It is assumed that the optimal delivery schedule has been found when at least one element of every set K_i stands for a fictitious run.

5.3. Nondecomposition Algorithm. The ND algorithm solves the full MILP problem formulation at once. Then, it tackles a single MILP mathematical model that accounts for all planned batch injections. The number of individual operations is initially fixed at its lower bound $|K| = \Sigma_{i'}\Sigma_{i \le i'} |J_{i,i'}|$, i.e., the total number of aggregate deliveries, and then recursively increased by one until no further improvement is observed. If the full MILP model is solved to optimality, then the global optimal solution is found but the computational cost may be excessive.

6. HEURISTIC RULES FOR SELECTING THE RECEIVING TERMINAL

Cafaro et al.[17] introduced a discrete-event simulation model of a trunk pipeline transporting refined products from a single origin to multiple distribution terminals. It was developed on Arena[18] to simulate the pipeline operational plan generated by the continuous-time mathematical formulation of Cafaro and Cerdá.[2] The discrete-event simulation model permits one to (i) validate the proposed aggregate plan by developing a feasible detailed schedule and (ii) visualize the pipeline operations by means of a friendly animation interface illustrating the dynamics of the pipeline system over time. The simulation model structure involves three blocks, each one representing a main component of the pipeline system: the input station, the receiving terminals, and the pipeline segments. Figure 7 shows the model blocks together with key simulation elements, such as entities (elementary batch units) and resources.

6.1. Role of Priority Rules on a Discrete-Event Simulation Model of Pipeline Operations. The trunk pipeline can be regarded as a sequence of segments, each one connecting either an input to an output terminal or just a pair of depots between themselves. Moreover, it can be viewed as a multiserver queuing system with a single server at the extreme of every pipeline segment. In other words, each segment is a FIFO queue and the server at the pipe extreme permits the movement of batch elements, called entities, to either the associate depot or the next conduct. Since every pipeline segment is permanently full of liquid and has a constant volume, the length of any server queue will remain fixed throughout the whole time horizon. The actions that a server can take on the entity waiting for service are as follows: (i) no action, (ii) move it to the next pipe, or (iii) load it in a terminal tank.

The servers perform their tasks in a synchronized manner. When a new entity is injected in the pipeline, there are a limited number of servers that are eligible to dispatch an elementary

Table 1. Pipeline Section Capacities

pipeline section	volume capacity (10^2 m^3)	pipeline section	volume capacity (10^2 m^3)
refinery to terminal D1	400	terminal D3 to terminal D4	600
terminal D1 to terminal D2	250	terminal D4 to terminal D5	135
terminal D2 to terminal D3	250		

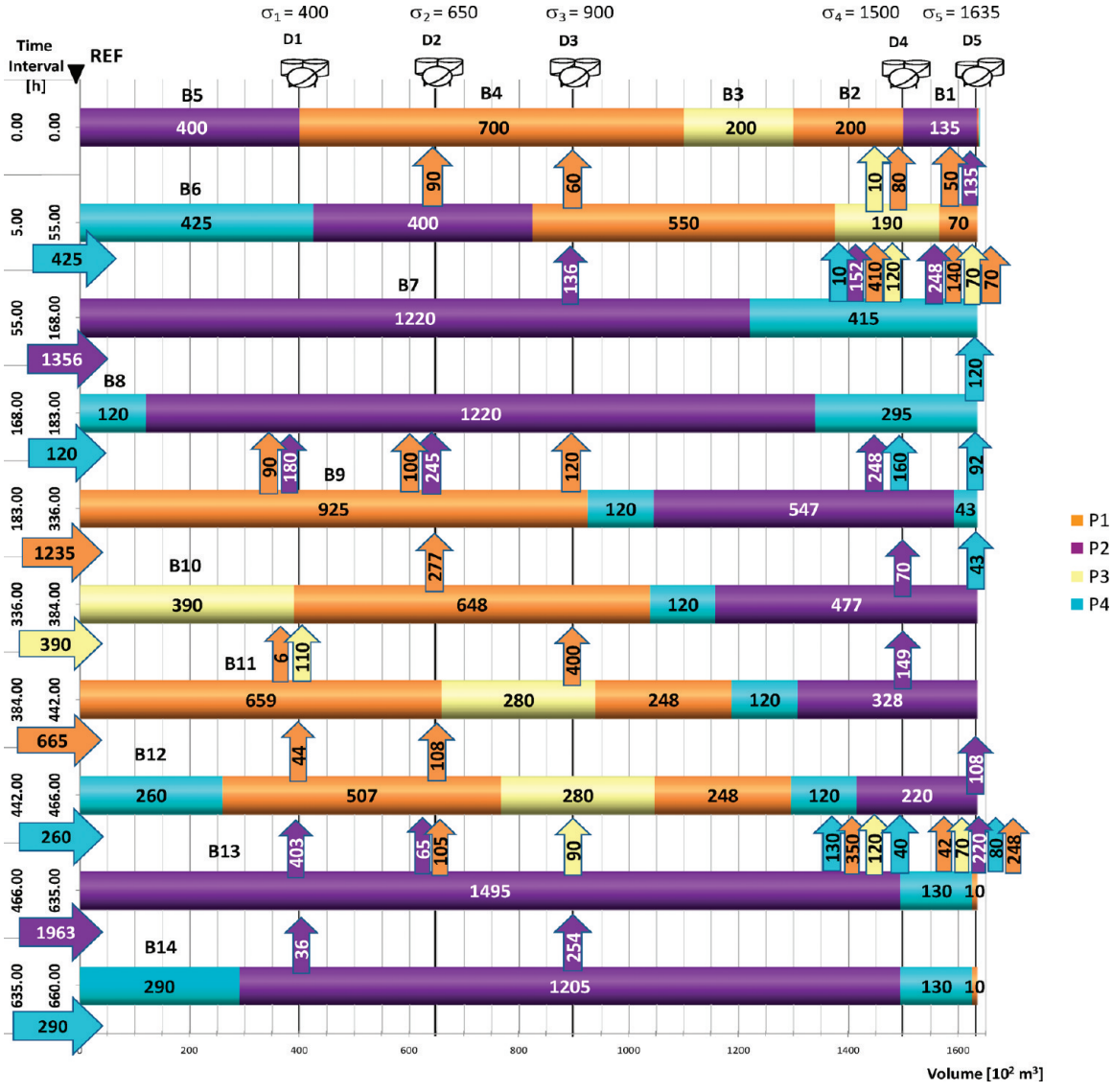

Figure 8. Aggregate pipeline schedule for the complete case study.

batch to the related terminal. Such a set of eligible servers is directly inferred from the aggregate delivery schedule provided by the optimization module. As a result, each server is aware that the first entity on its queue is or is not eligible for delivery to the associated terminal. If not, the batch element should either move to the next pipe or remain still at the same queue. For operational reasons, only one server can be active at any input event. If two or more servers are eligible for dispatching the leading entity to their related terminals, the simulation model should decide which one is chosen. To select the active server (or equivalently, the receiving terminal) at every pumping event, the simulation model applies some heuristics assigning different priorities to terminals. In case none of the eligible terminals is restrictive (see section 6.2), the one with the highest priority is selected as the receiving terminal.

6.2. Six-Step Algorithm for the Choice of the Receiving Terminal. Each time an entity is inputted in the pipeline system, the following six-step algorithm is performed:

(1) *Identify the eligible terminals at the current event k.* We assume that the current input event k is injecting an element of batch i', and the first entity on the server queue of terminal $j \in J_{i,i'}$ belongs to batch i. If the aggregate delivery from batch i to

Table 2. Aggregate Input Schedule for the Complete Case Study

batch	product	volume (10^2 m^3)	initial time (h)
B6	P4	425	5.0
B7	P2	1356	55.0
B8	P4	120	168.0
B9	P1	1235	183.0
B10	P3	390	336.0
B11	P1	665	384.0
B12	P4	260	442.0
B13	P2	1963	466.0
B14	P4	290	635.0

Table 3. Aggregate Product Deliveries While Pumping Batch B7 (in 10^2 m^3)

	delivered batch					
terminal	B2	B3	B4	B5	B6	B7
D3						136
D4		120	410	152	10	
D5	70	70	140	248		

terminal j during the injection of lot i' ($\mathrm{dd}_{i,j}^{(i')}$) has not been fully performed up to the event $(k-1)$, then the depot j is eligible to receive that entity from batch i. We suppose that $d_{i,j,(k-1)}^{(i')}$ is the amount already transferred to terminal j at event $(k-1)$. Then, depot j will be eligible only if $d_{i,j,(k-1)}^{(i')} < \mathrm{dd}_{i,j}^{(i')}$.

(2) *Identify the restrictive terminals at the current event k.* Suppose that terminal $j \in J_{i,i'}$ is eligible to receive an entity from batch i, and the current event k is introducing an element of batch i'. Since a unidirectional pipeline is considered, an entity that overpasses terminal j can no longer be transferred to it. If such an entity is absolutely necessary to meet its specified aggregate delivery, the resulting output schedule would be infeasible. Hence, such a terminal j is said to be restrictive, and any downstream terminal j' (located farther from the origin than depot j) should not be selected as the receiving depot. Otherwise, the specified aggregate delivery to terminal j cannot be fully satisfied. Therefore, only the eligible terminals located between the origin and depot j are feasible candidates for receiving terminal at the current event k, and the one with the highest priority is selected.

We assume that the parameter v represents the volume of every entity. To verify the existence of restrictive terminals, compute the number of entities of batch i located between the pipeline origin and the eligible terminal j at event $(k-1)$ pumping lot i' ($N_{i,j,(k-1)}^{(i')}$). The terminal j is restrictive whenever $[\mathrm{dd}_{ij}^{(i')} - d_{i,j,(k-1)}^{(i')}] = v\, N_{i,j,(k-1)}^{(i')}$; that is, the amount of product still to be transferred to complete the aggregate delivery $\mathrm{dd}_{i,j}^{(i')}$.

(3) *Identify the most restrictive terminal.* Among the restrictive terminals, select the most restricted one, i.e., the restrictive depot j closest to the origin. Eligible terminals j' with $\sigma_{j'} > \sigma_j$ can no longer be chosen as the receiving depot at event k.

(4) *Select the receiving terminal.* Among the eligible terminals located between the pipeline origin and the most restrictive terminal j, select the depot j^* with the highest priority. Choose j^* as the receiving terminal.

(5) *Execute the input and output operations.* Pump a new entity into the pipeline system and simultaneously transfer the entity of batch i waiting for service at the selected terminal j^*. Update the amount already transferred to terminal j from batch i: $d_{i,j,k}^{(i')} = d_{i,j,(k-1)}^{(i')} + v$.

(6) *Update the pipeline content.* Move the first entity on the queue of any server closer to the origin (with regard to j^*) to the next pipe. Return to step 1.

6.3. Priority Rules. To generate a detailed pipeline schedule using the discrete-event simulation model of Cafaro et al.,[17] three priority rules have been tested. They are as follows: (a) the *nearest-first (NF) rule*, (b) the *farthest-first (FF) rule*, and (c) the *nearest-to-the-current-terminal (NC) rule*. A brief description of the priority rules is given as follows.

(a) The nearest-first rule prioritizes the product delivery to the eligible terminal closest to the origin. If two alternative terminals are able to receive some amount of product from the line, the depot closer to the origin will be selected. In this way, no batch moving along the line will overpass a demanding terminal and restrictive terminals never arise.

(b) The farthest-first rule prioritizes product deliveries to the eligible terminal farthest from the origin. In this case, however, it is necessary to verify if some other terminals are restrictive. When some eligible terminals are restrictive, choose the one closest to the origin as the receiving terminal.

(c) The nearest-to-the-current-terminal rule prioritizes the eligible depot closer to the one being currently served as the destination for the next delivery. If there are only two eligible terminals and both are located at the same distance of the current active terminal, the upstream depot is selected. If eligible, the current terminal has obviously the highest priority. By favoring the nearest-to-the-current terminal, an important reduction in both the total pipeline restart and stop volumes, and a decrease in the number of stripping operations (turning on/off pump stations) are simultaneously obtained in most cases.

7. RESULTS AND DISCUSSION

The proposed optimization and discrete-event simulation approaches are both applied to a real-world case study first introduced by Rejowski and Pinto[6] and later solved by Cafaro and Cerdá.[1,2] This challenging transportation problem addresses the daily operation of a pipeline that transports almost 20% of Brazilian oil derivatives. The pipeline has a length of 955 km and connects the REPLAN refinery at Sao Paulo to five distribution centers. In fact, it is composed of five pipeline segments whose volumetric capacities are given in Table 1.

The case study involves the distribution of four commodities (gasoline, diesel, LPG, and jet fuel). We consider the problem of developing the detailed operational schedule for such a single-source pipeline system over a 4 week planning horizon. The aggregate schedule found through the dynamic optimization approach of Cafaro and Cerdá[2] is presented in Figure 8. The top row of Figure 8 describes the initial linefill, and the following lines illustrate the aggregate sequence of new batch injections. Problem data for the aggregate pipeline planning problem can be found in Cafaro and Cerdá.[2] It is observed that a sequence of nine new batches is to be injected at the input terminal starting with batch B6 and ending with lot B14 (see Figure 8). The product and the original size of every new batch, as well as the start time of the related input operation, are given in Table 2.

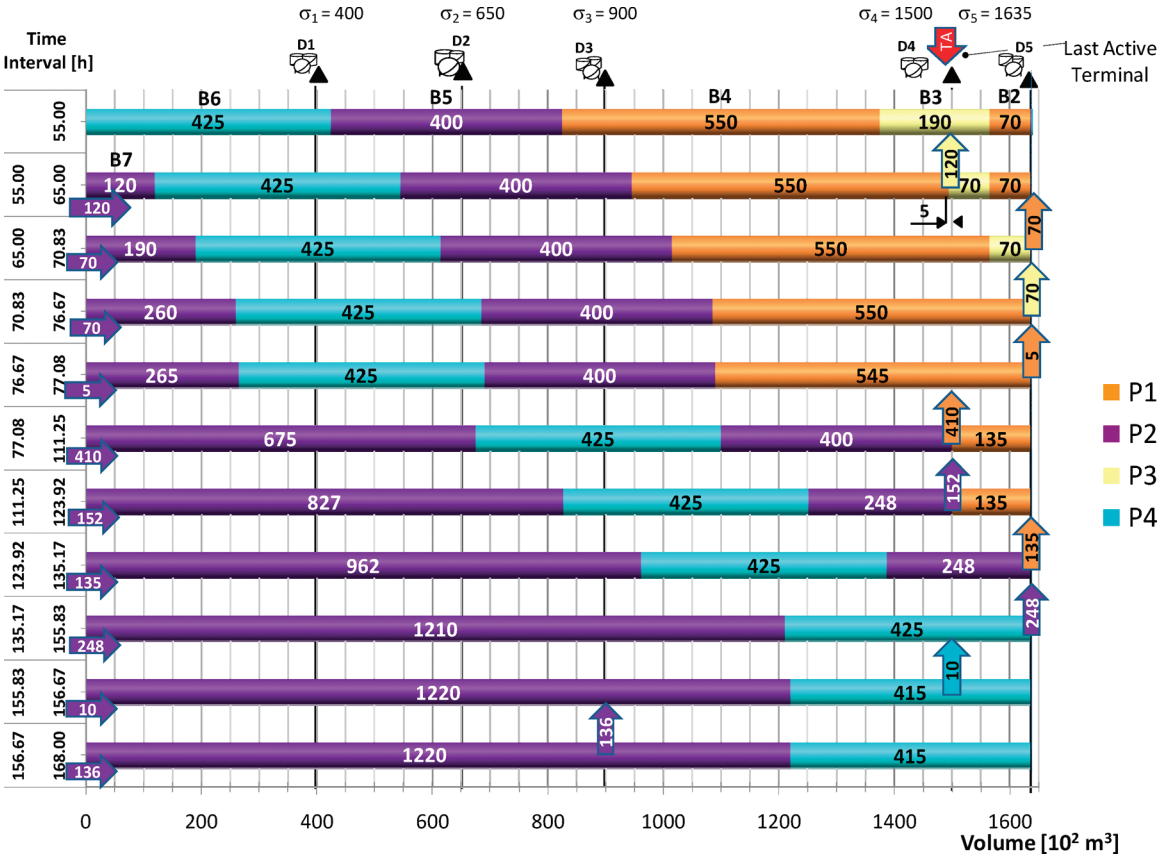

Figure 9. Detailed output operations while injecting batch B7.

Table 4. Computational Results for Examples 1−3

example	methodology		pump runs $\|K\|$	activate vol (10^2 m^3)	total cost ($)	opt gap (%)	CPU time (s)	cont vars	binary vars	eqs
1	MILP	ND	10	270	12 700	0.0	0.33	2072	100	1180
	simul.	NC	10	270	12 700	0.0	0.75			
		FF	12	1140	23 400	84.3	0.75			
		NF	13	1275	25 750	102.8	0.75			
2	MILP	ND	25	3475	59 750	0.0	6.36	6188	216	2785
		PD	25	3475	59 750	0.0	1.72*	3292§	142§	1784§
		FD	25	3475	59 750	0.0	0.64*	2072†	100†	1180†
	simul.	NC	26	3475	60 750	1.7	3.00			
		NF	32	5715	86 650	45.0	3.00			
		FF	31	6180	92 800	55.3	3.00			
3	MILP	ND	53	7950	132 500	0.0	2261.5	18729	481	6423
		PD	53	7950	132 500	0.0	108.1*	3769§	222§	2298§
		FD	53	8800	141 000	6.4	32.1*	3167†	210†	2044†
	simul.	NC	55	7950	134 500	1.5	6.00			
		NF	65	13645	201 450	52.0	6.00			
		FF	63	14325	206 250	55.7	6.00			

* Total CPU time (summing over all iterations). § For the largest MILP PD formulation (example 2, pair B6−B7; example 3, pair B12−B13). † For the largest MILP FD formulation (example 2, batch B7; example 3, batch B13).

Three instances of the proposed case study will be tackled. Example 1 faces the problem of generating the detailed sequence of product deliveries (cuts) to carry out during the injection of one of the batches of the aggregate plan (batch B7). In turn, example 2 aims at developing the detailed schedule of the pipeline system being studied over a 2 week planning horizon. Finally, example 3 deals with the complete case study.

At any problem instance, the following values for the objective function coefficients have been adopted: ca = 0.10 ($/m^3), cs = 0.00 ($/m^3), and fco = 1000 ($). The MILP mathematical models are solved to optimality on an Intel Core i7 Processor

dx.doi.org/10.1021/ie200007a |Ind. Eng. Chem. Res. 2011, 50, 6240–6259

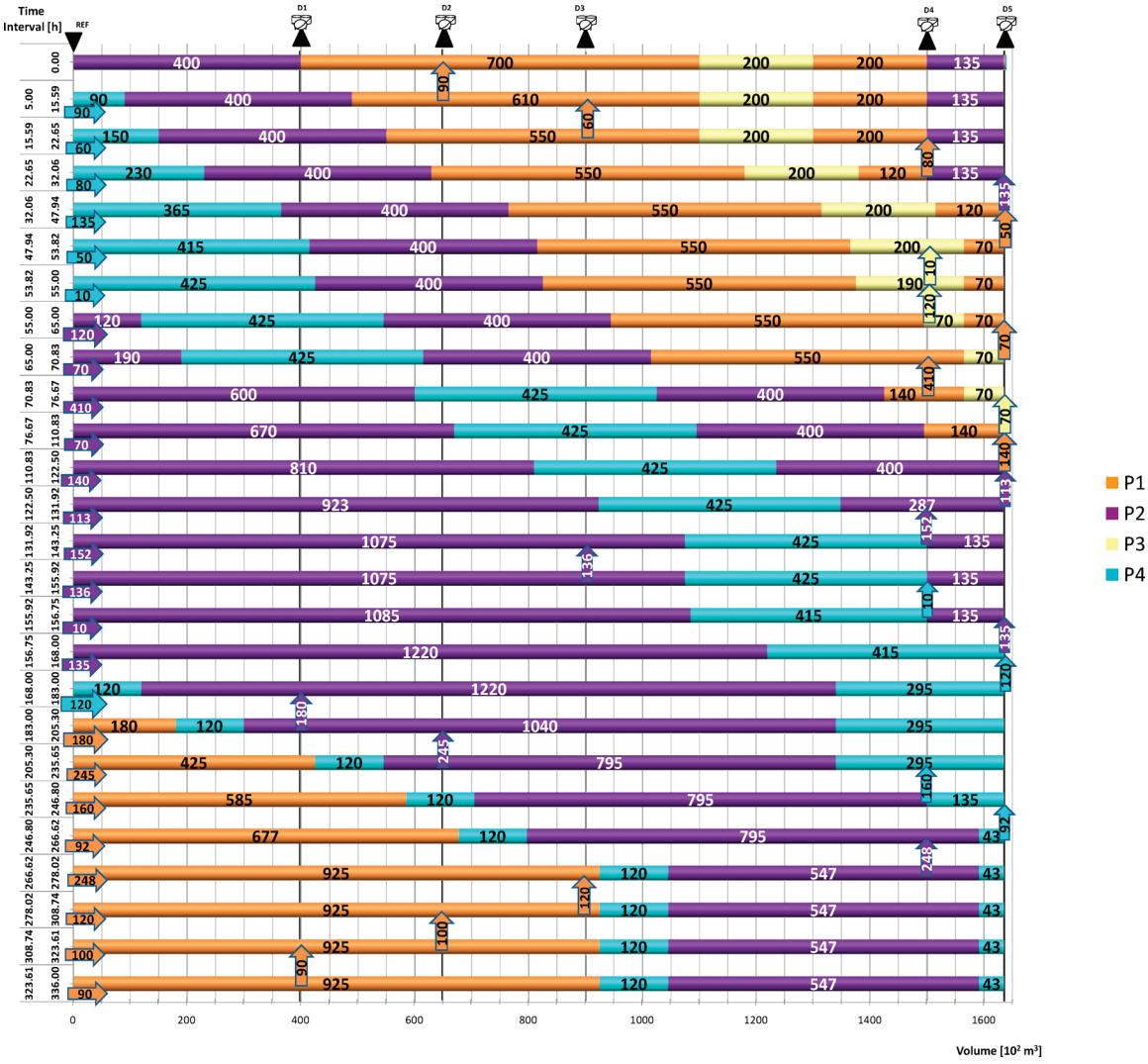

Figure 10. Optimal solution for example 2 by using the full MILP formulation.

(3.33 GHz) with GAMS/GUROBI 3.0 (with four parallel threads) as the MILP solver.[19] An optimality tolerance of 10^{-9} is adopted. A similar processor is employed to run the discrete-event pipeline simulator developed on Arena 12.0.[18] The simulation model regards the content of the pipeline system as composed by a large number of relatively small entities whose volume is fixed at 100 m^3. Then, a total of 1635 entities is required to fill the whole pipeline.

7.1. Example 1. To better appreciate the refinement process carried out by the optimization approach transforming the aggregate schedule into a cost-effective sequence of cut operations (each one involving a single receiving terminal and a unique giving batch), a small instance of the proposed case study is first solved. It is aimed at scheduling the detailed input and output operations to perform while pumping the new batch B7 at the pipeline origin. As shown in the third line of Figure 8 and in Table 3, the solution provided by the continuous scheduling approach of Cafaro and Cerdá[2] specifies the execution of nine aggregate deliveries during the injection of B7. Therefore, the cardinality of the set of runs (K) must never be lower than 9. It was adopted $|K| = 10$, because otherwise the MILP model has no feasible solution. In other words, one of the aggregate deliveries requires a pair of nonconsecutive runs to be completed. The active terminal right before starting the injection of B7 is assumed to be depot D4.

The starting linefill on the top row of Figure 9 contains the following batches and volumes (given as subscripts in 10^2 m^3): $B6_{425}/B5_{400}/B4_{550}/B3_{190}/B2_{70}$. The output operations to be accomplished while injecting B7 into the line involve three receiving terminals (D3−D5) and six supplying batches (B2−B7) (see Table 3). Hence, a lower bound on the total restart volume is given by the content of segment D4−D5 connecting the initial active terminal D4 to the farthest destination D5, that is, 135 (10^2 m^3). However, no feasible solution is able to reach such a bound on the restart volume.

The best sequence of cut operations while pumping B7 in the line was found by solving the full MILP mathematical formulation in 0.33 CPU s, and is shown in Figure 9. Flow should be restarted two times in the last segment D4−D5 featuring a volume of 135 (10^2 m^3) during runs $k = 2$ and $k = 7$ (see lines 3 and 8 of Figure 9). As a result, the total activated volume where flow should be restored amounts to the following: $2(\sigma_{D5} - \sigma_{D4}) = 270$ (10^2 m^3). As shown in Figure 9, the MILP approach initially favors the delivery of 120 units of product P3 to the current active

dx.doi.org/10.1021/ie200007a |*Ind. Eng. Chem. Res.* 2011, 50, 6240–6259

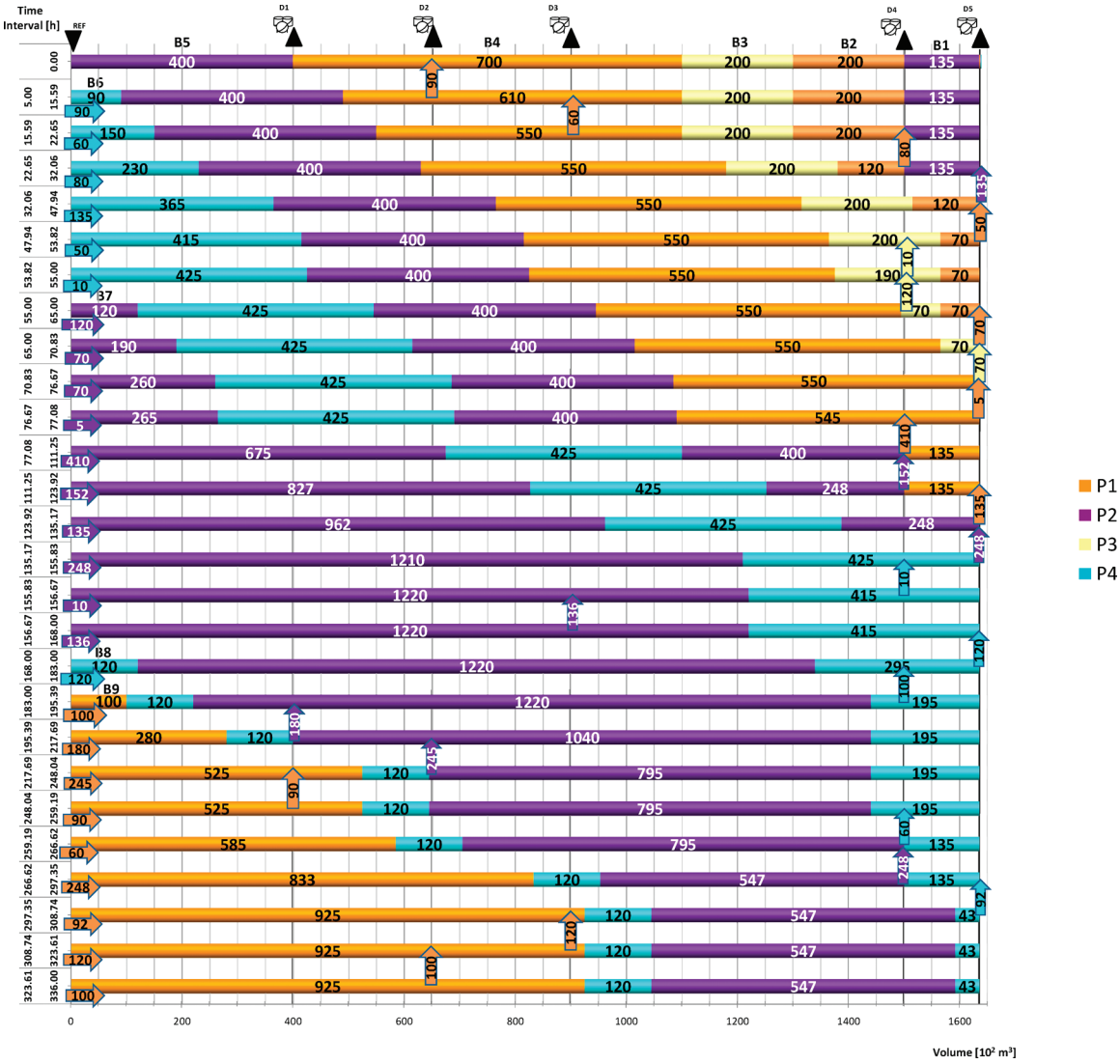

Figure 11. Detailed schedule for example 2 using the simulation model + NC rule.

terminal D4 to avoid restart costs. At the completion time of run $k = 1$, batch B4 is still 5 volumetric units away from depot D4, and only some of the specified deliveries to depot D5 can be accomplished. Then, the flow in D4−D5 must be restarted, and depot D5 is selected as the receiving terminal for the next three deliveries (70 units of B2, 70 units of B3, and 5 units of B4). In this way, no restart/stop operations are needed until run $k = 5$.

The stripping operation on batch B4 at depot D5 is limited to 5 units because otherwise the specified delivery of 410 units to terminal D4 from B4 cannot be fully performed. In fact, the maximum volume that can be transferred from B4 to D4 at the end of run $k = 4$ is exactly equal to 410 units, and consequently depot D4 becomes a restrictive terminal. Then, the flow is stopped in pipe D4−D5, and D4 is chosen as the next active terminal sequentially receiving 410 units from batch B4 ($k = 5$) and 152 units from B5 ($k = 6$). From Figure 9, it is clear that the change of the receiving terminal is not a good option for the MILP model unless it is strictly necessary. Afterward, the output operations move again to depot D5 to remove 135 units from B4 ($k = 7$) and 248 units from B5 ($k = 8$). Consequently, the flow in segment D4−D5 is restarted again. Note that the aggregate delivery of 140 units of product P1 to D5 from the giving batch B4 is made through a pair of nonconsecutive cut operations because depot D4 emerges as a restricted terminal, thus limiting the size of the first cut to just 5 units. The last two deliveries take place at the upstream locations D4 ($k = 9$) and D3 ($k = 10$). Therefore, there is no further flow restart, but the segments D4−D5 and D3−D4 must be sequentially stopped.

As a result, the accumulated stopped volume is equal to the following: $2(\sigma_{D5} - \sigma_{D4}) + (\sigma_{D4} - \sigma_{D3}) = 870$ (10^2 m^3). In example 1, the initial active terminal is D4 and the final one is D3. Hence, $\Delta = \sigma_{D3} - \sigma_{D4} = -600$ (10^2 m^3) and the equality AV = SV + Δ holds; that is, AV = 870 − 600 = 270. A detailed output schedule similar to the one depicted in Figure 9 is obtained by running the discrete-event simulation model with the nearest-to-the-current-terminal priority rule. This finding may indicate that the underlying pattern on the optimal sequence of active terminals is quite close to the one proposed by the NC rule. Computational results for example 1, including model size, CPU time, and optimal objective value, are all given in Table 4. When the farthest-first or the nearest-first rules choose the next active terminal, the simulation model produces suboptimal solutions

dx.doi.org/10.1021/ie200007a |*Ind. Eng. Chem. Res.* 2011, 50, 6240–6259

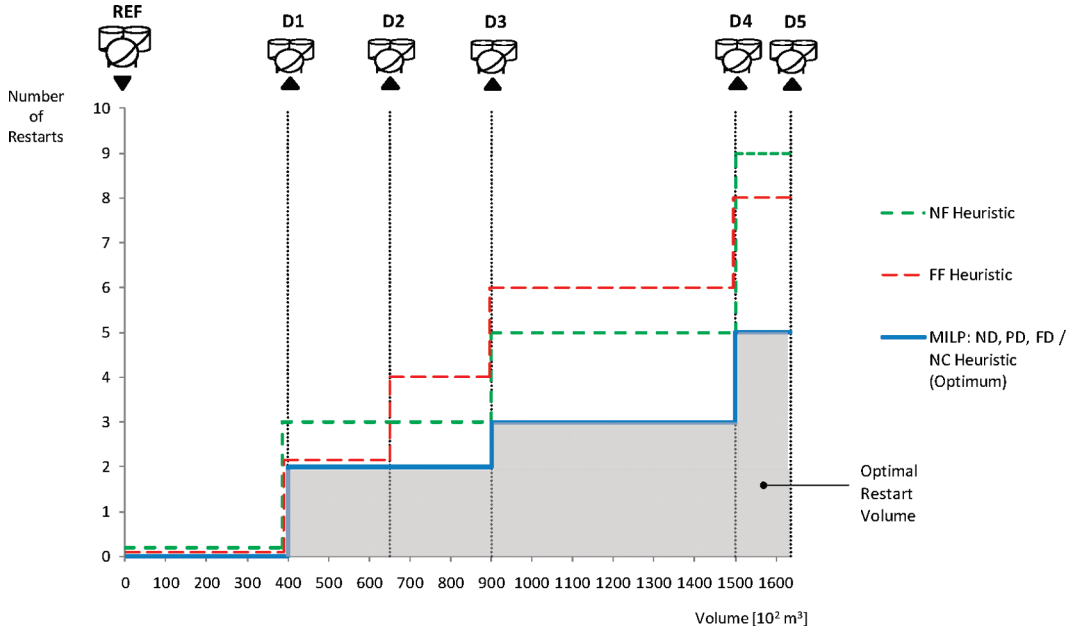

Figure 12. Distribution profiles of the restart flow along the pipeline for example 2.

featuring a total restart volume higher than the optimal one by a factor of 4, and a larger number of pumping runs. For any rule, the discrete-event simulation model generates the detailed pipeline schedule for the injection of B7 in 0.75 CPU s.

7.2. Example 2. Example 2 revisits the proposed case study involving a single-source pipeline, four commodities, and five distribution terminals, but this time the length of the planning horizon has been fixed at 2 weeks. The input schedule for the first 2 weeks comprises the pumping of the top four batches (B6−B9) shown in Table 2. Depot D1 was assumed to be the current active terminal at $t = 0$. Six aggregate deliveries are to be performed while pumping B6, nine while injecting batch B7, only one with B8, and another eight during the insertion of B9, amounting to a total of 24 aggregate deliveries. Therefore, the cardinality of the set K must not be lower than 24. In fact, $|K|$ should be increased to 25, because otherwise the full MILP formulation (algorithm ND) has no feasible solution (see Table 4). A summary of the computational results obtained with all the proposed approaches is given in Table 4. The optimal detailed pipeline schedule for a 2 week planning horizon was found in 6.36 CPU s through the MILP-ND problem formulation and is depicted in Figure 10.

All but one of the 24 aggregate deliveries is performed through a single run. The only product delivery requiring a pair of nonconsecutive input operations takes place during the injection of batch B7. It involves the withdrawal of 248 units of product P2 from batch B5 at depot D5. At line 13 of Figure 10, the amount of P2 diverted from B5 to depot D5 during run $k = 12$ has been limited to 113 units because D4 becomes a restrictive terminal. Therefore, input operations should move to terminal D4, or otherwise the model will fail to fully perform the delivery of the remaining 152 units of P2 from B5 to D4. From Figure 10 it follows that the optimal solution requires to restore the flow five times in pipe D4−D5, three times in the pipeline segment D3−D4, and two times in both sections D2−D3 and D1−D2. Therefore, $AV = 5 \times 135 + 3 \times 600 + 2 \times 250 + 2 \times 250 = 3475$ (10^2 m^3). Moreover, depot D1 is at the same time the starting and the final active terminal. Then, $\Delta = 0$ and $SV = AV = 3475$ (10^2 m^3) at the optimum. As also shown in Table 4, both decomposition algorithms FD and PD also achieve the optimal cost provided by the full MILP formulation at much lower computational cost. Interestingly, the overall CPU time decreases by a factor of 4 and 10, respectively.

On the other hand, the discrete-event simulation model again reaches the minimum restart volume when the NC rule chooses the next receiving terminal (see Figure 11). Although the NC rule yields the best schedule among those obtained by running the simulation model with different heuristic rules, it requires one additional run above the optimal value for the problem. In fact, the number of runs performed during the injection of batch B9 rises from 8 to 9. Moreover, the sequence of cuts during the pumping of B7 differs from the one adopted in Figure 10. In this case, the aggregate delivery of P1 from the giving batch B4 to depot D5 is the one completed through a pair of nonconsecutive runs. Therefore, the same sequence of cuts provided by the MILP formulation at example 1 is carried out (see Figure 9). All the simulation runs are completed in 3 s of CPU time, half of the time required by the full MILP formulation. These results reveal the potential of the discrete-event simulation tool when a proper priority rule is applied. Instead, the relative performances of the FF and NF rules have improved with regard to example 1, but the required restart volume is still 64−77% higher than the optimal value. Moreover, they need to accomplish six to seven input operations above the minimum number to complete the aggregate deliveries.

Weighted flow restarts at every pipeline segment using all the alternative methodologies are depicted in Figure 12. The number of pipe restarts increases as the distance between the segment and the pipeline origin rises. The area below the distribution curve is a measure of the total flow restart volume.

7.3. Example 3. The complete case study is considered at example 3 by extending the length of the planning horizon from 2 to 4 weeks. Compared to example 2, the new problem should take into account the injection of five additional batches (B10−B14) at the input station and 25 further aggregate product

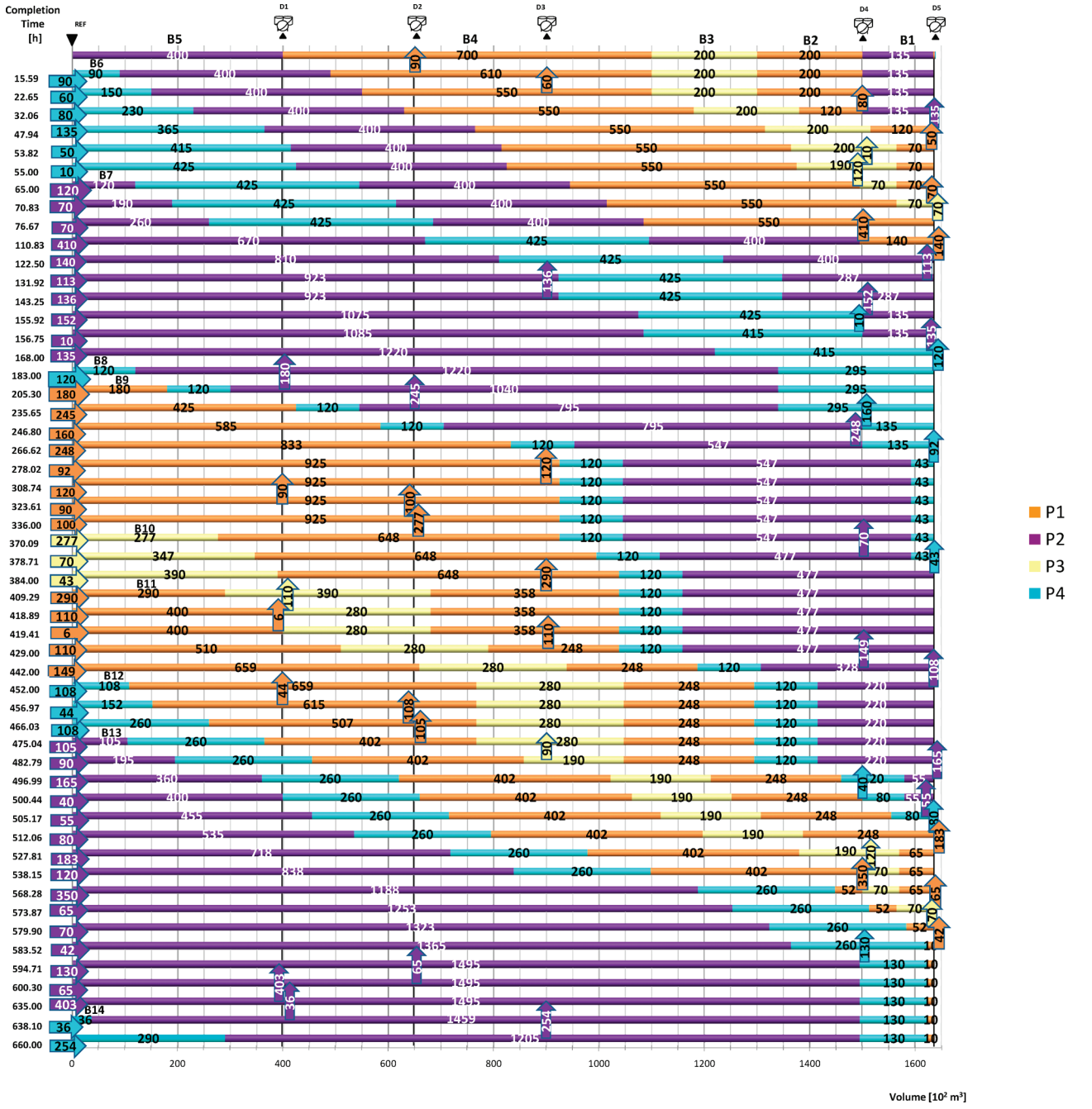

Figure 13. Optimal solution for example 3 using the full MILP formulation.

deliveries (see Table 2 and Figure 8). Note that 12 of the additional deliveries take place during the injection of lot B13. This implies a total of 49 aggregate deliveries and a similar lower bound for the cardinality of set K. Nonetheless, to get model feasibility, $|K|$ must be raised to 53. As a result, the size of the full optimization model significantly grows, and the number of binary and continuous variables rises up to 481 and 18 729, respectively (see Table 4). Consequently, the computational efficiency of the optimization approach drastically diminishes, and the solution time shows an exponential increase. Hence, example 3 looks like an adequate benchmark problem for checking out the advantages of using some kind of decomposition techniques to solve the MILP optimization model. Not only the savings on CPU time will be important for the evaluation of alternative solution techniques but also the quality of the solution found in terms of the total flow restart volume and the required number of runs. Figure 13 shows the optimal solution for example 3 obtained by solving the full MILP formulation.

As expected, the CPU time jumped from 6.36 s for example 2 to 2261.5 s for example 3, i.e., a factor of 355. Usually, there are several detailed schedules featuring the lowest restart volume and the least number of operations. From Table 4, it follows that the PD strategy also finds an optimal solution but requiring a CPU time of 108.1 s, more than 20 times lower than the solution time for the full (ND) MILP formulation. Contrarily, the simulation approach with the NC heuristic fails to minimize the value of the

dx.doi.org/10.1021/ie200007a |*Ind. Eng. Chem. Res.* 2011, 50, 6240–6259

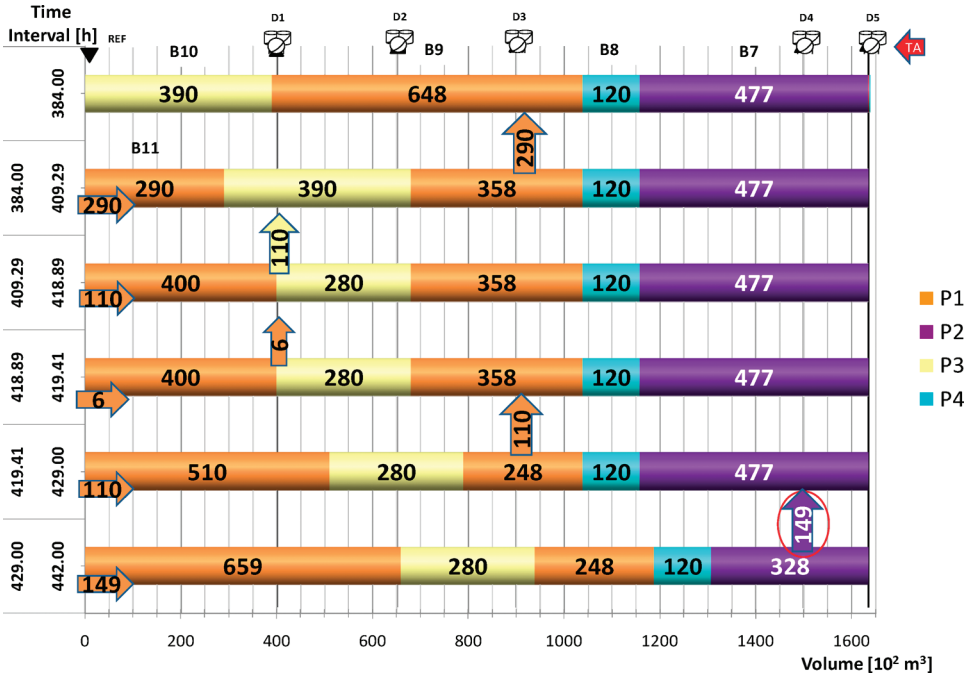

Figure 14. Optimal detailed schedule for the pumping of batch B11 at example 3.

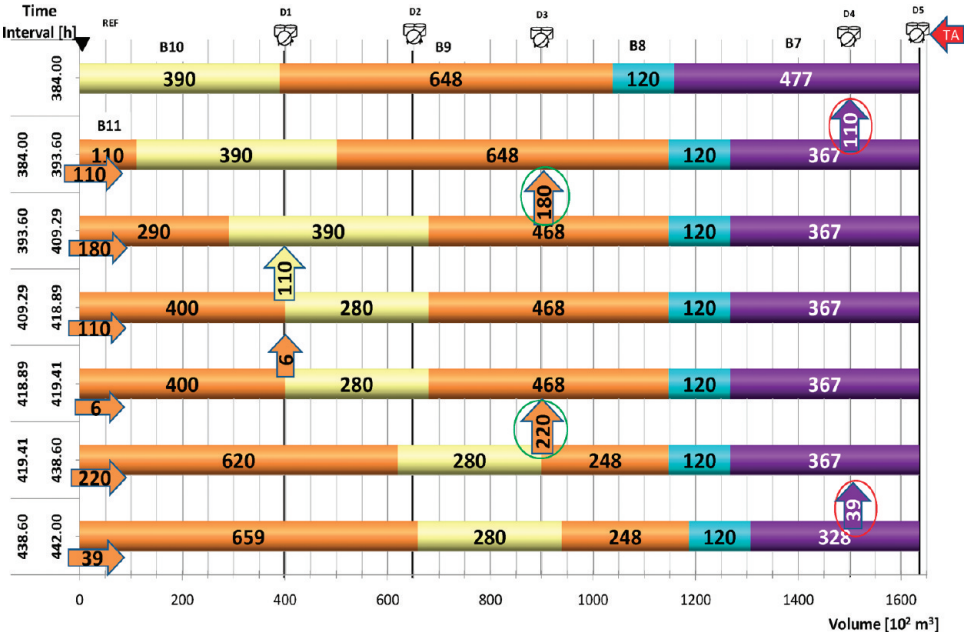

Figure 15. Detailed schedule for the pumping of B11 provided by the NC rule (example 3).

objective function for example 3. It reaches the least restart volume but using a higher number of input operations, that is, 55 runs instead of 53. These two additional runs are required while injecting batches B9 and B11. For instance, the detailed schedule provided by the full MILP formulation or the PD solution strategy for the pumping of B11 completes each of the specified aggregate deliveries through a single run, except for the P1 delivery to depot D3 (see Figure 14).

In contrast, the simulation model with the NC rule should accomplish a pair of runs to carry out (i) the delivery of 400 units of P1 from batch B9 to depot D3 and (ii) the removal of 149 units of P2 at depot D4 from lot B7 (see Figure 15). However, it only demands a CPU time of 6 s to develop the detailed schedule.

When using a full decomposition strategy to solve the MILP formulation, an acceptable feasible solution was discovered featuring the minimum number of pumping runs, but a value of AV 10.7% higher than the optimal value. On the other hand, the NF and FF rules in combination with the discrete-event simulation model lead to rather poor suboptimal solutions requiring 10−12 more runs and a flow restart volume that almost

dx.doi.org/10.1021/ie200007a |*Ind. Eng. Chem. Res.* 2011, 50, 6240–6259

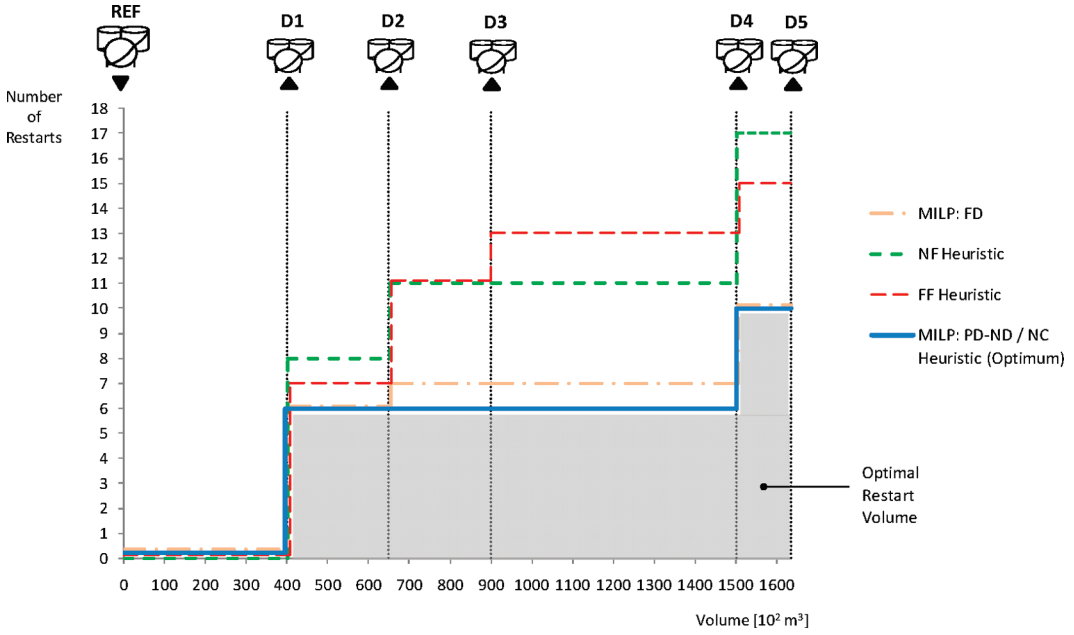

Figure 16. Distribution Profile of the restart flow along the pipeline for example 3.

Table 5. Computational Results for Examples 2 and 3 Using the Full MILP Model

example	pump runs $\|K\|$	activate vol (10^2 m^3)	total cost ($)	opt gap (%)	CPU time (s)	cont vars	binary vars	eqs
2	24	infes			0.01	5 940	206	2669
	25	3475	59 750 *	0.0	6.36	6 188	216	2785
	26	3475	59 750 *	0.0	12.92	6 436	226	2901
3	49	infes			4.10	17 302	438	5898
	50	infes			7.41	17 650	448	6014
	51	infes			28.81	18 001	453	6117
	52	8085	132 850	0.3	1 669.1	18 381	471	6307
	53	7950	132 500 *	0.0	2 261.5	18 729	481	6423
	54	7950	132 500 *	0.0	29 931.5	19 093	495	6576

* Optimal cost.

doubles the optimal one. Finally, the distribution profiles of the flow restart volume along the pipeline using the optimization and discrete-event simulation tools are depicted in Figure 16. From Table 4, it follows that the PD solution strategy appears as a very cost-effective way to find an optimal or near-optimal detailed schedule for large pipeline scheduling problems involving rather long time horizons.

7.4. Effect of the Number of Runs on the Computational Cost. A thorough analysis of the effect of the proposed number of runs $|K|$ on the computational cost for solving examples 2 and 3 through the full MILP mathematical formulation is presented in Table 5.

At example 2, the value of $|K|$ is initially fixed at 24, i.e., the total number of aggregate deliveries. However, the full MILP mathematical model is infeasible, and $|K|$ should be increased by 1. The new number of runs $|K| = 25$ is large enough to find a feasible pipeline schedule with a total cost of $59 750 in 6.36 s. At the same time, the number of variables and constraints slightly grow. To verify if the optimal solution has been found, the value of $|K|$ is raised to 26. The new instance of the full MILP formulation was again solved to optimality in 12.92 s. The solution time has doubled, but the total cost at the optimum is still equal to $59 750. Moreover, one of the proposed runs is never executed. Therefore, the best detailed pipeline schedule is assumed to be found. A similar analysis was made for example 3 requiring the execution of 49 aggregate deliveries. The instances of the full MILP formulation for $|K| = 49$, $|K| = 50$, and $|K| = 51$ were all infeasible, but the CPU time needed to prove infeasibility rises 7 times when $|K|$ goes up from 49 to 51. A further increase of $|K|$ to 52 leads to a feasible solution with a total cost of $132 850 at the expense of a computational effort 58 times larger. To verify the optimality of the solution found, the full MILP model was solved again with $|K| = 53$. In this way, a better feasible solution with a total cost of $132 500 is discovered in 2261.5 s. A further increase of $|K|$ to 54 does not yield any improvement on the total cost, and a fictitious run arises at the optimum. Therefore, the best detailed pipeline schedule for example 3 has been found. For $|K| = 54$, the solution time steps up by a factor of 13, to 29 931.5 s. From these results, it is quite evident the potential of the PD optimization algorithm finding the optimal solution to example 3 in only 108.1 s.

8. CONCLUSIONS

Two alternative approaches for generating the detailed operational schedule of single-source pipeline systems with several

dx.doi.org/10.1021/ie200007a |Ind. Eng. Chem. Res. 2011, 50, 6240–6259

distribution terminals have been developed. They both assume that the aggregate pipeline plan provided by a continuous scheduling approach such as the one proposed by Cafaro and Cerdá[2] is already available. An aggregate plan consists of a series of multidelivery pumping runs with no information on the way product deliveries are performed during every batch injection. However, the pipeline operator needs to have a detailed schedule involving a sequence of single-delivery runs, each one transferring some amount of product from a single batch to a unique receiving terminal. Reducing energy consumption and pump maintenance costs is one of the major goals for the disaggregation process of the delivery schedule. As suggested by Hane and Ratliff,[16] such a goal can be achieved by minimizing both the total pipeline volume where flow should be restarted and the number of single-delivery runs. One of the proposed methodologies is based on a rigorous MILP mathematical formulation whose size increases with the number of batch injections. To deal with large scheduling problems, decomposition-based solution strategies have been developed to solve a sequence of smaller MILP models at much lower computational cost. They are based on the empirical fact that the effect of delivery decisions concerning a batch injection is mostly confined to the next run. The so-called pair-decomposition (PD) technique proves to be a very reliable way to obtain the optimal detailed schedule, even for large problems, at a reasonable computational cost. The other proposed approach runs a discrete-event simulation model in combination with heuristic rules for selecting the next receiving terminal. Of the three heuristics analyzed, the nearest-to-the-current terminal (NC) rule always generates very good solutions. Moreover, simulation runs only take a few seconds to develop a detailed pipeline schedule. For the real-world case study tackled in this paper, the PD decomposition-based strategy for solving the MILP formulation finds the global optimal solution in approximately 100 CPU s, while the simulation model together with the NC rule provide a near-optimal schedule, featuring an objective value only 1.57% over the optimal cost, in only 6 CPU s.

AUTHOR INFORMATION

Corresponding Author

*Tel.: +54 342 4559175. Fax: +54 342 4550944. E-mail: jcerda@intec.unl.edu.ar.

ACKNOWLEDGMENT

Financial support received from FONCYT-ANPCyT under Grant PICT 01837, from CONICET under Grant PIP-2221, and from Universidad Nacional del Litoral under CAI+D 66-335 is fully appreciated.

NOMENCLATURE

(a) Sets

I = chronologically arranged batches ($I^{old} \cup I^{new}$)
I^{new} = new batches to be injected at the origin of the pipeline
I^{old} = old batches inside the pipeline at the start of the time horizon
J = distribution terminals
$J_{i,i'}$ = subset of terminals receiving a portion of batch i during the injection of batch i'
K = ordered set of detailed pumping operations
K_i = subset of operations injecting batch i
P = refined petroleum products
P_i = product transported by lot i
S_{inew} = subset of new batches considered in the MILP subproblem

(b) Parameters

ca = unit flow restart cost
cs = unit flow stoppage cost
$dd_{i,j}^{(i')}$ = aggregate delivery from batch i to terminal j during the injection of batch i'
$d_{i,j,k}^{(i')}$ = amount of product from batch i already transferred to terminal j at event k of injection i'
d_{min} = minimum delivery size for a single operation
fco = fixed cost for performing a single pumping run
$ft_{i'}$ = completion time for the injection of batch i'
h_{max} = horizon length
$l_{max}^{(i')}$ = upper bound on the length of an input operation of batch i'
l_{min} = minimum length of a detailed run
pv = total pipeline volume
$qq_{i'}$ = overall size of the new lot i'
$st_{i'}$ = starting time for the injection of batch i'
v = entity volume for the discrete-event simulation model
$vb_{min}^{(i')}/vb_{max}^{(i')}$ = minimum/maximum injection rates of batch i'
wo_i = initial volume of old batch i
σ_j = volumetric coordinate of depot j from the origin of the pipeline

(c) Variables

Continuous Variables

AV_k = pipeline volume activated at the start time of run k
C_k/L_k = completion time/length of pumping run k
$D_{i,j}^{(k)}$ = volume of batch i diverted to depot j while performing operation k
$F_i^{(k)}$ = front coordinate of batch i at time C_k
$N_{i,j,k}^{(i')}$ = number of entities of batch i located between the pipeline origin and terminal j at event k pumping lot i'
Q_k = injected volume during run k
SV_k = pipeline volume stopped at the start time of run k
$W_i^{(k)}$ = size of batch i at time C_k

Binary Variables

u_k = existence of run k
$x_{i,j}^{(k)}$ = existence of a delivery from batch i to depot j while performing run k

REFERENCES

(1) Cafaro, D. C.; Cerdá, J. Optimal Scheduling of Multiproduct Pipeline Systems Using a Non-Discrete MILP Formulation. *Comput. Chem. Eng.* **2004**, *28*, 2053–2068.

(2) Cafaro, D. C.; Cerdá, J. Dynamic Scheduling of Multiproduct Pipelines with Multiple Delivery Due Dates. *Comput. Chem. Eng.* **2008**, *32*, 728–753.

(3) Castro, P. M. Optimal Scheduling of Pipeline Systems with a Resource−Task Network Continuous-Time Formulation. *Ind. Eng. Chem. Res.* **2010**, *49*, 11491–11505.

(4) Relvas, S.; Matos, H. A.; Barbosa-Póvoa, A. P. F. D.; Fialho, J.; Pinheiro, A. S. Pipeline Scheduling and Inventory Management of a Multiproduct Distribution Oil System. *Ind. Eng. Chem. Res.* **2006**, *45*, 7841–7855.

(5) Cafaro, D. C.; Cerdá, J. Efficient Tool for the Scheduling of Multiproduct Pipelines and Terminal Operations. *Ind. Eng. Chem. Res.* **2008**, *47*, 9941–9956.

(6) Rejowski, R.; Pinto, J. M. Scheduling of a Multiproduct Pipeline System. *Comput. Chem. Eng.* **2003**, *27*, 1229–1246.

(7) Magatão, L.; Arruda, L. V. R.; Neves, F., Jr. A Mixed Integer Programming Approach for Scheduling Commodities in a Pipeline. *Comput. Chem. Eng.* **2004**, *28*, 171–185.

(8) Zyngier, D.; Kelly, J. D. Multi-Product Inventory Logistics Modeling in the Process Industries. *Optim. Logistics Challenges Enterp.* **2009**, *30*, 61–95.

(9) Herrán, A.; de la Cruz, J. M.; de Andrés, B. A Mathematical Model for Planning Transportation of Multiple Petroleum Products in a Multi-Pipeline System. *Comput. Chem. Eng.* **2010**, *34*, 401–413.

(10) Rejowski, R.; Pinto, J. M. A Novel Continuous Time Representation for the Scheduling of Pipeline Systems with Pumping Yield Rate Constraints. *Comput. Chem. Eng.* **2008**, *32*, 1042–1066.

(11) Neves, F., Jr.; Magatão, L.; Stebel, S. L.; Boschetto, S. N.; Felizari, L. C.; Czaikowski, D. I.; Rocha, R.; Ribas, P. C. An Efficient Approach to the Operational Scheduling of a Real-World Pipeline Network. *Comput.-Aided Chem. Eng.* **2007**, *24*, 697–702.

(12) Mori, F. M.; Lüders, R.; Arruda, L. V. R.; Yamamoto, L.; Bonacin, M. V.; Polli, H. L.; Aires, M. C.; Bernardo, L. F. J. Simulating the Operational Scheduling of a Realworld Pipeline Network. *Comput.-Aided Chem. Eng.* **2007**, *24*, 691–696.

(13) Moura, A. V.; de Souza, C. C.; Cire, A. A.; Lopes, T. M. Planning and Scheduling the Operation of a Very Large Oil Pipeline Network. *Lect. Notes Comput. Sci.* **2008**, *5202*, 36–51.

(14) García-Sánchez, A.; Arreche, L. M.; Ortega-Mier, M. Combining Simulation and Tabu Search for Oil-Derivatives Pipeline Scheduling. *Stud. Comput. Intell.* **2008**, *128*, 301–325.

(15) Boschetto, S. N.; Magatão, L.; Brondani, W. M.; Neves, F., Jr.; Arruda, L. V. R.; Barbosa-Póvoa, A. P. F. D.; Relvas, S. An Operational Scheduling Model to Product Distribution through a Pipeline Network. *Ind. Eng. Chem. Res.* **2010**, *49*, 5661–5682.

(16) Hane, C. A.; Ratliff, H. D. Sequencing Inputs to Multicommodity Pipelines. *Ann. Oper. Res.* **1995**, *57* (1), 73–101.

(17) Cafaro, V. G.; Cafaro, D. C.; Méndez, C. A.; Cerdá, J. Oil-Derivatives Pipeline Logistics Using Discrete-Event Simulation. *Winter Simul. Conf. Proc.* **2010**, 2101–2113.

(18) Kelton, W. D.; Sadowski, R. P.; Sturrock, D. T. *Simulation with ARENA*; McGraw-Hill: New York, 2007.

(19) Brooke, A.; Kendrick, D.; Meeraus, A.; Raman, R. *GAMS—A User's Guide*; GAMS Development: Washington, DC, 2006.