Planning and Scheduling the Operation of a Very Large Oil Pipeline Network

Arnaldo V. Moura*, Cid C. de Souza, Andre A. Cire, and Tony M.T. Lopes

Institute of Computing - University of Campinas, 13084-971 - Campinas, Brazil
{arnaldo,cid}@ic.unicamp.br, {andre.cire,tony.lopes}@gmail.com

Abstract. Brazilian PETROBRAS is one of the world largest oil companies. Recurrently, it faces a very difficult over-constrained planning challenge: how to operate a large pipeline network in order to adequately transport oil derivatives and biofuels from refineries to local markets. In spite of being more economical and environmentally safer, the use of a complex pipeline network poses serious operational difficulties. The network has a complex topology, with around 30 interconnecting pipelines, over 30 different products in circulation, and about 14 distribution depots which harbor more than 200 tanks, with a combined capacity for storing up to 65 million barrels. The problem is how to schedule individual pumping operations, given the daily production and demand of each product, at each location in the network, over a given time horizon. We describe a solution based on a two-phase problem decomposition strategy. A novel Constraint Programming (CP) model plays a key role in modeling operational constraints that are usually overlooked in literature, but that are essential in order to guarantee viable solutions. The use of CP was crucial, since it allowed the modeling of complex constraints, including nonlinearities. The full strategy was implemented and produced very adequate results when tested over large real instances. In contrast, other approaches known from the literature failed, even when applied to much less complex networks.

1 Introduction

PETROBRAS is ranked as the 14th largest oil company in the world (see www.energyintel.com). One of the major sources of costs faced by PETROBRAS is related to transportation, specially regarding petroleum derivatives, such as gasoline, and biofuel, like ethanol. In this context, pipeline networks are considered the main inland transportation mode in contrast to rail and road, since they are much more economical and environmentally safer.

However, these advantages ensue a very high operational complexity. For instance, the Brazilian pipeline network owned and operated by PETROBRAS has an extension of 7,000 kilometers, comprising 29 individual interconnecting pipelines in which more than 30 different types of products are in circulation. There are 14 distribution depots that can store up to 65 millions barrels of these products,

* Corresponding author.

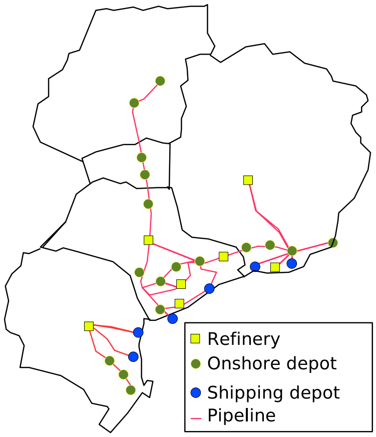

Fig. 1. PETROBRAS pipeline networks

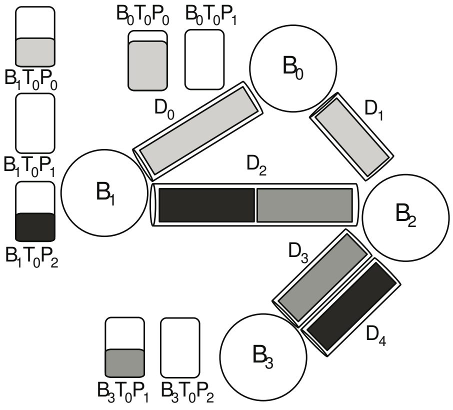

Fig. 2. A pipeline network example

stocked in more than 200 tanks located at such depots. A partial illustration of the Brazilian southeastern network is shown in Figure 1. Pipelines must always be completely filled with products, meaning that a volume must be *pushed* into a duct in order to pump out the same volume at the other extremity. Moreover, due to chemical properties, certain products can not make contact with each other - they are called *incompatibles*. Also, each product has its own flow rate interval, and that depends on the flow direction and on the particular pipeline being used. At depots, not all departing and arriving operations can be simultaneous, due to restrictions imposed both by the internal valve and ducts layout, as well as by the number of local pumps. Tanks can store just one type of product, and extraction or injection of volumes can not be simultaneous.

The problem is how to schedule all individual *pumping operations* in order to fulfill market demands and store all the planned production. Each pumping operation is defined by origin and destination tanks, a pipeline *route*, start and end times, a specific product and its respective volume. The operations must obey all constraints over the given time horizon. The management of all these resources gives rise to a complex planning and scheduling problem.

Currently, the problem is solved manually by executing a trial-and-error process with the aid of a proprietary simulator that checks whether some simple physical constraints are being satisfied. This process is very time consuming and, not rarely, the final results still violate some of the more complex restrictions. Clearly, this manual process is far from optimal and limits the efficiency of the network operation. In fact, it is common for the company to use trucks for transporting pending volumes, thus increasing the overall transportation costs, a situation that could be avoided by a more intelligent use of the pipeline network.

Due to its size and complexity, as well as to its financial impact, the efficient operation of this large oil pipeline network is one of the most strategic problems faced by logistics at PETROBRAS today. As will be discussed later, CP was at the core of a computational model devised and used to find good operational solutions for real problem instances, in an adequate amount of computer time.

Problem Description. As an illustration, Figure 2 shows a sample network with 4 depots, B_0, B_1, B_2, and B_3, interconnected by 5 pipelines. Between depots B_2 and B_3, there are 2 pipelines, which is common to occur in practice. Each depot also has its own tank farm. For instance, depot B_1 has storage tanks for products P_0, P_1 and P_2. Each tank contains an initial volume. Ducts must always be completely filled. All of these quantities are measured in standardized units.

The following constraints must be satisfied:

(1) During the whole planning horizon, a tank can store only a pre-defined product and its capacity must always be respected. But a depot not necessarily contains tanks to store all types of products. All injection and extraction operations in a tank must be disjunctive in time.
(2) Pipelines operate in an intermittent fashion and must always be completely filled. No interface losses between products are considered. Furthermore, volumes pumped out can either enter a tank or move directly into another pipe in an assigned route. The initial sequence of products inside each pipe is given.

Flows in pipelines can change direction dinamically, an event called *pipeline flow reversal*. An example of reversal is illustrated in Figure 3 for a single pipeline topology. From instants $t = 0$ to $t = 2$, a product extracted from tank $B_0T_0P_2$ in depot B_0 is being used to push another product into the tank $B_1T_0P_0$ in depot B_1. As soon as the first product is completely injected into its destination tank at $t = 3$, the second volume must return to the first depot, since there is no tank for it in depot B_1. This is done by using the product from tank $B_1T_0P_1$ to push it back to the origin tank, changing the pipeline direction at $t = 4$ and $t = 5$.
(3) Depending on the internal arrangement of a depot, certain operations can not be active simultaneously. Such sets of operations are called *forbidden alignment configurations*. Also, each depot has an upper limit on the number of outgoing pump operations, which depends on the number of available pumps.
(4) A *route* is an alternating sequence of depots and non-repeating connecting ducts. For example, the sequence $(B_0, D_1, B_2, D_3, B_3)$ represents a valid route in figure 2. Each product in circulation must have a route assigned to it, and a volume can only leave its route at the final destination tank. Although there is no restriction barring the creation of new routes, the most common choices obtained from human experience should be preferred.
(5) Least maximum flow rates among all products in any route must be enforced.
(6) To separate two incompatible products, it is possible to use a third product, called a *plug*, compatible with both products it separates.
(7) Production and demand volumes are defined per depot and per product, each with its own duration interval.

A solution is defined by a set of *pumping operations*. Each such operation is taken as a continuous and atomic pumping stream. An operation is defined by specifying information about the product, volume, route, origin and destination tanks, as well as start and end pumping times. Once a pumping operation starts, the volume must follow its designated route until it reaches the destination tank. However, a pumping operation can be stopped at any time, as long as no pumped volume (*i.e.* a volume that composes a whole operation) is interleaved with other products at any intermediate depot along its assigned route. The main goal is to find a solution that respects all operational and physical constrains of the network, as well as that uses stocks and productions to satisfy all local demands, while storing away any remaining production.

2 Why CP and Related Work

Previous studies from the literature frequently have focused on more restricted or much smaller network topologies. Usually, they consist of a single pipe connecting one origin (a refinery) to multiple depots. Different problem decompositions together with several MILP formulations [1,2,3,4] were proposed for these cases. Some studies also deal with variable pumping flow rates and other non-linear constraints [5,6]. Other approaches handle multiple origins and destinations within a more realistic network, albeit neglecting most of the hard constraints in order to make the problem tractable [7]. In [8], a MILP based on a network flow model was created to solve a relaxed version of the problem, but it took more than 50 hours of computer time only to find the LP initial basis.

As our research indicated, taking advantage of the problem structure using single MILP models is not practical for two reasons. First, most of the problem restrictions are computationally costly, or even impossible, to model as linear constraints, specially those related with variable flow rates and transmission between pipelines. Besides, MILP models would have to deal with multiple pipelines and depots, and investigation showed that the number of integer variables and constraints would increase at an unacceptable rate. On the other hand, heuristic and meta-heuristic strategies *per se* are greatly impaired when too many operational constraints are considered. This is particularly disturbing when slight modifications in a solution give rise to serious collateral perturbations over the problem structure as a whole. For example, since products can flow directly from one pipeline to another, changing a single pumping start time may delay the arrival of a number of other products that pass through connected pipelines. This can easily render a candidate solution into an infeasible one.

In face of all these issues, the use of CP was seen to offer great advantages for modeling and solving this problem. Firstly, its powerful modeling language allowed for the implementation of operationally crucial constraints, besides providing enough flexibility to extend the model if new restrictions were risen by pipeline operators. Secondly, and most importantly, it was possible to exploit specific problems patterns explicitly. This is done, for instance, by modeling multiple subproblem representations in order to use specialized and adequate constraint propagation mechanisms to solve each of the subproblems. In fact, such multiple perspectives played an important role in the final model, greatly improving domain reductions. Furthermore, a preliminary study [9] already indicated that CP would be flexible and powerful enough to treat the real problem faced by PETROBRAS. Finally, the use of CP was further fostered by its well-known good performance when treating scheduling problems [10]. In addition, CP is more suitable for our case since any feasible solution is enough.

3 How CP ?

The complete problem was solved using a hybrid approach that combined a randomized constructive heuristic and a novel CP model. The hybridization main cycle is schematically presented in Figure 4. The *planning phase*, implemented as a constructive heuristic, is responsible for creating a set of *delivery orders*. Each such order is defined by a volume, origin and destination tanks, product type, route and a delivery deadline. The planning phase must guarantee that, if all delivery orders are completed within their respective deadlines, local market demands will be fulfilled and the excess production will be correctly stored away. The *scheduling phase* takes the set of delivery orders generated by the planning phase. It must both sequence the pumping operations at the initial pipeline in each route present in a delivery order, as well as determine the start times of each of the pumping operations, while ensuring that no network operational constraint is violated at any time. The scheduling phase represents the problem's central decision process and it was implemented as a CP model. In the sequel, each phase will be discussed, with the CP model described in more detail.

Planning and Routing. To generate delivery orders, we created a randomized constructive heuristic that makes use of the accumulated experience at PETROBRAS. The purpose of the randomization is to generate diversified sets of orders in case the main cycle restarts, increasing the chance of finding solutions. Also, it takes into consideration other criteria that are difficult to handle manually, such as estimating the time for product volumes to arrive at depots.

Delivery orders are created incrementally as follows: **(1)** randomly select a local product demand in any depot, giving higher priority to demands that must be fulfilled earlier in time; **(2)** randomly choose depots that could supply volumes of the required products, as well as the routes that these volumes should traverse. In order to do so, consider factors such as pipeline occupation rate,

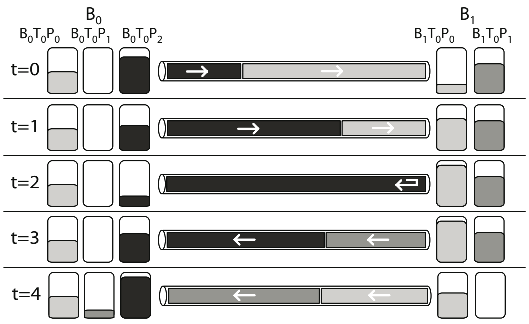

Fig. 3. Example of a flow reversal

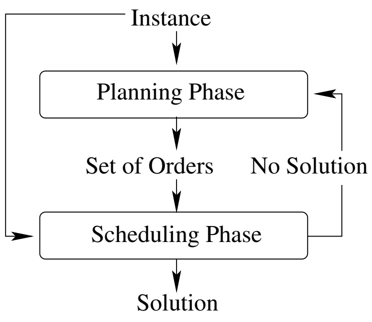

Fig. 4. Solver Framework

production schedules, present product stocks and estimated time of arrivals; **(3)** select origin and destination tanks, setting order volumes accordingly. Also, set order deadlines so as to guarantee demand fulfillment.

As soon as there is no more demands to choose from, the planning phase ends. At this point network operators can interfere adding, modifying, or removing orders according to their particular needs. This flexibility is interesting since sudden needs might unexpectedly arise. For example, there might be exceptional cases where the operators want to empty certain tanks for emergency maintenance purposes. This could be achieved by issuing new orders that remove products from those particular tanks.

All demands are guaranteed to be satisfied if the resulting orders can be scheduled to arrive by their respective deadlines. Of course, at this stage, it is not possible to know if the whole pipeline network can be operated in a way that meets all delivery order deadlines, while satisfying all problem constraints.

Orders are *indivisible*, i.e., once a volume starts to be pumped in, no other pump operation, at that same origin, can be started before the first one completes. However, orders are *preemptive* in the sense that they can be interrupted and be resumed at a later time. For instance, it is possible that a segment of the route that is being used in this pumping must also carry other products, with more pressing deadlines, along another route that has that segment in common. In such cases, it may be necessary to interrupt the present pumping operation, allowing for the more urgent products to circulate in the common pipeline segment, later resuming the first pumping.

Sequencing and Scheduling Orders. The scheduling phase must determine the pumping parameters in order to meet all delivery order deadlines, also taking into account the network operational constraints. Or it must prove that the present set of delivery orders can not be sequenced nor have their start times set in a way that observe all their assigned deadlines. At this point, orders already have their routes, volumes and origin/destination tanks assigned by the planning and routing phase, besides their deadlines.

In a typical scenario comprising 14400 minutes (*i.e.* 10 days), the model is expected to deal with around 900 delivery orders, involving dozens of products, leaving and reaching several tanks, circulating through many interconnected pipelines, and subject to thousands of constraints. In order to cope with this problem complexity, the CP model was further divided into two steps. A first model treats the sequencing of delivery orders, generating *time intervals* for the start of the respective pumping operations. After one such sequencing is completed, the most difficult constraints are guaranteed to be satisfied. Then a second, simpler, CP model takes over and determines the number of pumping operations for each delivery order (*i.e.* taking into account possible preemptions), as well as the start time of each operation.

All time variables represent minutes, the unit currently adopted by network operators. Therefore, all variables have integer domains. Time value roundings, *e.g.* due to some particular combination of flow rate and pipeline extension, can be safely neglected given the large volumes that are involved. Variable domains are easy to infer from the input data instances and are not further detailed here.

The Sequencing Model. This model must take into account product pair incompatibilities, tank capacities, pipeline flow direction restrictions and other essential operational constraints, such as no two products being pumped into a pipeline simultaneously. Furthermore, it must consider order deadlines and flow rates, in order to determine valid time bounds for the pumping operations.

The model interrelates two different *viewpoints* [11]. Firstly, the *order viewpoint* provides a *global* view of the problem, dealing mainly with routes and volume transmission between pipelines. In contrast, the *operations viewpoint* captures a *local* view of the problem, representing the pumping operation constraints in each pipeline. Both viewpoints are connected by *channeling* constraints.

The Order Viewpoint. The *order viewpoint* handles the problem globally, focusing on the relationship between orders and the pipelines that occur in their assigned routes. It also enforces constraints related to flow rates, deliver deadlines, disjunctions of pipeline operations, product incompatibilities, and tanks.

Let $\mathbf{P}$ be the pipeline set, $\mathbf{T}$ the tank set and $\mathbf{O} = \{o_1, \ldots, o_n\}$ the set of delivery orders received from the planning phase. For each $o_i \in \mathbf{O}$, let $\texttt{route}(o_i) = (p_l, \ldots, p_m)$ be the sequence of pipelines that order o_i must traverse. For each $p \in \texttt{route}(o_i)$, the volume specified by o_i can have one of four possible pipeline flow attributes when traversing pipeline p: N, if it follows the normative, or preferred, pipeline flow direction; R, if it follows the reverse direction; NR, if it starts in the normative direction, but later changes to the reverse direction, thus leaving the pipeline through the same extremity it was pumped into; and RN, similar to NR but starting in the reverse direction. Let variable $\texttt{direct}_{i,p}$ specify one among such possibilities. Finally, let $\texttt{origin}(o_i)$, $\texttt{destin}(o_i) \in \mathbf{T}$ be the origin and destination tanks, respectively, for order o_i.

For each $o_i \in \mathbf{O}$ and $p \in \texttt{route}(o_i)$, we define two *activities* [12], $snd_{i,p}$ and $rcv_{i,p}$, each composed by start and end time variables and an inferred non-negative duration variable. The first activity represents the time interval during which order o_i is being pumped into p, while the second represents the time interval during which the order is being pumped out of p. Using these activities, we can give bounds on flow rates and state delivery deadline constraints for each $o_i \in \mathbf{O}$ and each $p \in \texttt{route}(o_i)$:

$$EndTime(rcv_{i,p}) \leq \texttt{deadline}(o_i), \tag{1}$$

$$Duration(snd_{i,p}).\texttt{max\_flow\_rate}_{p,\texttt{direct}_{i,p}} \geq \texttt{volume}(o_i), \tag{2}$$

$$Duration(rcv_{i,p}).\texttt{max\_flow\_rate}_{p,\texttt{direct}_{i,p}} \geq \texttt{volume}(o_i). \tag{3}$$

Before an order exits a pipeline, it must first traverse all the pipeline extension. Thus, for each $o_i \in \mathbf{O}$ and $p \in \texttt{route}(o_i)$, we require:

$$StartTime(rcv_{i,p}) \geq StartTime(snd_{i,p}) + \left\lfloor \frac{\texttt{volume}(p)}{\texttt{max\_flow\_rate}_{p,\texttt{direct}_{i,p}}} \right\rfloor . \tag{4}$$

When an order is being pumped out of a pipeline, it is immediately pumped into the next pipeline in its route, without volume loss. This can be done by *unifying* [13] send and receive activity variables in the following way. For each $o_i \in \mathbf{O}$ and for each pair (p_l, p_m) of consecutive pipeline pairs in $\texttt{route}(o_i)$, let:

$$StartTime(rcv_{i,p_l}) = StartTime(snd_{i,p_m}), \tag{5}$$
$$EndTime(rcv_{i,p_l}) = EndTime(snd_{i,p_m}). \tag{6}$$

Order activities in a pipeline must all be *disjunctive* with respect to time; a send (or receive) activity from a certain order must not overlap with the send (or receive) activity of other orders in the pipe. In order to guarantee this, for each $p \in \mathbf{P}$, we define two *unary resources*[1]: $SndResource_p$ and $RcvResource_p$, and we associate the send and receive activities to these resources, respectively. Since an unary resource defines a mutually exclusive relationship between activities that are linked to it, each resource constraint is ranked during the solving process, *i.e.*, it is ordered along the time line. This ranking is explicitly represented in our model using positional variables $sndPos_{i,p}$ and $rcvPos_{i,p}$, accounting for, respectively, the send and receive activities positions of order o_i in pipeline $p \in \texttt{route}(o_i)$. The positional variables are connected directly to the resource's *precedence graph* [10,12], expressed by the constraints:

$$snd_{i,p} \ \textbf{startsBefore} \ snd_{j,p} \iff sndPos_{i,p} < sndPos_{j,p}, \tag{7}$$
$$rcv_{i,p} \ \textbf{startsBefore} \ rcv_{j,p} \iff rcvPos_{i,p} < rcvPos_{j,p}, \tag{8}$$
$$\forall o_i, o_j \in \mathbf{O}, \ \forall p \in \texttt{route}(o_i) \cap \texttt{route}(o_j).$$

We also add redundant *all different* global constraints [13]. For each $p \in \mathbf{P}$,

$$\textbf{all\_diff}(sndPos_{i,p}) \wedge \textbf{all\_diff}(rcvPos_{i,p}), \forall o_i \in \mathbf{O} \ \texttt{s.t.} \ p \in \texttt{route}(o_i). \tag{9}$$

In case two orders $o_i, o_j \in \mathbf{O}$ share at least one common consecutive pipeline pair $(p_l, p_m) \in \texttt{route}(o_i) \cap \texttt{route}(o_j)$, the activities precedence relations must be preserved in both pipelines. Here, we present the restrictions for flow directions N and R, the other cases being similar.

$$\begin{aligned} & sndPos_{i,p_l} > sndPos_{j,p_l} \iff sndPos_{i,p_m} > sndPos_{j,p_m} \\ \wedge \ & rcvPos_{i,p_l} > rcvPos_{j,p_l} \iff rcvPos_{i,p_m} > rcvPos_{j,p_m}, \\ & \forall o_i, o_j \in \mathbf{O}, \ \forall (p_l, p_m) \in \texttt{route}(o_i) \cap \texttt{route}(o_j). \end{aligned} \tag{10}$$

Positional variables also help discarding sequences that violate product incompatibilities. Given two orders $o_i, o_j \in \mathbf{O}$, if $\texttt{product}(o_i)$ and $\texttt{product}(o_j)$ are incompatible, then they can not make contact in a pipeline. A necessary condition for contact is that both orders enter consecutively at the same pipeline extremity, and this can only happen if they have the same entering (or leaving) pipeline flow direction. This scenario is represented by the following constraints, for each $o_i, o_j \in \mathbf{O}$, $p \in \texttt{route}(o_i) \cap \texttt{route}(o_j)$ and $\texttt{product}(o_i)$ **incompatible** $\texttt{product}(o_j)$.

$$\begin{aligned} |sndPos_{i,p} - sndPos_{j,p}| > 1 \ \texttt{if} \ & (\texttt{direct}_{i,p} \neq N \vee \texttt{direct}_{j,p} \neq RN) \\ \wedge \ & (\texttt{direct}_{i,p} \neq R \vee \texttt{direct}_{j,p} \neq NR), \end{aligned} \tag{11}$$
$$\begin{aligned} |rcvPos_{i,p} - rcvPos_{j,p}| > 1 \ \texttt{if} \ & (\texttt{direct}_{i,p} \neq N \vee \texttt{direct}_{j,p} \neq NR) \\ \wedge \ & (\texttt{direct}_{i,p} \neq R \vee \texttt{direct}_{j,p} \neq RN). \end{aligned} \tag{12}$$

[1] An *unary resource* is a resource that allows for only one activity at a time [12].

Next, we define two new activities: $ext_{i,\texttt{origin(o}_\texttt{i}\texttt{)}}$ and $inj_{i,\texttt{destin(o}_\texttt{i}\texttt{)}}$, representing volume extraction and injection, respectively, from the assigned tanks associated with order o_i. The relationship between send (receive) variables and tanks activities is the same as those for pipeline volume transmissions. For each $o_i \in \mathbf{O}$, letting p_0 and p_m be the first and last pipeline in $\texttt{route}(o_i)$, we state:

$$StartTime(snd_{i,p_0}) = StartTime(ext_{i,\texttt{origin(o}_\texttt{i}\texttt{)}}), \tag{13}$$
$$EndTime(snd_{i,p_0}) = EndTime(ext_{i,\texttt{origin(o}_\texttt{i}\texttt{)}}), \tag{14}$$
$$StartTime(rcv_{i,p_m}) = StartTime(inj_{i,\texttt{destin(o}_\texttt{i}\texttt{)}}), \tag{15}$$
$$EndTime(rcv_{i,p_m}) = EndTime(inj_{i,\texttt{destin(o}_\texttt{i}\texttt{)}}). \tag{16}$$

Injecting and extracting volumes from tanks must not overlap in time as well. Hence, activities $ext_{i,t}$ and $inj_{i,t}$ are associated with a new unary resource $TkDisj_t$, created for each tank $t \in \mathbf{T}$. However, capacities must also be taken into account in this case, requiring the combined use of a different type of resource $TkRes_t$, $t \in \mathbf{T}$, called a *reservoir* [12]. Such activities can both increase capacity (volume injection) or deplete capacity (volume extraction) from reservoirs.

Finally, we must also consider production and demand volumes in order to appropriately represent the behavior of tank capacities. Let **Dem** and **Pr** be, respectively, the sets of demands and productions given as input. For each $d \in \mathbf{Dem}$, an activity dem_d is created, with its associated constraints, considering demand time bounds and volume extraction from tanks. Similarly, an activity $prod_p$ is created for each $p \in \mathbf{Pr}$, but now considering volume injection instead of extraction. These activities are associated with the unary resources $TkDisj_t$ and reservoirs $TkRes_t$. Additional constraints are stated as follows.

$$StartTime(dem_d) \geq \texttt{DemandMinStartTime}(d), \tag{17}$$
$$EndTime(dem_d) \leq \texttt{DemandMaxEndTime}(d), \quad \forall d \in \mathbf{Dem}, \tag{18}$$
$$StartTime(prod_p) \geq \texttt{ProductMinStartTime}(p), \tag{19}$$
$$EndTime(prod_p) \leq \texttt{ProductMaxEndTime}(p), \quad \forall p \in \mathbf{Pr}. \tag{20}$$

The Operations Viewpoint. The main intuition for this viewpoint is to consider each pipeline individually (a *local* vision), since time variables domains will be automatically propagated by force of constraints defined in the order viewpoint. Although time bounds and disjunctions were already established, it is still necessary to model the fact that, in order for a certain volume to leave a pipeline, the exact amount of volume must be pumped in from the other extremity. Besides that, restrictions such as variable flow rates which depend on the products inside a pipeline, must also be considered. We will also use some ideas from previous studies [2,5,6], that treated the case of a single pipeline.

Given a pipeline $p \in P$, two time-ordered sets of *operation activities* are defined, $SndPipe_p$ and $RcvPipe_p$, where $|SndPipe_p| = |RcvPipe_p| = |\{o_i : o_i \in \mathbf{O}, p \in \texttt{route}(o_i)\}|$. As in the order viewpoint, they represent send and receive activities in p, respectively, but now with new precedence relations of the form

$$i < j \iff sndOp_{p,i} \;\; \textbf{startsBefore} \;\; sndOp_{p,j}, \; \forall sndOp_{p,i}, sndOp_{p,j} \in SndPipe_p,$$
$$i < j \iff rcvOp_{p,i} \;\; \textbf{startsBefore} \;\; rcvOp_{p,j}, \; \forall rcvOp_{p,i}, rcvOp_{p,j} \in RcvPipe_p.$$

A *volume* and a *product* variables are additionally associated with each activity belonging to $SndPipe_p$ and $RcvPipe_p$. We thus say that both sequences represent a valid ranking of *undetermined* delivery orders; they will only be *determined* when orders are ranked in their unary resources. However, since they are already time-ordered, we are able to create a more intuitive and compact model to represent pipeline flow behavior, in which constraints will also enforce propagation in the order viewpoint variable domains.

Let $rcvOp_{p,j} \in RcvPipe_p$ be an activity. A certain volume associated with it can only be received when an activity $sndOp_{p,i} \in SndPipe_p$ is being pumped at the other extremity of the pipe. In order to define which send activity i pushes a receive activity j, it is necessary to consider three factors: the pipeline volume, the volumes of the activities and the volumes between activities $sndOp_{p,i}$ and $rcvOp_{p,j}$, *i.e.*, the volume still in the pipeline before sending $sndOp_{p,i}$ and after receiving $rcvOp_{p,j}$. For the latter, a new variable $acc_{p,i,j}$ is created for each $sndOp_{p,i} \in SndPipe_p$, $rcvOp_{p,j} \in RcvPipe_p$, for $i \leq j$, as follows:

$$acc_{p,i,j} = \sum_{k<i} volume(sndOp_{p,k}) - \sum_{k \leq j} volume(rcvOp_{p,k}). \tag{21}$$

It can be shown that $sndOp_{p,i}$ never pushes $rcvOp_{p,j}$ off the pipeline, for $i > j$, due to the time-ordering of the relation. If $acc_{p,i,j} \geq volume(p)$, then it is not possible for $sndOp_{p,i}$ to push $rcvOp_{p,j}$, since a quantity greater or equal than the pipeline volume was already injected between activities i and j. On the other hand, if $acc_{p,i,j} + volume(sndOp_{p,i}) + volume(rcvOp_{p,j}) \leq volume(p)$, then the volume in $sndOp_{p,i}$ is not enough to push $rcvOp_{p,j}$ out of the pipeline. Thus, a necessary and sufficient condition for activity i to push activity j out of the pipeline (a $\mathbf{push}_{i,j}$ event), can be stated as:

$$\begin{aligned} \mathbf{push}_{i,j} \iff & acc_{p,i,j} < volume(p) \\ \wedge \quad & acc_{p,i,j} + volume(sndOp_{p,i}) + volume(rcvOp_{p,j}) > volume(p). \end{aligned} \tag{22}$$

Similar ideas can be used to determine the exact amount of volume involved in the pumping. Let $flow_{i,j}$ be the volume used in $sndOp_{p,i}$ to push the same volume $flow_{i,j}$ from $rcvOp_{p,j}$ out of the pipeline. We have:

$$\begin{aligned} \mathbf{push}_{i,j} \implies & flow_{i,j} = \mathtt{min}[volume(rcvOp_{p,j}), volume(rcvOp_{p,j}) \\ & + volume(sndOp_{p,i}) + acc_{p,i,j} - volume(p)] \\ & - \mathtt{max}[0, volume(rcvOp_{p,j}) + acc_{p,i,j} - volume(p)], \end{aligned} \tag{23}$$

$$\neg\mathbf{push}_{i,j} \implies flow_{i,j} = 0. \tag{24}$$

These flow variables can be seen as flow edges in a capacitated network. Assuming that each operation activity represents a node, the amount of volume (flow) pushed into a pipe must be equal to the volume pushed out it, and both are equal to the operation's total volume. To model this restriction, we can use a **flow** global constraint [10] for send and receive variable sequences.

In order to ensure flow rate bounds consistency, it is necessary to limit the flow rate of send and receive activities according to the products that are inside the pipeline when the pumping activity occurs. This can now be easily done using the earlier condition $acc_{p,i,j} + volume(sndOp_{p,i}) + volume(rcvOp_{p,j}) \leq volume(p)$, which is true if $sndOp_{p,i}$ and $rcvOp_{p,j}$ are both inside the pipeline at the moment of pumping. Let $MaxFl_i$ be a variable representing the maximum flow rate for activity $sndOp_{p,i}$, related to the variable $product(sndOp_{p,i})$, and let $MaxFl_j$ stand similarly for activity $rcvOp_{p,j}$. We state:

$$\begin{aligned} & acc_{p,i,j} + volume(sndOp_{p,i}) + volume(rcvOp_{p,j}) \leq volume(p) \\ \Longrightarrow & \left\lfloor \frac{volume(sndOp_{p,i})}{(EndTime(sndOp_{p,i}) - StartTime(sndOp_{p,i}))} \right\rfloor \leq MaxFl_j \\ \wedge & \left\lfloor \frac{volume(rcvOp_{p,j})}{(EndTime(rcvOp_{p,j}) - StartTime(rcvOp_{p,j}))} \right\rfloor \leq MaxFl_i . \end{aligned} \tag{25}$$

Finally, flow directions in the pipeline must be consistent as well. For instance, if an activity has its direction attribute set to N, the next activity along the pipe must necessarily have its direction attributes set to N or NR. Direction attributes such as R and RN are only consistent after a sequence of NR activities whose volume sum is equal to the pipeline volume. Attribute RN is treated similarly. These valid pairs are enforced using a *Table Constraint* [12].

For the pipeline reversal, a special constraint **reversal** was created, encapsulating the rules for the reversal of flow direction. This global constraints also controls the relation between the sequences $sndPos_{i,p}$ and $rcvPos_{i,p}$, since orders do not enter and live a pipe in the same order when there is a flow reversal, as showed in figure 3. If there is no flow reversal in a pipe p, then a constraint $sndPos_{i,p} = rcvPos_{i,p}$ is added to the model.

The Channeling Constraints. The order and operation viewpoints can be easily connected using the *element* constraint [13] and positional variables. Notice that a similar set of constraints is applied to the receive sequence.

$$\begin{aligned} StartTime(snd_{i,p}) &= StartTime(sndOp_{\,sndPos_{i,p},\,p}), && (26) \\ EndTime(snd_{i,p}) &= EndTime(sndOp_{\,sndPos_{i,p},\,p}), && (27) \\ \texttt{volume}(o_i) &= Volume(sndOp_{\,sndPos_{i,p},\,p}), && (28) \\ \texttt{direct}(o_i) &= Direct(sndOp_{\,sndPos_{i,p},\,p}), && (29) \\ \texttt{product}(o_i) &= Product(sndOp_{\,sndPos_{i,p},\,p}), && (30) \\ & \forall o_i \in \mathbf{O}, p \in \texttt{route}(o_i). \end{aligned}$$

The Scheduling Model. The second part of the complete CP model is a simpler model which is responsible for assigning the exact times to pumping operations, respecting forbidden alignment configurations and avoiding simultaneous pipe usage. The pumping operations are created by checking the $flow_{p,i,j}$ variables values for each activity i and pipeline p. If $flow_{p,i,j} > 0$, for a certain j, then there is a pumping operation of volume $flow_{p,i,j}$ with flow rate and time bounds already established by the sequencing step. In that case, a new activity, *pumpOp*, is created and its time constraints are included in the model. Note that the precedence among activities can be inferred from the orders' sequence.

Let **Dep** be the set of depots, and let $PumpOps_d$ give the pumping operations that will start at depot d, for each $d \in \mathbf{Dep}$. The simultaneous sending constraint can be implemented using a *discrete resource*, $DiscSending_d$, a resource which limits the number of consecutive operations by a certain capacity [12]. Thus, we associate each operation in $PumpOps_d$ in its respective resource $DiscSending_d$, limited by the input $\texttt{DepotMaxSimultaneousOperations}_d$. Similarly, the forbidden alignment configurations are enforced with discrete resources $AlignDisc_{a,d}$, created for each alignment restriction a. The operations associated with each resource are easily identifiable by checking their product type and flow direction.

Free Delivery Orders. In certain scenarios, the orders created by the planning phase are not enough to guarantee a valid pumping solution. For instance, suppose that only two orders need to be scheduled, and they have incompatible products. Consequently, one can not push the other in a pipeline. A third product must be used between them. For that, *free delivery orders* are arbitrary created before entering the scheduling phase. In contrast to regular orders, their volumes, products, and origin/destination tanks are treated as variables instead of constants, and they do not have a deadline. Note also that free orders may have a null volume associated with them. Furthermore, their routes are previously determined by choosing among the ones typically used by pipeline operators. Operators can change such routes by editing a configuration file.

Free orders are also used to represent products that remain in the pipeline at the end of the process, for the purpose of ensuring that all pipes are always completely filled. These orders do not have a destination tank, and constraints are used to indicate they are the last ones to be pumped into the pipelines.

Only minor changes to the previous model are necessary in order to accommodate free orders. Among them, in the order viewpoint, constraints where volume and product were constants should be changed to variables, and an *Alternative Resource Set* [12] can be used to indicate that an origin and destination tank must be assigned to free orders. The operation viewpoint remains unchanged.

Search Strategy. Different types of search strategies were tested for solving both the sequencing and the scheduling models. The currently implemented version is shown as Algorithm 1. It combines a *backtracking* mechanism [13] with a special variable ordering, being divided into three consecutive parts: *disjunctive components determination*, *adaptive backtracking* and *time assignment.* In the *Disjunctive Components Determination*, a disjunctive component is defined as a subset of the network which can be scheduled separately, without affecting other regions. Two pipelines belong to the same component if they are both contained in at least one order's route. The same reasoning applies to tanks.

For the *Adaptive Backtracking*, we implemented backtracking using positional variables for each pipeline. The term *adaptive* comes from the fact that it is based on a *restart* strategy [14]. As such, the pipeline sequencing order is changed dynamically according to the number of fails that occurred during the search. The values of positional variables are randomly chosen, giving higher probabilities to orders with the earliest deadlines. For free orders, volumes, products and tanks are set after their respective positional variables are labeled.

Algorithm 1. Procedure for search strategy

```
1  begin
2      Identify network *disjunctive components* $C$
3      for *each* $c \in C$ do
4          Build *pipe graph* $G(c)$ and sort it topologically, obtaining order $N$
5          $N' := N$; $k := initial\_k$
6          while $N' \neq \emptyset$ do
7              $p :=$ first element from sequence $N'$; $N' := N'/\{p\}$
8              Label positional variables, and volumes/tanks in case of free orders
9              if *fails in labeling* $\geq k$ *and not cyclic condition* then
10                 $k := k+$incremental factor; $N' := N$
11                 Move $p$ to the beginning of sequence $N'$
12     while *no scheduling solution found* do
13         Create scheduling model and assign times as earliest as possible
14         if *no solution* then request *next* sequencing solution
15 end
```

The initial sequencing is constructed as follows. Firstly a *pipe graph* is created, in which pipelines are nodes and there is a direct arc from node p to q if there is a consecutive pair (p, q) in some order's route. In case there are two arcs (p, q) and (q, p), only the one associated with the order having the earliest deadline is maintained. Secondly, the graph is topologically sorted, the result being the desired initial sequencing. Clearly, this strategy considers first those pipelines with the least number of orders that come directly from other pipelines.

After the occurrence of k fails involving a pipeline, the backtrack tree is reinitialized with that pipeline as the first element in the topological ordering, and k is incremented by a constant. This implementation was motivated by the fact that, during test runs, it was observed that a fair number of fails were caused by earlier decisions taken when instantiating variables in related pipelines in the given sequencing. We empirically determined $k = 150$ and 100 as the increment.

Finally, in *Time Assignment*, executed after the sequencing is completed, the CP scheduling model is created and the time variables are instantiated with the least possible value in their domains. This forces pumping to start as soon as possible. In case a failure ensues, a new sequencing solution is requested, most certainly a different one due to the randomization present in the model.

4 Results

Solutions were obtained on a *Intel Pentium D* 3.40 Ghz CPU platform, with 4GB of memory. The planning and scheduling phases were coded in C++ and compiled using *GCC-4.0*. The CP model was solved using *ILOG Solver 6.2* and *ILOG Scheduler 6.2*, with medium to high propagation enforcement. Part of a typical solution is presented in Table 1. As described earlier, a solution is a sequence of pump operations and each line in the table describes one such

Table 1. Solution Example

$\mathbf{T_i}$	$\mathbf{T_f}$	**Vol.**	**Pd**	$\mathbf{Tk_{Or}}$	$\mathbf{Tk_{Dt}}$	**Route**
2075	2362	858	*G*	T004	T005	SUG03
4857	4868	30	*N*	T160	T087	GUG03
4870	5111	722	*D*	T008	T005	BUG03
...	...	...	...	...	...	...

Table 2. Solver Results

Instance	1	2	3	4
Horizon	10 days	7 days	7 days	7 days
Orders	924	645	724	693
Planning Phase Time	4 min	5 min	4 min	6 min
Planning Phase Peak Memory	78MB	61MB	67MB	63MB
Sequencing Model Variables	37,326	21,381	25,938	24,315
Sequencing Model Constraints	382,565	148,075	160,302	155,409
Sequencing Choice Points	3,355	2,462	3,417	2,518
Sequencing Fails	2,301	1,291	987	1,902
Sequencing Time	2 min	1 min	1 min	1 min
Sequencing Peak Memory	450 MB	240 MB	310 MB	270 MB
Scheduling Model Variables	12,350	7,530	8,931	8,032
Scheduling Model Constraints	27,088	16,768	19,231	18,292
Scheduling Choice Points	1,516	1,164	801	1,810
Scheduling Fails	301	429	210	120
Scheduling Time	2 min	1 min	1 min	1 min
Scheduling Peak Memory	450 MB	250 MB	290 MB	280 MB
Total Time	8 min	7 min	6 min	8 min

operation. Column headings are as follows: T_i and T_f are the start and end times (in minutes), respectively; Vol is the pumped volume (in m^3); P_d is the product code (G is gasoline, N is naphtha and D is diesel); Tk_{Or} and Tk_{Dt} are origin and destination tank codes, respectively; and *Route* is the route code. A full solution table would contain several hundred such lines.

We used four real field instances to test the models. The first two rows in Table 2 indicate the time horizon and the number of deliver orders generated by the planning phase, respectively, for each of the test instances. The remaining lines give details of typical runs. All instances share the same network topology of 14 depots, 29 pipelines, 32 different product types and 242 tanks distributed among the depots. Pipelines volumes range from 30 to 8,000 m^3, and most of the tank capacities are between 4,000 and 30,000 m^3.

In all cases the solver found a solution in a reasonable amount of computer time, *e.g.*, within 10 minutes. Most variables were instantiated as a result of constraint propagation. The search heuristic, which proved crucial in the planning phase, was also instrumental to improve other important aspects of the solution quality, as noticed by logistic engineers. For instance, usually, a typical solution showed only a very small number of pipeline flow reversions, the kind of operation that engineers prefer to keep to a minimum. Also, new and interesting routes were identified. Some of them came as a surprise to logistic engineers, who were biased towards the same traditional routes they were using when manually planning the network operation.

5 Added Value and Conclusions

We proposed a novel procedure for generating feasible solutions for real instances stemming from planning and scheduling the operation of a very-large pipeline network used do move petroleum derivatives. The operation of such a network is subject to a complex set of physical and operational constraints, and it makes possible the delivery of oil and biofuel to local markets, as well as the storing of the excess production from refineries. Using the CP paradigm, these constraints were adequately modeled. Problems of this size and complexity, as known by the authors, would not be solved by other approaches reported in the literature to date, in which much of the difficult constraints and topologies are overlooked.

The procedure is already integrated with a proprietary flow simulation tool and the company is currently considering it for routine use on a daily basis. The tool has already proved its value, showing that it can save many valuable work hours of skilled engineers. Also, using the tool, many different planning and scheduling scenarios can be easily setup and quickly tested, by varying local demand needs and production schedules at refineries.

The present modeling and implementation stage was reached after 2 years of problem specification, data gathering, model development, and testing. As work progresses, it is expected that new constraints will be introduced. Such could include inventory management restrictions, limitations on energy use at specific time intervals and at specific depots, and shutdown periods or partial operation intervals for tanks and pipelines. When modeling such new constraints, we feel that the flexibility of the CP paradigm will again prove to be crucial.

As for future directions, one could implement more sophisticated search heuristics for both the planning and scheduling phases, making the overall approach capable of dealing with more specific instance classes. Finally, one could consider objective functions that would help guide the heuristics. This would provide a yardstick that could be used to gauge solution quality.

Acknowledgments. This research was supported by grants 05/57343-0 and 05/57344-7 from FAPESP and grants 301732/07-8, 478470/06-1, 472504/07-0, and 473726/07-6, 305781/2005-7 from CNPq. The authors are also grateful to Fernando Marcellino and the team of engineers from PETROBRAS-TI/SP.

References

1. Rejowski, R., Pinto, J.M.: Efficient MILP formulations and valid cuts for multiproduct pipeline scheduling. Comput. Chem. Eng. 28(8), 1511–1528 (2004)
2. Cafaro, D.C., Cerdá, J.: Optimal scheduling of multiproduct pipeline systems using a non-discrete MILP formulation. Comput. Chem. Eng. 28(10), 2053–2058 (2004)

3. Relvas, S., Barbosa-Póvoa, A.P.F.D., Matos, H.A., Fialho, J., Pinheiro, A.S.: Pipeline scheduling and distribution centre management - a real-world scenario at clc. In: 16th Eur. Symp. Comput. Aided Process Eng. and 9th Int. Symp. Process Syst. Eng., pp. 2135–2140. Garmisch-Partenkirchen, Germany (2006)
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. 45(23), 7841–7855 (2006)
5. Rejowski, R., Pinto, J.M.: A novel continuous time representation for the scheduling of pipeline systems with pumping yield rate constraints. Comput. Chem. Eng. 32, 1042–1066 (2008)
6. Relvas, S., Matos, H.A., Barbosa-Póvoa, A.P.F.D., Fialho, J.: Reactive scheduling framework for a multiproduct pipeline with inventory management. Ind. Eng. Chem. Res. 46(17), 5659–5672 (2007)
7. Crane, D.S., Wainwright, R.L., Schoenefeld, D.A.: Scheduling of multi-product fungible liquid pipelines using genetic algorithms. In: Proc. of the 1999 ACM Symposium on Applied Computing, San Antonio, USA, pp. 280–285 (1999)
8. Camponogara, E.: A-Teams para um problema de transporte de derivados de petróleo. Master's thesis, Instituto de Matemática, Estatística e Ciência da Computação, Universidade Estadual de Campinas, Campinas, Brazil (1995)
9. Moura, A.V., de Souza, C.C., Cire, A.A., Lopes, T.M.T.: Heuristics and constraint programming hybridizations for a real pipeline planning and scheduling problem. In: Proc. IEEE 11th Int. Conf. Comput. Sci. Eng., São Paulo, Brazil (to appear, 2008)
10. Hooker, J.N.: Integrated Methods for Optimization (International Series in Operations Research & Management Science). Springer, Secaucus (2006)
11. Cheng, B.M.W., Choi, K.M.F., Lee, J.H.M., Wu, J.C.K.: Increasing constraint propagation by redundant modeling: an experience report. Constraints 4(2), 167–192 (1999)
12. ILOG: ILOG Scheduler 6.2: User's Manual (2006)
13. Marriot, K., Stuckey, P.: Programming with Constraints: An Introduction, 1st edn. MIT Press, Cambridge (1998)
14. Kautz, H., Horvitz, E., Ruan, Y., Gomes, C., Selman, B.: Dynamic restarts policies. In: Proc. of the 18th Nat. Conf. on Artif. Intell., Edmonton, Alberta (July 2002)