Pipeline Transportation of Petroleum Products[1]

Ruy Luiz Milidiú
milidiu@inf.puc-rio.br
Artur Alves Pessoa
artur@inf.puc-rio.br
Eduardo Sany Laber
laber@cos.ufrj.br

PUC-RioInf.MCC37/00 September, 2000

Abstract: In this paper, we introduce a new model for pipeline transportation of petroleum products. We use a directed graph with arcs representing pipes and nodes representing locations. Moreover, we define an order as a requirement to transport an amount of a given product. Initially, we assume that all the ordered volumes are unitary. Orders are grouped into packages that are received on-line. A feasible solution is a pumping sequence for each new package such that all residual orders from the previous package are satisfied. The pumping restriction is required because products must be "pushed" by other products to move in pipelines. Our goal is to minimize the total pumping cost.

Furthermore, we consider the problem of finding a strategy for the next order package, when the graph is acyclic. For that, we propose the Batch-to-Pipe Assignment (BPA) algorithm. This algorithm finds a minimum cost solution in $O(n^3 + r^3 + rm)$ time, for m pipelines, n locations and r unitary orders. Moreover, we observe that this solution is also optimal for on-line package sequences. Finally, we show that our results can be extended to deal with general graphs and with non-unitary orders.

Keywords: Petroleum, Transportation, Oil Pipeline, Network Flows, Optimization

Resumo: Neste documento, introduzimos um novo modelo para o transporte de produtos de petróleo através de oleodutos. Utilizamos um grafo direcionado onde os arcos representam os dutos e os nós representam as áreas operacionais da rede. Inicialmente, assumimos que os volumes transportados são unitários. As requisições de transporte ou ordens são agrupadas em pacotes. Estes pacotes são recebidos *on-line*. Um solução viável associa a cada novo pacote uma sequência de bombeamentos tal que todas as ordens do pacote anterior são satisfeitas. A restrição quanto ao pacote anterior é necessária porque produtos contidos em um oleoduto precisam ser "empurrados" por outros produtos para se moverem. Nosso objetivo é minimizar o custo total de bombeamento.

Consideramos o problema de encontrar uma estratégia de bombeamento apenas para o próximo pacote de ordens, assumindo que o grafo é acíclico. Para este problema, propomos o algoritmo BPA. Este algoritmo encontra uma solução de custo mínimo em tempo $O(n^3 + r^3 + rm)$, para uma instância com m dutos, n áreas operacionais e r ordens unitárias. Observamos que esta solução também é ótima quando aplicada a uma sequência de pacotes de ordens recebidos *on-line*. Finalmente, mostramos que estes resultados podem ser estendidos para tratar grafos genéricos e ordens não unitárias.

Palavras-Chave: Petróleo, Transporte, Oleoduto, Fluxo em Redes, Otimização

[1]Sponsored by FINEP, CTPetro, and CENPES/Petrobras

1 Introdution

Petroleum products are typically transported in pipelines which are different from all other transportation methods. Pipelines use stationary carriers whose cargo moves rather than moving carriers of stationary cargo. Optimizing the transportation in pipelines is a problem of high relevance, since it leads to both technological and scientific issues. Nevertheless, as far as we know, just a few authors have specifically addressed this problem [HR95, Han91, Cam95]. Let us define an order as a requirement to transport a given amount of some product from one location to another. In [HR95], Hane and Ratliff presented a model that assumes that the orders are cyclic, that is, they always repeat after the completion of a given period. The authors also introduced the concept of pipeline *restarts*, which occur when the pipeline flow rate changes from zero to non-zero. In Hane's model, the objective is to minimize the total number of pipeline *restarts*. In this paper, we consider the problem of transporting petroleum products in pipelines with non-cyclic orders [MPL00].

An important characteristic of pipelines is that they must be always full. Hence, assuming incompressible fluids, an elementary pipeline operation is the following: pump an amount of product into the pipeline and remove the same amount of another (or the same) product from the opposite side.

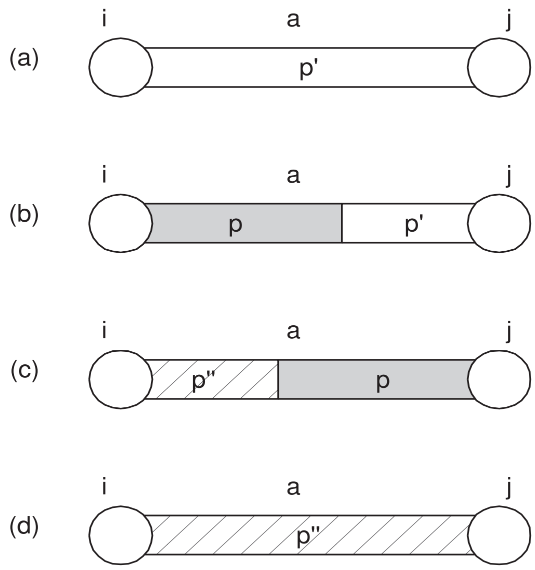

Figure 1: The transportation of a product p from location i to location j, using a pipeline a.

As an example, let us assume that we need to transport a volume v of the product p

from location i to location j. For that, we use a pipeline a that connects i to j, having an internal capacity of $v' > v$. Since pipelines must be always full, we also assume that there is a volume v' of the product p' inside a. This situation is illustrated by figure 1.(a). Figure 1.(b) shows what happens when we pump the volume v of p into the pipeline. Observe that the product removed at location j is p'. In this case, the delivery of p starts only when a volume $v' - v$ of product p'' has already been pumped into a. Figure 1.(c) represents the content of a at this moment. Finally, the transportation of p is completed when the product p'' completely fils the pipeline a, as shown in figure 1.(d).

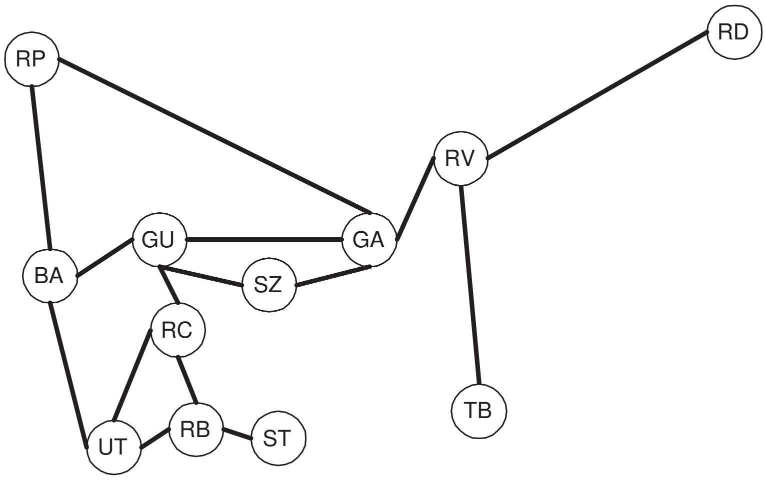

Figure 2: Petrobras' pipeline system in São Paulo state, Brazil.

Typical pipeline systems can be viewed as generic graphs where nodes represent locations and arcs represent pipelines. As an example, figure 2 shows a Petrobras' pipeline system[2]. It is located at the São Paulo state, in Brazil. Observe that each arc in the pipeline system must be full of residual products corresponding to previous orders. Now, let us consider a package of new transportation orders. Throughout this paper, we consider the following two natural questions:

1. When is it possible to advance the transportation of the new orders such that one satisfies all the previous ones?

2. Which one is the cheapest way to do it?

Motivated by these two questions, we propose a new model for pipeline transportation of petroleum products. In this model, we use a standard cost function for transportation

[2]Petrobras is a brazilian petrochemical industry.

problems. Moreover, we assume that the flow inside each pipeline is unidirectional. Hence, we model a pipeline system by a directed graph, where each node represents a location and each directed arc represents a pipeline, indicating the corresponding flow direction. We also assume that all the ordered volumes are unitary. Orders are grouped into packages that are received on-line. Throughout this paper, we use the term *batch* to denote the amount of product that corresponds to a given order. We assume that these batches cannot be split during the transportation. A feasible solution is a pumping sequence for each new package such that all residual orders from the previous package are satisfied. The pumping restriction is motivated by question one. Our goal is to minimize the total pumping cost.

We also consider the problem of finding a minimum cost strategy for the next order package, when the graph is acyclic. For that, we propose the Batch-to-Pipe Assignment (BPA) algorithm. This algorithm finds an optimal solution in $O(n^3 + r^3 + rm)$ time, for m pipelines, n locations and r orders. Furthermore, we observe that this solution is also optimal for on-line package sequences. Finally, we show that our results can be extended to deal with general graphs and with non-unitary orders.

This paper is organized as follows. In section 2, we formalize our model. In section 3, we describe the BPA algorithm for the problem of finding a minimum cost strategy for the next order package, when the graph is acyclic. In section 4, we show that the BPA algorithm can also be useful for general graphs. We also discuss how to extend the model to deal with non-unitary orders. Furthermore, we introduce an extension of the BPA algorithm for the improved model. Finally, in section 5, we summarize our conclusions and comment our future work.

2 The model

In this section, we formalize our model for pipeline transportation of petroleum products. The following simplifying assumptions are considered:

1. fluids are incompressible;

2. ordered volumes are unitary;

3. order packages are received on-line;

4. current orders must be satisfied before the receipt of a new order package;

5. flow rates are fixed;

6. location storages are unlimited;

7. batches cannot be split;

8. each batch pumping cost depends only on the chosen route.

2.1 The Pipeline System and the Orders

Let $G = (N, A)$ be a directed graph, where N is the set of nodes and A is the set of arcs. Arcs represent pipes and nodes represent locations. Each arc $a \in A$ has an associated integer capacity $v(a)$.

Let also O be a set of orders, where each order $k \in O$ determines that a batch with an integer volume $v(k)$ shall be transported from $s_k \in N$ to $d_k \in N$. To simplify our notation, let us also use k to denote the batch that corresponds to the order k, for $k \in O$. Furthermore, each batch $k \in O$ has an associated weight $w(k)$. Initially, we consider only the cases where all batch volumes are unitaries, that is, we have $v(k) = 1$, for all $k \in O$.

The orders of O are grouped into packages that are sequentially received on-line. Hence, let us define the corresponding list of order packages $[P_0, P_1, \ldots, P_q]$, sorted by receipt, where each order belongs to exactly one package.

2.2 Pipeline Contents

Now, let us discuss our representation of the pipeline contents at a given time. In a pipeline system, batches can be stored at arcs and at nodes. Let us assume that the storage capacity of each node is unlimited. On the other hand, the storage capacity of an arc $a \in A$ is exactly $v(a)$. We assume that all batch volumes are unitary and all arc capacities are integers. However, observe that the contents of an arc $a \in A$ at a given time can have two partial batches: one being pumped and another one being delivered at the opposite side. To simplify our model, let us skip these times and only represent the content of a pipeline system at the times where each arc $a \in A$ contains exactly $v(a)$ integral batches. In this case, the transportation process can be viewed as a discrete sequence of states, where the positions of all batches are well-defined.

Let us use the function $f_t(\cdot)$ to represent the content of each arc immediately before the arrival of the order package P_t, for $t = 1, \ldots, q$. If $f_t(a)$ returns the list $[b_1, b_2, \ldots, b_{v(a)}]$, then b_ℓ is the ℓth most recently pumped batch contained in the arc a, for $\ell = 1, \ldots, v(a)$. On the other hand, the function $p_t(\cdot)$ consistently gives the position of each batch immediately before the arrival of P_t. If $p_t(b) = a \in A$, then the batch b is contained at the arc a.

Otherwise, if $p_t(b) = i \in N$, then the batch b is stored at the node i. When a new order package P_{t+1} arrives, every batch $b \in P_{t+1}$ becomes available at the corresponding source node s_b. Furthermore, we define $p_t(b) = out$ when b is not contained in the pipeline system immediately before the arrival of P_t. For example, if either b has already been delivered or b belongs to an incoming order package, then we have $p_t(b) = out$.

We point out that a batch b is not necessarily delivered at the first time it passes by the corresponding destination node d_b. Instead, let us assume that b can pass by d_b as many times as needed.

Clearly, the state of the pipeline system immediately before the arrival of P_t depends on the way we process the orders of P_{t-1}. In this paper, we always assume that P_0 has already been received and processed and that both $f_1(\cdot)$ and $p_1(\cdot)$ represent the initial state of the pipeline system. Furthermore, we use $f_{q+1}(\cdot)$ and $p_{q+1}(\cdot)$ to denote the two functions that represent the final state of the pipeline system.

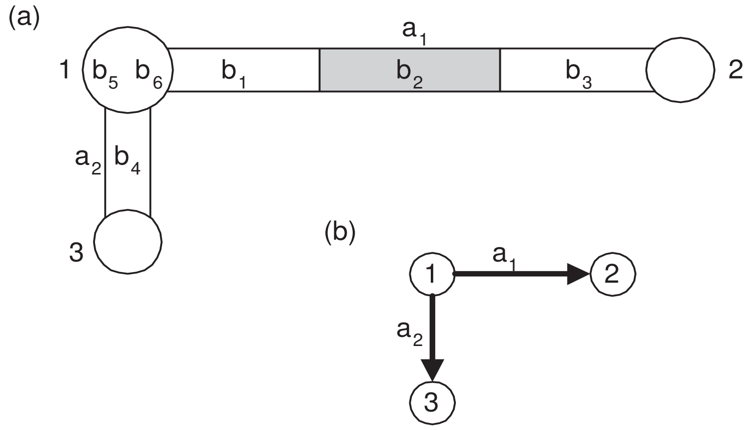

Figure 3: (a) The content of a pipeline system; (b) the corresponding graph.

As an example, figure 3.(a) represents the pipeline contents corresponding to the graph of figure 3.(b). Observe that the system has two pipelines a_1 and a_2, whose flow direction is indicated by the corresponding arcs. The two arcs have corresponding capacities $v(a_1) = 3$ and $v(a_2) = 1$. Let us assume that figure 3.(a) corresponds to the initial state. In this case, we have $f_0(a_1) = [b_1, b_2, b_3]$, since the three batches b_1, b_2 and b_3 are contained in a_1 and b_1 is the most recently pumped batch. Moreover, we have $f_0(a_2) = [b_4]$. Consistently, we have $p_0(b_1) = p_0(b_2) = p_0(b_3) = a_1$ and $p_0(b_4) = a_2$. Furthermore, we have $p_0(b_5) = p_0(b_6) = 1$ since both b_5 and b_6 are stored at node 1.

2.3 The Operation Restrictions

Now, let us define an elementary pipeline operation (EPO) for our model. Let $a = (i, j)$ be an arc of G, whose content at a given time t is given by the list $[b_1, b_2, \ldots, b_{v(a)}]$. Moreover, let b be a batch stored at node s at this time. An EPO is to pump b into a. As a result of this operation, b occupies the first position in a, $b_1, b_2, \ldots, b_{v(a)-1}$ are moved to their next positions, and $b_{v(a)}$ is either stored or delivered at node j. Hence, the content of a at the next time is given by the list $[b, b_1, b_2, \ldots, b_{v(a)-1}]$. As our first restriction, we state that all state changes in a pipeline system must correpond to an EPO.

Let us recall one of our key issues: "When is it possible to advance the transportation of the new orders such that one satisfies all the previous ones?" Motivated by this question, we add the following restriction to our model: *immediately before the arrival of P_{t+2}, for $t = 0, \ldots, q-2$, all the batches of P_t must have already been delivered.* This means that we must advance the transportation of the orders of P_t such that one satisfies all the orders of P_{t-2}. Throughout this paper, we respectively refer to the order packages P_t, P_{t+1} and P_{t+2} as the *current*, the *new* and the *next* order package. Observe that, due to the previous restriction, any batch b contained in the pipeline system immediately before the arrival of the next package P_{t+2} must belong to the new package P_{t+1}. Moreover, the pipeline system contains only batches of P_0 at the initial state and must contain only batches of P_q at the final state.

For instance, let us assume that we have $q = 2$, $P_0 = \{b_1, b_2\}$, $P_1 = \{b_3, b_4\}$, and $P_2 = \{b_5, b_6\}$. In this case, only the batches b_1 and b_2 can be contained in the pipeline system, at the initial state. These batches must be delivered before P_2 is received. For that, we shall use the batches b_3 and b_4 of P_1 to "push" them, since only these batches can be contained in the pipeline system immediately before the arrival of P_2. Both b_3 and b_4 must be delivered before the final state. Moreover, only the batches b_5 and b_6 can be contained in the pipeline system, at the final state.

Our last restriction is obvious: it must be possible to deliver every batch of P_q from its position at the final state, that is, for all $b \in P_q$, there must be a valid route from $p_q(b)$ to d_b.

2.4 The Objective Function

First, let us define the pumping cost of an EPO. Let a be an arc of G, whose content at a given time is given by the list $[b_1, b_2, \ldots, b_{v(a)}]$. If a batch b is pumped into a, then the pumping cost due to this operation is given by

$$w(b)/2 + w(b_{v(a)})/2 + \sum_{\ell=1}^{v(a)-1} w(b_\ell).$$

Observe that each term of the previous sum is due to one of the batches that are contained in a during the operation. Now, let us consider that the transportation of a given batch b uses the route R, defined as the sequence of arcs $a_1, a_2, \ldots, a_{|R|}$. In this case, the total pumping cost $c(b, R)$ due to b through R is given by

$$c(b, R) = w(b) \times \sum_{i=1}^{|R|} v(a_i).$$

Now, let us define *the local pumping cost* due to the transportation of a new order package P_{t+1}. We shall define this cost so that it reflects all the additional pumping cost due to the processing of the new package P_{t+1}. Here, we assume that the pumping cost to transport the batches of the current package P_t to their corresponding destinations, using a previously determined route, is not considered since it has already been charged. On the other hand, we consider the whole pumping cost associated to the route chosen for each batch of the new package P_{t+1}. Moreover, if a given batch of P_t is transported through a more expensive route, then we also charge the corresponding additional pumping cost.

To formalize this notion, we use R_t^b to denote the previously chosen route for the batch b of the current package P_t. A route for b is any sequence of arcs connecting s_b to d_b. As observed before, we may choose a more expensive route for b when processing the new package P_{t+1}, that is, we may have $R_t^b \neq R_{t+1}^b$. Hence, the total pumping cost c_{t+1} of processing the new package P_{t+1} is given by

$$c_{t+1} = \sum_{b \in P_t} (c(b, R_{t+1}^b) - c(b, R_t^b)) + \sum_{b \in P_{t+1}} c(b, R_{t+1}^b).$$

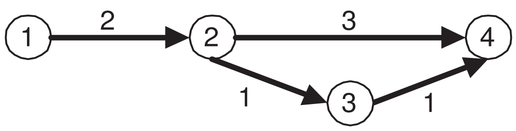

Figure 4: A graph representing a pipeline system with capacities as arc labels.

As an example, let us consider the graph represented in figure 4. In this figure, numbers inside the nodes are indices and arc labels are capacities. Let us assume that the batch $b \in P_t$ has weight $w(b) = 1$ and is initially contained in the arc $(1, 2)$. Moreover, the route

R_t^b is given by the sequence of arcs $(1,2),(2,3),(3,4)$. In this case, if we choose another route $R_{t+1}^b = [(1,2),(2,4)]$ for b, when processing the new package P_{t+1}, then the additional pumping cost due to b is $(2+3)-(2+1+1)=1$.

Now, let us consider a batch $b \in P_t$. Observe that one cannot change the subroute of the previously chosen route for b that has already been traversed. As a consequence, only the non-traversed subroute of this route can still be changed, when processing the new package P_{t+1}. To simplify our model, let us assume that this non-traversed subroute is always a shortest path. As a result, the pumping cost due to the batches of the current package P_t cannot decrease, when processing the new package P_{t+1}.

Finally, we define *the total pumping cost* c due to the transportation of a sequence of new order packages $P_1, P_2, \ldots, P_q$ as the sum of the corresponding local pumping costs, that is,

$$c = c_1 + c_2 + \cdots + c_q.$$

2.5 Problems

Now, we introduce four pipeline transportation problems (PTP). In this paper, these four problems are variations on the basic model proposed in this section. Next, we enumerate these problems:

the Local PTP: the problem of finding a minimum cost transportation for a new order package P_{t+1};

the On-line PTP: the on-line problem of finding a minimum cost transportation for a sequence of new order packages $P_1, P_2, \ldots, P_q$;

the Local Acyclic PTP: a variation of the Local PTP where the graph G is acyclic;

the On-line Acyclic PTP: a variation of the On-line PTP where the graph G is acyclic.

Next, we give an overview on both the input and the output data required by these problems. For the on-line problems, the following items are the required input data:

1. a graph $G = (N, A)$, where each arc $a \in A$ has an associated volume $v(a)$;
2. a set of orders O partitioned into a list of order packages $[P_0, P_1, \ldots, P_q]$, where each order $b \in O$ has the following attributes:

(a) a source node s_b,

(b) a destination node d_b,

(c) a batch weight $w(b)$;

3. an initial state for the pipeline system, determined by the functions $f_1(\cdot)$ and $p_1(\cdot)$.

For the local problems, the list of order packages is replaced by both the current and the new order packages, respectively denoted by P_t and P_{t+1}. Moreover, the initial state is replaced by a current state, determined by the functions $f_{t+1}(\cdot)$ and $p_{t+1}(\cdot)$.

Finally, the required output data is given by a sequence of EPO's. Each EPO may be defined by a triple (b, a, δ), where $b \in O$ is the pumped batch, $a = (i, j) \in A$ is the pipeline, and $\delta \in \{0, 1\}$ determines whether the remaining batch is delivered at the node j ($\delta = 1$) or not ($\delta = 0$). In a feasible output solution, immediately before starting each operation $(b, (i, j), \delta)$, the pumped batch b must be stored at the pumping node s. Observe that both the route R^b_{t+1} of each batch $b \in O$ and the final state of the pipeline system can be obtained from the given sequence EPO's. Therefore, they must also satisfy the restrictions imposed by the model.

3 The Batch-to-Pipe Assignment Algorithm

In this section, we propose the Batch-to-Pipe Assignment (BPA) algorithm. The BPA algorithm exactly solves both the Local and the On-line Acyclic PTP. Let $n = |N|$, $m = |A|$ and $r = |P_{t+1}|$. For the Local Acyclic PTP, it runs in $O(n^3 + r^3 + rm)$. Moreover, we point out that some of the following results also hold for the case where the graph G has one or more cycles. Therefore, we also use them to develop lower bounds for both the Local and the On-line PTP. Throughout this paper, we refer to the state of the pipeline system immediately before the arrival of the next package P_{t+2} as *the following state*.

3.1 A Necessary Condition for the Feasibility

Our first result is a simple necessary condition for an instance of the Local PTP to be feasible. Since the pipeline system must contain only batches of P_{t+1} at the following state, any feasible solution for an instance of the Local PTP assigns to each arc $a \in A$, exactly $v(a)$ distinct batches of P_{t+1}. This assignment determines which batches fill each arc at the following state. Therefore, each batch is assigned to at most one arc. Moreover, a batch

b can be assigned to an arc a only if there is a valid route R^b_{t+1} that passes by the arc a. Our necessary condition is that such an assignment exists.

Proposition 1 *A necessary condition for an instance I of the Local PTP to be feasible is that there exists an assigment from P_{t+1} to A with the following properties:*

1. *it assigns to each arc $a \in A$, exactly $v(a)$ distinct batches of P_{t+1};*
2. *each batch $b \in P_{t+1}$ is assigned to at most one arc of A;*
3. *if a batch b is assigned to an arc a, then there is a valid route R^b_{t+1} that passes by the arc a.*

3.2 Dependence Graphs

Now, let us define a dependence graph. Given

1. an instance of the Local PTP with a corresponding graph G;
2. a valid following state, defined by the functions $f_{t+2}(\cdot)$ and $p_{t+2}(\cdot)$;
3. a valid route R^b_{t+1} for each batch $b \in O$, consistent with the given following state.

the corresponding dependence graph $H = (A, D)$ is a directed graph defined as follows:

1. each node of H corresponds to an arc $a \in A$;
2. there is an arc $d = (a, a') \in D$ if and only if at least one of the following conditions hold:
   (a) there is a batch $b \in P_t$, with $R^b_{t+1} = [a_1, a_2, \ldots, a_{|R^b_{t+1}|}]$ and $p_{t+1}(b) = a_i$, such that $a_j = a$ and $a_{j+1} = a'$ for some $j \in \{i, i+1, \ldots, |R^b_{t+1}| - 1\}$;
   (b) there is a batch $b \in P_{t+1}$, with $R^b_{t+1} = [a_1, a_2, \ldots, a_{|R^b_{t+1}|}]$ and $p_{t+1}(b) = a_i$, such that $a_j = a$ and $a_{j+1} = a'$, for some $j \in \{1, 2, \ldots, i-1\}$.

For example, let us consider the graph G represented by figure 5.(a) and the corresponding batch positions and routes, given by table 1. Figure 5.(b) shows the resulting dependence graph H. Observe that there is an arc from a_2 to a_3 in H because the batch b_1 of P_t, with route $R^{b_1}_{t+1} = [a_2, a_3]$, is currently contained in the arc a_2. On the other hand, the route $R^{b_2}_{t+1} = [a_3, a_1]$ does not produce an arc from a_3 to a_1 because the batch b_2 of P_t

Figure 5: (a) A graph G; (b) the corresponding dependence graph H.

Table 1: An example of batch positions and routes

Batch b_i	Route $R_{t+1}^{b_i}$	Package P_t	Current pos $p_{t+1}(b_i)$	Following pos $p_{t+2}(b_i)$
b_1	$[a_2, a_3]$	P_t	a_2	*out*
b_2	$[a_3, a_1]$	P_t	a_1	*out*
b_3	$[a_3]$	P_t	a_3	*out*
b_4	$[a_1, a_2]$	P_{t+1}	out	a_2
b_5	$[a_3, a_1]$	P_{t+1}	out	a_3
b_6	$[a_1]$	P_{t+1}	out	a_1

is currently contained in the arc a_1. Observe also that there is an arc from a_1 to a_2 in H because the batch b_4 of P_{t+1}, with route $R_{t+1}^{b_4} = [a_1, a_2]$, and is contained in the arc a_2 at the following state. On the other hand, the route $R_{t+1}^{b_5} = [a_3, a_1]$ does not produce an arc from a_3 to a_1 because the batch b_5 of P_{t+1} is contained in the arc a_3 at the following state.

3.3 A Sufficient Condition for the Feasibility

Observe that, if there is an arc in H from a to a', then there is at least one batch that leaves the arc a and enters the arc a', when processing the package P_{t+1}. Hence, the dependence graph can be interpreted as follows: an arc in H from a to a' indicates that the execution of an EPO in a' may depend on the execution of some EPO in a. Conversely, the absence of this arc assures that such a dependence does not exist. This observation motivates the following theorem.

Theorem 1 *Given an instance I of the Local PTP, a valid following state and a valid route for each batch, if the resulting dependence graph H is acyclic, then there is a feasible solution for I that uses the given routes.*

Proof: We prove this theorem by showing how to construct such a feasible solution for I. Since H is acyclic, it has a source node. Let $a = (i, j)$ be an arc of G that corresponds to a source node in H. Every batch b that must pass by the arc a is stored at the node i at the current state, otherwise a would have an incoming arc in H. Hence, we start our feasible solution by a sequence of EPO's that pump these batches into the arc a. We point out that the tail of this sequence must correspond to the reversal of the list of batches $f_{t+2}(a)$ since it determines the content of the arc a at the following state. After this sequence is executed, every batch b that must pass by the arc $a = (i, j)$ is already stored at the node j. Moreover, every batch b contained in a must stay at this arc until the following state. Therefore, we may remove the node a (and all its outgoing arcs) from H. By repeating this procedure until the dependence graph H is empty, we assure that the content of every arc corresponds to the given following state after the produced solution is executed. ■

Theorem 1 has the following corollary.

Corollary 1 *The condition of proposition 1 is a sufficient condition for an instance I of the Local Acyclic PTP to be feasible.*

Proof: The condition of proposition 1 assures that there exists a valid following state and a valid route for each batch that is consistent with the given following state. Moreover, the dependence graph H corresponding to an acyclic graph G is also acyclic. Therefore, by theorem 1, there is a feasible solution for I. ■

We point out that, if the dependence graph H has one or more cycles, then, given a valid following state and a valid route for each batch, it is possible that no corresponding feasible solution exists. Next, we present an instance I of the Local PTP, a valid following state and a valid route for each batch with no corresponding feasible solution. For this instance, we have $P_t = \{b_1, b_2, b_3, b_4\}$ and $P_{t+1} = \{b_5, b_6, b_7, b_8\}$.

Figure 6.(a) represents the content of the pipeline system, immediately after the package P_{t+1} is received. In this figure, the batches of P_{t+1} are already stored at the corresponding source nodes. The graph G is represented by figure 6.(b). The pipeline system has four pipelines a_1, a_2, a_3 and a_4, whose flow direction is indicated by the corresponding arcs in G.

Table 2 contains additional information about the batches of the instance I. In this table, the second, the third, the fourth and the fifth columns respectively contain the current position, the source node, the destination node and the chosen route, for each batch. Furthermore, the last column of table 2 contains the position of each batch at the chosen following state. Figure 6.(c) shows the corresponding dependence graph H. Observe that H is a cycle.

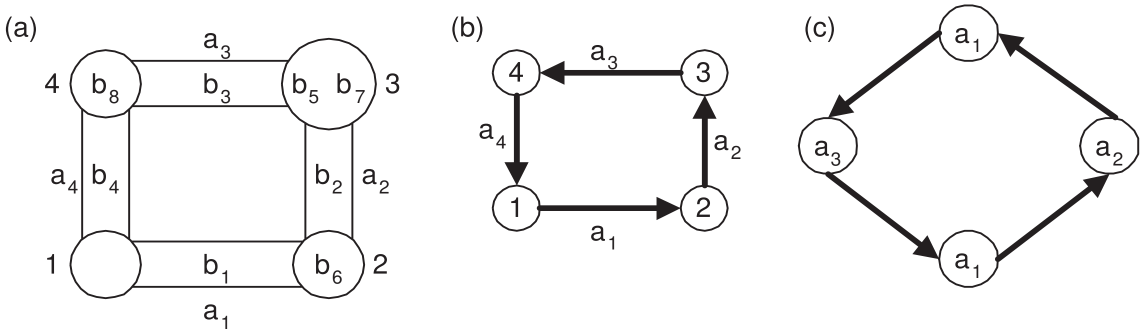

Figure 6: An instance of the Local PTP: (a) The content of the pipeline system immediately after the package P_{t+1} is received; (b) the corresponding graph G; (c) the corresponding dependence graph H.

Table 2: An instance of the Local PTP

Batch b_i	Curr pos $p_{t+1}(b_i)$	Src s_{b_i}	Dest d_{b_i}	Route $R_{t+1}^{b_i}$	Following pos $p_{t+2}(b_i)$
b_1	a_1	1	4	$[a_1, a_2, a_3]$	out
b_2	a_2	2	3	$[a_2]$	out
b_3	a_3	3	4	$[a_3]$	out
b_4	a_4	4	1	$[a_4]$	out
b_5	out	3	2	$[a_3, a_4, a_1]$	a_1
b_6	out	2	3	$[a_2]$	a_2
b_7	out	3	4	$[a_3]$	a_3
b_8	out	4	1	$[a_4]$	a_4

Now, we shall prove the there is no feasible solution that corresponds to the chosen routes and the chosen following state.

Proposition 2 *For the instance I, there is no feasible solution that corresponds to the chosen routes and the chosen following state.*

Proof: It is enough to prove that the batch b_1 cannot be delivered, using the chosen routes and the chosen following state. Observe that the b_5 is the only batch that can "push" b_1 to the node 2, that is, b_5 is necessarily contained in the arc a_1 when b_1 is stored at the node 2. Observe also that b_7 is the only batch that can "push" b_5 to the node 4. Since b_5 must reach the node 4 before the arc a_1, b_7 is necessarily contained in the arc a_3 when b_1 is "pushed" by b_5 to the node 2.

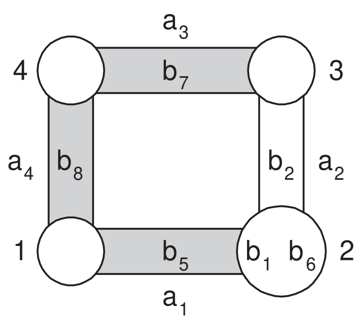

Figure 7: The content of the pipeline system when b_1 is stored at the node 2.

Figure 7 illustrates this situation. In this figure, b_1 is already stored at the node 2. For that, b_7 had to be pumped into the arc a_3, to "push" b_5, and b_5 had to be pumped into the arc a_1, to "push" b_1. The batches b_5, b_7 and b_8 are shaded in figure 7.

Since b_1 must reach the node 2 before the arc a_3, pumping b_1 into the arc a_3 would remove b_7 from this arc. Furthermore, since b_7 is contained in the arc a_3 at the given following state, b_1 cannot traverse the arc a_3. Therefore, b_1 cannot be delivered, using the chosen routes and the chosen following state. ■

3.4 An Exact Algorithm for the Local Acyclic PTP

Observe that both proposition 1 and corollary 1 state that the existance of an assignment from the batches of P_{t+1} to the capacity units of the arcs of G is a necessary and sufficient condition for an instance I of the Local Acyclic PTP to be feasible. Observe also that such an assignment yields to a following state for I. Moreover, the proof of theorem 1 gives a method for finding a feasible solution for I, when all batch routes and the following state are given. Since the cost of a solution dependes only on the chosen routes, we can always choose the cheapest valid route that is consistent with the given following state. Hence, the cost of assigning a batch b of P_{t+1} to a capacity unit of an arc a is exactly the pumping cost of the cheapest route for b that passes by a. Now, let us consider a batch b that either belongs to P_t or is not assigned to any arc of G. Since these batches are not contained in the pipeline system at the following state, we can always choose the cheapest route for them.

Based on the previous observations, we propose the BPA algorithm for the Local Acyclic PTP. BPA reduces an instance I of the Local Acyclic PTP to an instance of the well-known assignment problem [AMO93]. Let $u = \sum_{a \in A} v(a)$. First, BPA checks if $r < u$. In the positive case, the algorithm stops since the problem is infeasible. Otherwise, BPA performs the four main steps summarized in the pseudo code of figure 8.

The BPA Algorithm	
Step 1:	For every pair of nodes $i, j \in N$, find the shortest path $S(i, j)$ from i to j;
Step 2:	construct a bipartite graph B, where each batch of P_{t+1} corresponds to a node from one partition and each arc of G corresponds to exactly $v(a)$ nodes from the other partition;
Step 3:	solve the corresponding assignment problem;
Step 4:	find a feasible solution corresponding to the assignment obtained at step 3, using the method given by the proof of theorem 1.

Figure 8: A pseudo-code for the BPA algorithm.

At the first step, the length of each arc $a \in A$ is assumed to be the corresponding arc capacity $v(a)$. For the graph constructed at the step 2, the batch b corresponds to the node $n(b)$, for all $b \in P_{t+1}$. Moreover, for each arc a of G the algorithm creates the corresponding nodes $n(a, 1), n(a, 2), \ldots, n(a, v(a))$. Let $v(s, d)$ be the length of the shortest path from s to d. If there is no path from s to d, then let $v(s, d) = \infty$. Given a batch b, an arc $a = (i, j)$ and an index $k \in \{1, 2, \ldots, v(a)\}$, the cost of associated to the arc $(n(b), n(a, k))$ of the graph B is given by

$$w(b)(v(s_b, i) + v(a) + v(j, d_b)).$$

Consistently, if the arc $(n(b), n(a, k))$ is used by the assignment obtained at the step 3, then the route of b is given by the concatenation of the three paths $S(s_b, i)$, $[a]$ and $S(j, d_b)$. In this case, we have $p_{t+2}(b) = a$. Furthermore, if we have $r > u$ then $r - u$ additional nodes are created for the graph B. We use $n(out, k)$ to denote the kth additional node, for $k = 1, 2, \ldots, r - u$. Given a batch b and an index $k \in \{1, 2, \ldots, r - u\}$, the cost of associated to the arc $(n(b), n(out, k))$ of the graph B is $w(b)v(s_b, d_b)$. Consistently, if the arc $(n(b), n(out, k))$ is used, then the route of b is the shortest path from s_b to d_b and b is delivered before the following state. Finally, the route of each batch $b \in P_t$ is the shortest valid route, that is, if b is either contained at the arc $a = (i, j)$ or stored at the node j, then the tail of the route R^b_{t+1} is the shortest path from j to d_b. Since we assumed that such a route is equal to R^b_t, there is no additional cost due to the batches of P_t in this case.

If the assignment obtained at the step 3 uses an arc of the graph B with associated cost equal to ∞, then the problem is infeasible. Otherwise, the cost of this assignment is equal to the pumping cost due to the solution obtained at the step 4.

We shall mention that step 4 can be performed without constructing the dependence graph H, when the graph G is acyclic. For that, we use the following fact.

Fact 1 *Given a graph G and a corresponding dependence graph H, if a node i of G is a source, then all its outgoing arcs in G correspond to source nodes in H.*

Using the fact 1, we can perform step 4 as described in the pseudo-code of figure 9.

```
Let X be an empty sequence of EPO's;
While G is not empty do
        let i be a source node of G;
        For each outgoing arc a of i do
                add the EPO's of a to X;
        Done;
        remove i and all its outgoing arcs from G;
Done;
Let X be the final solution.
```

Figure 9: A pseudo-code for the step 4 of BPA when G is acyclic.

3.4.1 Analysis

Now, we present an analysis for the BPA algorithm. The following theorem considers the obtained solution for the Local Acyclic PTP.

Theorem 2 *The feasible solution obtained by the BPA algorithm is optimal for the Local Acyclic PTP.*

Proof: Let us consider the pseudo-code of the BPA algorithm, given by figure 8. By proposition 1, every feasible solution for an instance I of the Local Acyclic PTP corresponds to a feasible assignment for the graph B, constructed at the step 2. Furthermore, the cost of an optimal solution for I is not smaller than the cost of its corresponding assignment at the graph B. This is true because the cost of this assignment is defined as the sum of the cheapest corresponding routes. Since the cost of the solution obtained at the step 4 is equal to the cost of an optimal assignment for the graph B, this solution is optimal. ■

Observe that a feasible solution for the On-line Acyclic PTP can be obtained on-line by executing the BPA algorithm, for each incoming order package P_{t+1}, for $t = 0, \ldots, q-1$. The following theorem considers this solution.

Theorem 3 *The feasible solution obtained through succesive executions of the BPA algorithm is optimal for the On-line Acyclic PTP.*

Proof: Observe that the local cost of the solution obtained by BPA for the package P_{t+1} does not depend on the state of the pipeline system before the arrival of this package, for $t = 0, \ldots, q-1$. Since the solution obtained by BPA minimizes the local cost for each incoming package P_{t+1}, for $t = 0, \ldots, q-1$, it is optimal for the On-line Acyclic PTP. ■

Furthermore, the following theorem considers the execution time of BPA.

Theorem 4 *The BPA algorithm runs in $O(n^3 + r^3 + rm)$ time.*

Proof: If we have $r > u$, then the theorem holds since BPA stops before step 1. Hence, let us consider steps in the pseudo-code of figure 8, performed only when $r \leq u$. Step 1 can be executed in $O(n^3)$ time, using the Floyd-Warshall algorithm [Flo62, AMO93]. Moreover, the graph B constructed at step 2 has exactly $2r$ node and r^2 arcs. Clearly, this construction requires $O(r^2)$ time, since all shortest paths are already obtained at step 1. Furthermore, the resulting assignment problem requires $O(r^3)$ time to be solved at step 3, using the Successive Shortest Paths algorithm [Iri60, AMO93]. Finally, step 4 is performed as described in the pseudo-code of figure 9. In this case, finding the n source nodes requires $O(m)$ time if the indegree of all nodes are previously computed and maintained during the execution. This is the well-known Topological Ordering algorithm [AMO93]. In addition, the algorithm appends to X at most r EPO's, for each arc of G. Hence, step 4 requires $O(rm)$ time. As a result, the execution time of steps 1, 2, 3 and 4 is bounded above by $O(n^3) + O(r^2) + O(r^3) + O(rm) = O(n^3 + r^3 + rm)$. ■

4 Extensions of the BPA Algorithm

In this section, introduce two extensions for the BPA algorithm. The first extension is called the Cycle BPA algorithm and deals with general graphs. For the second extension, we introduce a variation of our model that allows non-unitary orders. We show that the

BPA algorithm can also be extended to solve the problems related to this variation. The resulting method is called the Flow BPA algorithm.

4.1 The Cycle BPA algorithm

Now, we propose the Cycle BPA algorithm. Given an instance I of the Local PTP, Cycle BPA either obtains an optimal solution for I or provides a lower bound on its optimum cost. Let us consider the pseudo-code of the BPA algorithm, given by figure 8. Observe that steps 1, 2 and 3 can be executed when the graph G contains one or more cycles. Moreover, by theorem 1, if the dependence graph H is acyclic, then step 4 can also be executed in this case. Hence, we define the Cycle BPA algorithm as an extension of the BPA algorithm that constructs the dependence graph H and checks whether it is acyclic or not before the execution of step 4. If it is acyclic, then it executes step 4 and returns an optimal solution for the problem. Otherwise, it returns the cost calculated at the step 3. A pseudo-code for the Cycle BPA algorithm can be obtained by including the following step in the pseudo-code of figure 8.

```
Step 3½:  construct the corresponding dependence graph H;
          if H has one or more cycles then
                stop and return the cost obtained at step 3;
```

Observe that the Cycle BPA algorithm always returns the value of a cost. By the following proposition, this cost is a lower bound on the optimum cost for the Local PTP.

Proposition 3 *The cost returned by the Cycle BPA algorithm is a lower bound on the optimum cost for the Local PTP.*

Proof: Proposition 1 provides a necessary condition for a following state to be valid. Moreover, the cost obtained at step 3 is the smallest cost associated to a following state with this necessary condition. Given a feasible solution, since it leads to a valid following state, its cost is not smaller than that obtained at step 3. ■

4.2 The Flow BPA algorithm

Next, we introduce a variation of our model that allows non-unitary orders. In the improved model, an order k has an integer total volume $v(k)$. Moreover, each batch that correponds to the order k is may have any volume not smaller than one and not greater than $v(k)$.

The minimum volume restriction imposed to each batch is necessary because, in the real problem, we have mixtures at the batch interfaces. Observe that, by properly setting the volume unit, the minimum volume restriction can be used to assure that the ratio between mixtured and non-mixtured volumes is always small. As a result, mixtures can be ignored by the model.

Now, let us introduce four extended pipeline transportation problems:

1. the Local Acyclic Non-unitary PTP;
2. the On-line Acyclic Non-unitary PTP;
3. the Local Non-unitary PTP;
4. the On-line Non-unitary PTP.

In all cases, if we drop the word "Non-unitary", then we obtain the corresponding problem with unitary orders.

Fortunately, the results presented in both sections 3 and 4.1, have a direct extension in the improved model. For example, it is easy to see that proposition 1, theorem 1 and corollary 1 are all valid for the improved model.

Now, we briefly describe the Flow BPA algorithm, an extension of the BPA algorithm that solves both the Local and the On-line Acyclic Non-unitary PTP. For that, let us consider the pseudo-code of figure 8. A pseudo-code for Flow BPA is similar. Step 1 is identical to that of BPA. At step 2, Flow BPA constructs directed bipartite graph B as follows:

1. for each order k of P_{t+1}, create a corresponding node at the first partition, with an associated *excess* $v(k)$;
2. for each arc a of G, create a corresponding node at the second partition, with an associated *demand* $v(a)$;
3. if $\sum_{a \in A} v(a) < \sum_{k \in P_{t+1}} v(k)$, then create a dummy node at the second partition with an associated *demand*
$$\sum_{k \in P_{t+1}} v(k) - \sum_{a \in A} v(a);$$

4. for each pair (k, a) with $k \in P_{t+1}$ and $a = (i, j) \in A$, create an arc from the node of the order k to the node of the arc a, with an associated *capacity* ∞ and an associated *cost* given by

$$w(k)(v(s_k, i) + v(a) + v(j, d_k));$$

5. for each order k of P_{t+1}, create an arc from the node of the order k to the dummy node, with an associated *capacity* ∞ and an associated *cost* given by

$$w(k)v(s_k, d_k).$$

At step 3, Flow BPA solves the *minimum cost flow problem* [AMO93], for the constructed graph B. In the obtained solution, a flow f in the arc that connects the node of the order k to the node of the arc a represents a batch b with the following attributes:

- b has volume f;
- the route of b is the shortest route from s_k to d_k that passes by the arc a;
- b is contained at the arc a, at the following state.

By means of the optimal solutions obtained by minimum cost flow algorithms when excesses, demands and capacities are integers, every obtained batch has an integer volume. Therefore, the minimum volume restriction is always satisfied.

As in the BPA algorithm, the step 4 of Flow BPA uses the procedure given by the pseudo-code of figure 9. We only remark that the final solution must be represented in an more compact way, otherwise its size is not polynomially bounded. For that, we define an *extended pipeline operation* (XPO) as the action of pumping a batch with *any volume* into a pipeline and storing the remaining batches at the node in the opposite side. For the improved model, the final solution is represented by a sequence of XPOs. Since each arc in the graph B corresponds to at most one batch, we have at most rm batches, where r is the number of orders. Therefore, the size of the final solution obtained by Flow BPA is $O(rm^2)$.

As a final remark, we point out that the correctness of Flow BPA can be shown as a direct extension of theorems 2 and 3. Furthermore, it is easy to show that Flow BPA runs in a polynomial time.

5 Conclusions

In this paper, we introduce a new model for pipeline transportation of petroleum products. The following two questions motivate this model:

1. When is it possible to advance the transportation of new orders such that one satisfies all the previous ones?

2. Which one is the cheapest way to do it?

Furthermore, the following simplifying assumptions are considered:

1. fluids are incompressible;

2. ordered volumes are unitary;

3. order packages are received on-line;

4. current orders must be satisfied before the receipt of a new order package;

5. flow rates are fixed;

6. location storages are unlimited;

7. batches cannot be split;

8. each batch pumping cost depends only on the chosen route.

Based on this model, we define the following four pipeline transportation problems:

the Local PTP: the problem of finding a minimum cost transportation for a new order package;

the On-line PTP: the on-line problem of finding a minimum cost transportation for a sequence of new order packages;

the Local Acyclic PTP: a variation of the Local PTP where the graph is acyclic;

the On-line Acyclic PTP: a variation of the On-line PTP where the graph is acyclic.

So, we propose the Batch-to-Pipe Assignment (BPA) algorithm. This algorithm obtains an optimal solution for the Local Acyclic PTP in $O(n^3 + r^3 + rm)$ time. For the On-line Acyclic PTP, we prove that the solution obtained through succesive executions of the BPA algorithm is optimal. Furthermore, we show how to extend the BPA algorithm to deal with general graphs and with non-unitary orders.

As enhancements to the current model, we plan to include the following additional restrictions:

1. due dates;
2. orders with multiple sources and multiple destinations;
3. batch interfaces depending on product compatibility;
4. limited storage at locations.

Furthermore, we aim to determine the complexity of several problems derived from extensions of our model. As a result, we shall provide a theoretical background for developing effective methods to optimize the transportation of petroleum products in pipelines.

References

[AMO93] R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows: Theory, Algorithms and Applications.* Prentice Hall, 1993.

[Cam95] Eduardo Camponogara. A-teams para um problema de transporte de derivados de petróleo. Master's thesis, Departamento de Ciência da Computação, IMECC - UNICAMP, December 1995.

[Flo62] R. W. Floyd. Algorithm 97: Shortest path. *Communications of ACM*, 5:345, 1962.

[Han91] Christopher A. Hane. *Scheduling Multi-Product Flows in Pipelines.* PhD thesis, Georgia Institute of Technology, School of Industrial and Systems Engineering, September 1991.

[HR95] Christopher A. Hane and H. Donald Ratliff. Sequencing inputs to multi-commodity pipelines. *Annals of Operations Research*, 57, 1995. Mathematics of Industrial Systems I.

[Iri60] M. Iri. A new method of solving transportation-network problems. *Journal of the Operations Research Society of Japan*, 3:27–87, 1960.

[MPL00] Ruy L. Milidiú, Artur A. Pessoa, and Eduardo S. Laber. Transporting petroleum products in pipelines (abstract). In *ISMP 2000 – 17th International Symposium on Mathematical Programming*, pages 134–135, Atlanta, Georgia, USA, August 2000.