Pipesworld: Planning Pipeline Transportation of Petroleum Derivatives [*]

Ruy Luiz Milidiú and **Frederico dos Santos Liporace** and **Carlos José P. de Lucena**
Departamento de Informática PUC-Rio
Av. Marquês de São Vicente 225, RDC
CEP: 22453-900 Gávea
Rio de Janeiro, RJ, Brazil
{milidiu,liporace,lucena@inf.puc-rio.br}

Abstract

This paper describes the Pipesworld planning domain, inspired by the pipeline transportation of petroleum derivatives. Most transportation problems consist of moving carriers of stationary cargos. Pipelines are unique in the sense that they are stationary carriers of moving cargos. As a consequence, the planning problem of these systems has singularities that make them very challenging. We present a PDDL model of the domain. Next, we comment on valid plans for some illustrative instances. We focus on the domain's particularities, and suggest how the model may be enhanced to take into account real world restrictions.

Introduction

Pipelines play an important role in the transportation of Petroleum and its derivatives, since it is the most effective way to transport large volumes over large distances.

Typically, oil pipelines are a few inches wide and several miles long. As a result, reasonable amounts of distinct products can be transported through the same pipeline with a very small loss due to the mixing at liquid boundaries.

Pipeline management is a complex task, where planning and logistics are important issues, among others like maintenance and environmental safety.

Optimizing the transportation through oil pipelines is a problem of high relevance, since a non negligible component of a petroleum product's price depends on its transportation cost. Nevertheless, as far as we know, just a few authors have specifically addressed the problem (Hane & Ratliff 1995; Camponogara 1995). In (Milidiú, Pessoa, & Laber 2002b) they show that finding a solution to some subproblems is hard, and propose a polynomial time algorithm to solve an special case. In (Milidiú, Pessoa, & Laber 2002a) they discuss the approximability of one subproblem and propose an approximate algorithm for it.

This paper is organized as follows. First, we introduce the pipeline transportation problem in general. Next, we present a simplified PDDL model of the domain. Then we analyze some illustrative instances and valid plans to solve them. Each one of the chosen instances highlights a particular problem characteristic. Finally, we describe the planned model extensions.

Pipeline Transportation

The pipeline transportation problem has an unique characteristic, which distinguishes it from other transportation methods: it uses stationary carriers whose cargo moves rather than the more usual moving carriers of stationary cargo.

These pipeline networks may be very long and complex. An example of a company dedicated to pipeline management is Transpetro, the transportation company of Petrobras, the Brazilian state-run oil company. Altogether, Transpetro operates more than 6700 kilometer of pipelines.

Pipelines may be divided in two major groups, based on the nature of the cargo: those that transport liquid and those that transport gas. This paper describes the liquid pipeline transportation problem, more specifically multi-commodity liquid pipelines, where more than one product may be transported. The term product will be used in this paper to refer to petroleum derivatives, such as diesel, gasoline and aviation kerosene.

The main components of a pipeline network are operational areas and the pipeline segments. Operational areas may be distribution centers, ports or refineries. These areas are connected by one or more pipeline segments. The oil derivatives are moved between the areas through the pipelines.

In the sections that follow, we describe the main characteristics of the domain and how they affect planning.

Pressurized pipelines

Liquid pipelines must always, for safety reasons, be pressurized. That is, they must be completely full of liquid. A typical operation in a pipeline segment $S_{1,2}$ that connects areas A_1 and A_2, as illustrated in Figure 1, is to pump some amount of liquid from an area A_1 and receive the same amount of liquid, assuming incompressible fluids, in area A_2.

As the pipeline segment may be filled with distinct products, the product that area A_2 receives is not necessarily the same that area A_1 pumps. For instance, if the pipeline segment is initially filled with diesel, and area A_1 pumps some

[*]Sponsored by Conselho Nacional de Desenvolvimento Cient´ıfi co e Tecnol´ogico, CNPq. The authors would also like to thank Artur Pessoa for important insights on the construction of the example instances.

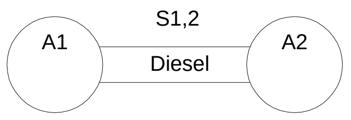

Figure 1: Pipeline network example

amount of gasoline, A_2 will receive initially the diesel that was previously stored in the pipeline segment. Moreover, A_2 only begins to receive the gasoline originated from A_1 after receiving all the diesel volume that was initially in $S_{1,2}$.

Interface restrictions

As distinct products have direct contact inside the pipeline segment, it is unavoidable that there is some loss due to the mixture in the interface between them. These interface losses are a major concern in pipeline operation, because the mixed products can not be simply discarded, they must pass through a special treatment that usually involves sending them back to a refinery and requires special tanks.

The severity of interface losses depends on the products that interface inside the pipeline segment. If two products are known to generate high interface losses, the pipeline schedule must not place them adjacently into the segment.

The interface restrictions deals with that situation. An interface graph G is defined for the products that are moved by the pipeline. In this graph, each product P_i is represented by a node. An edge between nodes P_i and P_j indicates that the interface losses between P_i and P_j are acceptable.

Reversion

In general, the content of a pipeline segment $S_{i,j}$ may be moved from A_j to A_i or from A_i to A_j. We say in that case that the segment is reversible. If the segment is not reversible, a single operational flow direction is defined.

The interface losses are higher when the flow in the pipeline segment is started or stopped, due to the transient in the flow regime until it reaches an steady state. As a consequence, a pipeline schedule should use reversions only when strictly necessary.

Storage in Areas

The products may be stored in tanks, located in the areas. Each tank may be used for a single product and have a maximum volume capacity. The tanks also have minimum and maximum recommended operational volumes. It is desirable, although not imperative, that the tankage volume is kept between these recommended values during the pipeline operation.

Figure 2 shows an example on how tankage may affect the pipeline operation. Let's call $T_{i,j}$ the tankage for product i in area j. Suppose that A_2 has a demand for product P_2, but its tankage current volume for P_3, $T_{3,2}$ is at the maximum level. It is not possible to directly move the P_2 volume that is inside $S_{1,2}$ to A_2 because there is no available storage for P_3 in A_2. The two only alternatives are: if the segment is reversible, move all $S_{1,2}$ contents to A_1, and then start the pumping from A_1 with P_2 or a product for which storage is available at A_2; or wait until there is demand for P_3 in A_2, which will free some space in $T_{3,2}$.

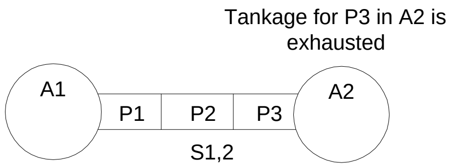

Figure 2: Instance with tight tankage restriction

Operational flow rates

Each segment $S_{i,j}$ may define an operational flow rate range $(f_{min,i,j}, f_{max,i,j})$. This rate is defined as product volume per unit of time. The upper bound of the range, $f_{max,i,j}$ is fixed by the pumps power. More powerful pumps may pump a greater amount of products in less time. The lower bound of the range, $f_{min,i,j}$ is a physical pump restriction. It is also related to interface losses, since these losses are higher for small flow rates. In fact, we say that two products P_i and P_j may interface if there is an edge between the nodes they represent in the interface graph G *and* the minimum flow rate for the segment is respected.

A segment $S_{i,j}$ content may be either stopped or moving in a rate between $f_{min,i,j}$ and $f_{max,i,j}$.

Demands and Productions

The pipeline schedule purpose is to elaborate a sequence of segment content movements such as there are available products at the areas where it is demanded, and there is no tankage violation in the areas where the products are received by the pipeline, typically refineries or ports.

Usually the demands and productions have an associated timestamp and flow rate, fixing when they must be started and the flow rate in which they will occur.

Pipesworld model

This section presents an annotated PDDL (Ghallab *et al.* 1998) model of the Pipesworld domain and problem instances, which takes into account the main characteristics of the pipeline transportation problem.

The name Pipesworld is derived from the similarities between the proposed model and the classical Blocksworld domain, in the sense that each pipeline segment may be modeled as a block stack, with the main difference being that each stack must keep its size constant.

Model concepts

We introduce in this section some concepts used to translate the pipeline transportation problem to the PDDL model.

Unitary batches The term batch is used in the oil pipeline industry to refer to an amount of the same product that must be transported through the pipeline. Batches are thus associated to a single product and have predefined volume.

Batches are also indivisible, once a batch B_i is pumped from an area A_j into a segment $S_{j,k}$, it is not possible for another batch to be pumped from A_j into $S_{j,k}$ until all B_i volume is pumped.

The proposed model defines the concept of *unitary batches*. Each batch is composed from one or more unitary batches. Unitary batches are indivisible portions of some product. The pipeline segments' volumes and tankage are defined in terms of unitary batches. The real volume associated to a unitary batch may be adjusted based on the instance characteristics. Smaller unitary batches decrease the rounding error that occurs when the batches are converted to unitary batches, but also increase the instance size, as more unitary batches are required to represent a single batch.

Push and Pop actions Pipeline segments always connect two areas. One area plays the "FROM" role, and the other plays the "TO" role. For each pipeline segment a default flow direction is defined. The default flow direction moves the pipeline contents towards the "TO" area.

The proposed model defines only two actions, PUSH and POP. The PUSH action moves the pipeline segment contents in the default direction, and the POP action moves the contents in the reverse direction.

Annotated PDDL model

```
;; PipesWorld
(define (domain pipelines)

(:requirements :typing :fluents)

;; Types:
;;  tank: represent product tankages
;;  batch-atom: an unitary batch
;;  pipe: a pipeline segment
;;  area: operational areas
;;  product: an oil derivative product, such as gasoline,
;;    kerosene, etc.
(:types tank batch-atom pipe area product)

(:predicates

;; Indicates that a pipeline segment connects
;; two areas
(connect ?from ?to - area ?pipe - pipe)

;; Tankages location and stored product
(tank-in-area ?tank - tank ?area - area)
(tank-product ?tank - tank ?product - product)

;; These predicates represent the pipeline segment contents
;; We define the first (nearest to  ‘‘from’’ area) and
;; last (nearest to  ‘‘to’’ area) batch-atom in a pipeline
;; segment, and their sequence is represented by the
;; ‘‘follow’’ predicate
(last ?batch-atom - batch-atom ?pipe - pipe)
(first ?batch-atom - batch-atom ?pipe - pipe)
(follow ?next ?previous - batch-atom)

;; An unitary batch product
(is-product ?batch-atom - batch-atom ?product - product)

;; Unitary batches that are on tankages
(on ?batch-atom - batch-atom ?tank - tank)

;; Indicates that two products may interface in the
;; pipeline segment
(may-interface ?product-a ?product-b - product)

)

;; Define the products (petroleum derivatives)
(:constants lco gasoleo rat-a oca1 oc1b - product)

;; Represent tankage capacity and current volume
(:functions (tank-capacity ?tank - tank)
            (tank-current-volume ?tank - tank))
```

```
;; Push action
;; Moves a batch-atom from a tankage to a pipeline segment
;; The Push action moves the pipeline segment contents towards
;; the ‘‘to-area’’ defined in the ‘‘connect’’ predicate
(:action PUSH
  :parameters(
    ;; Pipeline segment that will be moved
    ?pipe - pipe
    ;; Unitary batch that will be inserted into the pipeline
    ;; segment.
    ?batch-atom-in - batch-atom
    ;; Tankage that will receive the unitary batch that will
    ;; leave the pipeline segment
    ?destination-tank - tank
 )
  :vars(
    ?from-area - area
    ?to-area - area
    ?from-tank - tank
    ?first-batch-atom - batch-atom
    ?last-batch-atom - batch-atom
    ?product-batch-atom-in - product
    ?product-first-batch - product
    ?product-last-batch - product
  )
  :precondition
   (and
   ;; Binds :vars section
   (first ?first-batch-atom ?pipe)
   (last ?last-batch-atom ?pipe)
   (connect ?from-area ?to-area ?pipe)
   (on ?batch-atom-in ?from-tank)
   ;; Bind batch-atom products
   (is-product ?batch-atom-in ?product-batch-atom-in)
   (is-product ?first-batch-atom ?product-first-batch)
   (is-product ?last-batch-atom ?product-last-batch)
   ;; Interface restriction
   (may-interface ?product-batch-atom-in ?product-first-batch)
   ;; Tanks must be in correct areas
   (tank-in-area ?from-tank ?from-area)
   (tank-in-area ?destination-tank ?to-area)
   ;; Check if the recipient tankage may receive the product
   (tank-product ?destination-tank ?product-last-batch)
   ;; There must be space available at destination tank
   (< (tank-current-volume ?destination-tank)
      (tank-capacity ?destination-tank) )
 )
  :effect
   (and
     ;; The inserted unitary batch will be the pipeline segment
     ;; new first batch
     (first ?batch-atom-in ?pipe)
     (not (first ?first-batch-atom ?pipe))
     ;; Updates the follow and last relationship to the new
     ;; pipeline segment configuration
     (forall (?batch - batch-atom)
       (when (follow ?last-batch-atom ?batch)
         (and (last ?batch ?pipe)
              (not (follow ?last-batch-atom ?batch))
              (follow ?first-batch-atom ?batch-atom-in)))
     )
     ;; Special case for pipeline segment with a single
     ;; batch, "last" and "first" predicates refer to
     ;; the same batch atom
     (when (= ?first-batch-atom ?last-batch-atom)
           (last ?batch-atom-in ?pipe))
     (not (last ?last-batch-atom ?pipe))
     ;; Inserted batch-atom is removed from tank
     (not (on ?batch-atom-in ?from-tank))
     ;; Batch-atom removed from pipeline segment is inserted
     ;; into the destination tank
     (on ?last-batch-atom ?destination-tank)
     ;; Adjust both origin and destination tank volumes
     (increase (tank-current-volume ?destination-tank) 1)
     (decrease (tank-current-volume ?from-tank) 1))
)

(:action POP
   ;; This action is analogous to the Push action, the
   ;; difference
   ;; is in the pipeline segment movement direction. The
   ;; Pop action
   ;; moves the pipeline segment towards
   ;; the ‘‘to-area’’ defined in the ‘‘connect’’ predicate
   ;; ...
)
)
```

Pipesworld instances

This section presents six illustrative Pipesworld model instances, with solutions. Each instance explores a specific domain characteristic. The obtained results are commented in order to clarify the solution dynamics. Some model portions are omitted when the concept they represent is already well explained.

All the sample instances define a common set of products. There is a defined tankage for each product in each area. The tankage names follow the rule T[area-name][product-name]. For example, TA1OC1B is the tankage for product OC1B in A1.

The FF planning system (Hoffmann & Nebel 2001), with default parameters, was used to solve the sample instances. All the reported times are relative to the FF search phase execution on a 1 GHz Pentium III system, with 1GB of memory.

Reversion

This section presents an instance where a feasible plan must reverse the pipeline segment at least two times during the pipeline operation. Figure 3 shows the network configuration and the pipeline initial state. The goal is to move batch B_2 to A_2 and batch B_3 to A_1.

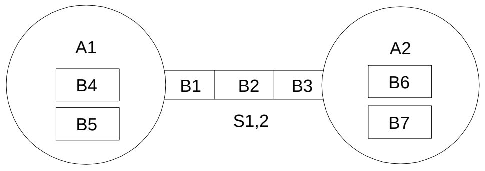

Figure 3: Reversion instance example

Instance definition in PDDL

```
;; Two areas, two reversion solution
;; Consider interface restriction
(define (problem reversion)
(:domain pipelines)
(:objects

;; Batches definition
B1 B2 B3 B4 B5 B6 B7 - batch-atom

;; Areas definition
A1 A2 - area

;; Pipeline segment definition
S12 - pipe

;; We will create one tankage per product, area
TA1LCO TA1GASOLEO TA1RAT-A TA1OCA1 TA1OC1B
TA2LCO TA2GASOLEO TA2RAT-A TA2OCA1 TA2OC1B - tank

)
(:init

;; Interfaces restrictions (partially represented)
(may-interface lco lco)
(may-interface gasoleo gasoleo)
;; ...

;; Network topology definition
(connect A1 A2 S12)

;; Specify tankage location (partially represented)
(tank-in-area TA1LCO A1)
(tank-in-area TA1GASOLEO A1)
;; ...

;; Specify tank maximum capacity (partially represented)
;; All tankage capacities are defined to 10
(= (tank-capacity TA1LCO) 10)
(= (tank-capacity TA1GASOLEO) 10)
;; ...

;; Specify tank product (partially represented)
(tank-product TA1LCO lco)
(tank-product TA1GASOLEO gasoleo)
;; ...

;; Batch-atoms products
(is-product B1 lco)
(is-product B2 gasoleo)
(is-product B3 rat-a)
(is-product B4 oc1b)
(is-product B5 oc1b)
(is-product B6 oca1)
(is-product B7 lco)

;; Specify tank current volume, 0 for all of them
;; (partially represented)
(= (tank-current-volume TA1LCO) 0)
(= (tank-current-volume TA1GASOLEO) 0)
;; ...

;; Batch-atoms initially located in areas
;; A1
(on B5 TA1OC1B)
(on B4 TA1OC1B)
;; A2
(on B6 TA2OCA1)
(on B7 TA2LCO)

;; Batch-atoms initially located in pipes
;; S12
(first B1 S12)
(last B3 S12)
(follow B2 B1)
(follow B3 B2)

)

(:goal (and
    (on B2 TA2GASOLEO)
    (on B3 TA1RAT-A)
))
)
```

FF was able to find a valid plan for the instance in less than a second, the plan is shown in Figure 4. The arguments after PUSH or POP are the pipeline segment number, the unitary batch name that is inserted into the segment and the destination tank, in that order. Since there is only one defined pipeline segment for this instance, the first argument is always $S_{1,2}$.

```
step    0: PUSH S12 B4 TA2RAT-A
        1: PUSH S12 B5 TA2GASOLEO
        2: POP S12 B3 TA1OC1B
        3: POP S12 B7 TA1OC1B
        4: POP S12 B6 TA1LCO
        5: POP S12 B2 TA1RAT-A
        6: PUSH S12 B1 TA2GASOLEO
```

Figure 4: Reversion instance solution

Scalability

Next, we conduct an experiment to check how the instance size may affect the problem complexity. We take the same instance and apply the following modification: each unitary batch B_n is replaced by two unitary batches, B_{n1} and B_{n2}.

The new instance accepts the same plan obtained for the original instance, with an appropriate transformation of the actions. For example, we may obtain a valid plan by replacing PUSH S12 B4 TA2RAT-A by PUSH S12 B41 TA2RAT-A; PUSH S12 B42 TA2RAT-A, and so on.

FF took five hours to solve this modified instance, a major increase on the time to solve the original instance. We have obtained similar results with the other proposed instances, which is a sign of the domain's complexity, even for small toy problems.

Interface

This next instance is obtained from the previous one, by adding an interface restriction that forbids the interface between LCO and OC1B. These are the products of B_1 and B_4, respectively. This additional restriction does not allow the previous plan first action.

FF was able to solve the new instance in approximately 10 seconds, giving as result the plan shown in Figure 5, which is four times longer than the previous plan, with seven reversions in the pipeline segment flow direction.

```
step    0: POP S12 B7 TA1LCO
        1: PUSH S12 B4 TA2LCO
        2: PUSH S12 B5 TA2RAT-A
        3: PUSH S12 B1 TA2GASOLEO
        4: POP S12 B6 TA1LCO
        5: POP S12 B7 TA1OC1B
        6: POP S12 B2 TA1OC1B
        7: POP S12 B3 TA1OCA1
        8: PUSH S12 B1 TA2RAT-A
        9: PUSH S12 B6 TA2GASOLEO
       10: PUSH S12 B4 TA2LCO
       11: POP S12 B2 TA1OC1B
       12: POP S12 B7 TA1OCA1
       13: POP S12 B3 TA1LCO
       14: PUSH S12 B4 TA2RAT-A
       15: PUSH S12 B1 TA2LCO
       16: PUSH S12 B6 TA2GASOLEO
       17: PUSH S12 B5 TA2OC1B
       18: POP S12 B3 TA1OC1B
       19: POP S12 B7 TA1OCA1
       20: POP S12 B2 TA1LCO
       21: POP S12 B4 TA1RAT-A
       22: PUSH S12 B6 TA2OC1B
       23: PUSH S12 B1 TA2GASOLEO
```

Figure 5: Interface instance solution

Tankage

This instance defines a new network, shown in Figure 6. For simplicity, only the LCO product is defined. The goal is to move B_2 to A_3.

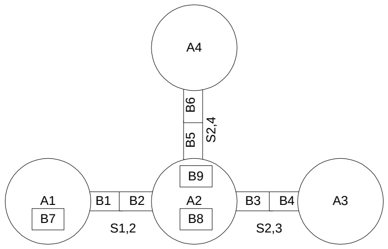

Figure 6: Instance with tight tankage restriction

If there is no tankage restriction, the solution is trivial and is shown in Figure 7.

If we limit the tankage for LCO at A_2 to 2, however, the plan must first free some space in A_2 in order to move B_2 from $S_{1,2}$ to A_2.

```
step    0: PUSH S12 B7 TA2LCO
        1: PUSH S23 B2 TA3LCO
        2: PUSH S23 B8 TA3LCO
        3: PUSH S23 B9 TA3LCO
```

Figure 7: Tankage instance trivial solution

The valid plan returned by FF for the new instance was found in less than five seconds, and is shown in Figure 8. The plan created space in A_2 by pumping products to $S_{2,4}$ and $S_{2,3}$, using this pipeline segment as "second level" storage for products.

```
step    0: PUSH S23 B8 TA3LCO
        1: PUSH S12 B7 TA2LCO
        2: PUSH S24 B9 TA4LCO
        3: POP S23 B4 TA2LCO
        4: PUSH S23 B2 TA3LCO
        5: POP S24 B6 TA2LCO
        6: PUSH S23 B8 TA3LCO
        7: PUSH S23 B9 TA3LCO
```

Figure 8: Tight tankage instance solution

Routing

Another important aspect of the pipeline transportation problem is routing. Figure 9 shows an instance where the batches may arrive at their destination following different routes. In this instance one must place B_8 in A_3. This can be accomplished by using three different paths: $P_1 = \{S_{1,4}, S_{4,3}\}$, $P_2 = \{S_{1,2}, S_{2,4}, S_{4,3}\}$ or $P_3 = \{S_{1,2}, S_{2,3}\}$. If we want a plan that minimizes the number of pump operations, P_1 is the best choice, since it requires only 4 operations, when P_2 and P_3 require 6 operations.

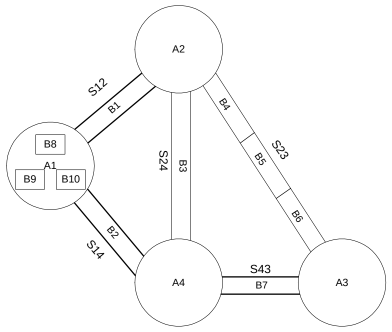

Figure 9: Instance with various routing options

FF was able to find P_1 in less than 12 seconds. The plan is shown in Figure 10.

```
step    0: PUSH S14 B8 TA4LCO
        1: PUSH S14 B9 TA4LCO
        2: PUSH S43 B8 TA3LCO
        3: PUSH S43 B2 TA3LCO
```

Figure 10: Routing instance solution

Cycling

An interesting variation of the previous instance is to remove B_9 and B_{10} from the model. It may seem at first that this makes the instance infeasible, but that is not the case.

The solution shown in Figure 11 was found by FF for this new instance, in less than 24 seconds:

```
step    0: PUSH S14 B8 TA4LCO
        1: POP S24 B2 TA2LCO
        2: POP S12 B3 TA1LCO
        3: PUSH S14 B1 TA4LCO
        4: PUSH S43 B8 TA3LCO
        5: POP S23 B7 TA2LCO
        6: PUSH S24 B4 TA4LCO
        7: PUSH S43 B2 TA3LCO
```

Figure 11: Cycling instance solution

We may observe that steps $\{0, 1, 2\}$ form a cycle in the batches flow, $C = \{S_{1,4}, S_{2,4}, S_{1,2}\}$. At the end of these steps, B_8 is positioned in $S_{1,4}$, and B_1 is in A_1. This batch is then used to move B_8 to A_4, in step 3. Step 4 move B_8 into $S_{4,3}$. A similar operation is then used to move B_2 to A_4, with this batch then being used to move B_8 to its final destination.

Planned extensions

We aim to extend the model to address real world situations. We also plan to study the performance of available general purpose solvers and to search for opportunities to derive new domain specific algorithms and control rules.

This section describes the planned improvements to the Pipesworld model, and the expected impact of the extensions.

Due dates for demands and productions

The current model goal fixes the final location of a group of unitary batches. In most real cases, however, a due date must be considered, both for demands and productions.

That is, there may be a demand that states that an unitary batch B_i must be at a location A_j before a deadline t_k.

The goal should then be changed from one that fixes the final batches location, as proposed, to one that guarantees that all demands are satisfied. This implies that the batch must be at its final destination in the desired time.

Flow rates

The inclusion of flow rate restrictions is challenging, since it imposes a range for the number of unitary batches pumped into a segment per unit of time. The use of unitary batches makes this kind of restriction very hard to model.

We plan to include flow rates restrictions into the model with numeric effects and durative actions. The numeric effects may be used to model the batches current volumes, thus removing the usage of unitary batches. Durative actions could be used to model operations with a constant flow rate.

Conclusions

We presented a new planning domain, Pipesworld, and showed with examples the unique characteristics of the domain. Although we have not made extensive tests with more than one general purpose solver, preliminary results indicate that this is a very challenging domain and will certainly need specific control rules to make the solution of real world sized instances feasible.

We intend to proceed our research in defining these control rules, and possibly deriving new algorithms to tackle this problem. We are specially interested on how network flow techniques (Ahuja, Magnanti, & Orlin 1993) may be used to construct the necessary control rules. Moreover, the domain will be extended to use temporal related restrictions, like due dates for demands and production and flow rate restrictions.

References

Ahuja, R.; Magnanti, T.; and Orlin, J. 1993. *Network Flows: Theory, Algorithms and Applications*. New Jersey: Prentice Hall.

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

Ghallab, M.; Howe, A.; Knoblock, C.; McDermott, D.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. Pddl—the planning domain definition language.

Hane, C. A., and Ratliff, H. D. 1995. Sequencing inputs to multi-commodity pipelines. *Annals of Operations Research* 57. Mathematics of Industrial Systems I.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253–302.

Milidiú, R. L.; Pessoa, A. A.; and Laber, E. S. 2002a. Complexity of makespan minimization for pipeline transportation of petroleum products. In *Proceedings of the APPROX'2002*, 243–255. Accepted for publication in TCSA.

Milidiú, R. L.; Pessoa, A. A.; and Laber, E. S. 2002b. Pipeline transportation of petroleum products with no due dates. In *Proceedings of the LATIN'2002*, 248–262.