Planning and Scheduling of Crude Oil Distribution in a Petroleum Plant

Tiago Stegun Vaquero[1,2] and **Fernando Sette**[1] and **José Reinaldo Silva**[1] and **J. Christopher Beck**[2]
[1] Department of Mechatronics, Universidade de São Paulo, Brazil
[2] Department of Mechanical & Industrial Engineering, University of Toronto, Canada
tiago.vaquero@poli.usp.br, fernando.sette@poli.usp.br, reinaldo@usp.br, jcb@mie.utoronto.ca

Abstract

Most real planning problems require intense knowledge management and reasoning about actions. These problems are challenging not only to designers during knowledge engineering but also to automated planners during the planning process itself. In this paper we present experience and results from designing a real planning application in the petroleum industry. We investigate the daily activities of a petroleum plant for docking, storing and distributing oil from a planning and scheduling perspective. Due to the complexity of this domain, the KE tool itSIMPLE was used to support the design process. The paper describes the construction of the domain model and the experimental results while testing the generated PDDL model with a modern planner. The experiments consider two semi-realistic scenarios in order to evaluate the approach.

Introduction

Over the last twenty years, an increasing interest in applying AI planning has led researchers and industry to investigate such technology in real world domain. In fact, the recent efficiency improvement of planning systems and the development of Knowledge Engineering (KE) tools have become a great motivation to investigate and experience the real design process of planning applications. From the challenges faced by researchers and experts in these applications new requirements and roadmaps emerge for the planning and scheduling community.

The main purpose of this work is to share the experience of designing and investigating a real application in the petroleum industry that can challenge both innovative KE tools and modern planning algorithms. The real problem presented in this paper deals with the planning of the daily activities of a petroleum plant including docking, storing, and distributing oil. These operations are very important to the functioning of refineries and they constitute a complex problem that is difficult to model mathematically (Dahal et al. 2003). When planning over this problem, engineers must deal with tanker allocation, docking scheduling, tank volume control, crude oil storage with price maximization (avoiding mixing certain types of crude oils) and the minimization of costs.

Due to the size and complexity of this real problem it is necessary to use tools that provide support for the design process, producing a input-ready model for planners. Designing the domain in PDDL (Fox and Long 2003) from scratch would have proven extremely difficult and time consuming. In this work we use the KE tool itSIMPLE (Vaquero et al. 2007) for the initial phases of the design including requirements acquisition, modeling, testing and plan analysis. In itSIMPLE's environment, the model is built using *Unified Modeling Language* (UML) (OMG 2005), a general purpose language broadly accepted in Software Engineering and Requirements Engineering. A PDDL model is automatically generated from the UML representation to be read by a chosen planner, in this case the SGPlan (Hsu et al. 2006).

The design process described in this paper extends our previous work on modeling this application (Sette et al. 2008) by developing a model that considers time constraints and introducing quality-metrics for plan analysis. In addition, two semi-realistic study cases are investigated in order to validate the resulting domain model.

This paper is structured as follows. First, we present the domain, its restrictions and requirements. Then we describe the design process, focusing on the modeling process using itSIMPLE. Next, we provide experimental results obtained by using SGPlan to solve two challenging planning problems. This paper ends with some conclusions.

Oil Supply as a Planning/Scheduling Problem

Operations with crude oil involve the unloading of tankers in docking stations into distribution tanks, and the supply of refineries through pipelines. Since the refineries are constantly consuming oil, the operations must guarantee that, at all moments, the amount of oil in the refineries remains above a minimum level, while minimizing the cost of distribution. Most research work done in this area has utilized mathematical programming in which the models are adapted to mixed-integer linear programming (MILP) or mixed-integer non-linear programming (MINLP) to find solutions to this problem. However, current methods have either failed to produce feasible solutions or required a great amount of time to solve these problems. Furthermore, MILP methods require the use of linearization, which leads to flaws in the final solutions, while the discretization necessary in MINLP methods greatly increases the size of the problem (Li et al.

2005). Therefore, as described in (Li et al. 2005), there is no reliable and robust algorithm for this real problem in current literature.

In this work we investigate the feasibility of an AI planning approach for a real oil supply problem encountered in one of the main distribution complexes of Brazil. The problem description and requirements used in this paper were based on the work of Mas and Pinto (2003), and the information provided by Petróleo Brasileiro S.A. (Petrobras), the main petroleum producer and distributor in Brazil.

In this problem, crude oil is processed in four refineries in the State of São Paulo (Brazil): Paulinia (REPLAN), São José dos Campos (REVAP), Cubatão (RPBC), and Capuava (RECAP). These refineries are supplied through a pipeline network that leaves the São Sebastião terminal (GEBAST). The system also contains two intermediate substations (SEBAT in Cubatão, and SEGUA in Guararema), as well as pumping stations in Rio Prado and Guaratuba. All the crude oil that is consumed by the State of São Paulo comes through GEBAST and is distributed by two pipelines: OSVAT and OSBAT. This system is detailed in Figure 1.

In this work we consider only the planning of three main operations performed daily in the São Sebastião terminal (GEBAST): docking of oil tankers; oil storage; and distribution of crude oil to refineries. These operations are performed in a distribution infrastructure that consists of a port, refineries and pipelines that carry the oil to the refineries. The port is composed of piers and tanks, along with internal pipelines that connects each pier to the tanks. This internal pipeline system has already been subject of study in the planning community, having appeared as a domain in the fourth International Planning Competition IPC'04. However, while the pipeline problem is operational in nature, this paper is concerned with a more strategic issue: the planning and scheduling of crude oil distribution in order to reduce costs and maximize profit.

The planning and scheduling of port operations involves several activities such as assignment of tankers to piers, unloading of the tankers to the tanks in the terminal, and unloading of the terminal tanks to the pipelines (Mas and Pinto 2003). The requirements associated to these activities are directly related to four main elements: tankers, tanks, pipelines and refinery. The main requirements for these elements are described below.

Tanker requirements: The crude oil arrives at GEBAST through oil tankers, which are unloaded at the docking stations and stored in the tanks of GEBAST. Each docking station has a limitation regarding the size of the tankers it can receive.

The unloading operation has to be done quickly and efficiently, since there are severe overstay costs in this process. Each tanker has a limited time that it can stay docked in the pier and unload oil without paying overstay costs. Therefore, the planning of this operation should respect this period whenever possible.

Finally, every tanker takes a certain time to dock and to leave the port. In practice, this means that, after the order to dock is given, a period must pass before unloading operations can begin. Moreover, a docking station is only able to receive another vessel a certain period after the exit order is issued to the tanker currently occupying it.

Tank requirements: Petrobras processes several different types of oil in its refineries. Since reserving a tank for each oil type is not practical, the oils are grouped into classes. The crude oil types that belong to the same class can be mixed together in a tank without losing value (Mas and Pinto 2003).

At a given moment, a tank can be in either one of three states: loading, inoperative, or unloading. Under no circumstance can a tank be unloading and loading simultaneously. Furthermore, there are some restrictions concerning the presence of brine in the tankers inventories. Since every oil type received from tankers at São Sebastião contains brine (even after separation in petroleum production platforms), the tanks must undergo a settling period (during which the tank remains inoperative) before they can send oil to the refineries. During this period (started after the last load process), the brine settles in the bottom of the tank. This is done in the tanks of GEBAST because it is not desirable to transport brine through the pipelines or send it to the refineries.

In order to prevent the accumulation of volatile components, the tanks operate using a floating roof system. Since a minimum safety level is required in order to avoid damage to these structures, the tanks can not be fully unloaded (Mas and Pinto 2003). This hard restriction is, usually, about two meters, which represents about 15% of the total capacity. Therefore, each tank has a maximum and minimum capacity that must be respected in the planning process.

Pipeline requirements: The pipelines are used to send oil from the terminal to the refineries that will process it. They are able to transport more than one crude oil type, sequentially allocated. During this transport operation, an interface forms between two different oil types resulting in a loss of their properties depending on their types (Mas and Pinto 2003). Petrobras uses a table that maps oil types and their interface costs. Moreover, a pipeline must not be used to unload distinct tanks simultaneously (i.e., one tank must be unloaded at a time)

Refinery requirements: The refineries have maximum and minimum capacity restrictions that must be respected throughout their operation. However, a complete model of the refinery operation will not be considered. Instead, it will be assumed, in the short term, that the refineries will have established an average rate of consumption of crude oil. Thus, the amount of oil required by the refinery to maintain its capacity restriction will be given in advance for the planning process. This oil amount is computed based on the refinery's consumption rate and the schedule horizon of tanker arrivals.

The Design Process with itSIMPLE

Since the planning application requires a careful design process that involves intensive knowledge acquisition and modeling, a Knowledge Engineering tool called itSIMPLE (Vaquero et al. 2007) was used to support the construction and

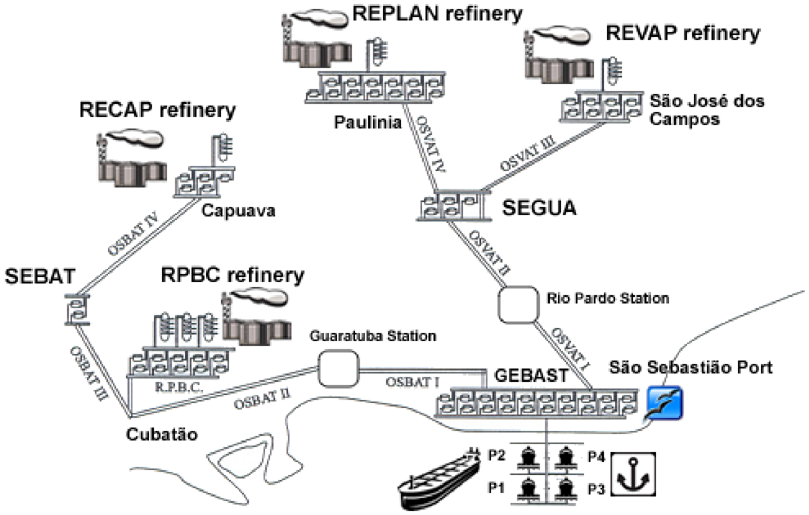


Figure 1: Crude oil distribution infrastructure of Petrobras in the State of São Paulo

development of a domain model. itSIMPLE's integrated environment focuses on the crucial initial phases of a design. The tool allows users to follow a disciplined design process to create knowledge intensive models of planning domains, from the informality of real world requirements to formal domain models that can be read by planners. The suggested design process for building planning domain models includes the following phases: requirements specification; modeling; model analysis; testing with planners; and plan evaluation (Vaquero et al. 2007). These phases are inherited from Software Engineering and Design Engineering, combined with real planning domain modeling experiences.

In this work we are going to focus on four of the main stages of such design process: requirements gathering, modeling, testing with planners, and plan analysis.

Gathering requirements

Requirements are gathered and represented using use case diagrams from UML. These diagrams model the domain in the highest abstraction level in which the scope is first defined. The diagrams usually facilitate the unification of the different viewpoints involved. The use case diagram for the São Sebastião terminal oil distribution activities is shown in Figure 2. In the diagram, each use case receives its description, pre- and post-condition, constraints, invariants, flow events and other relevant information.

As shown in Figure 2, the oil distribution system, which is centered at the terminal, possesses three independent agents (actors in UML): tanker, the terminal (port) itself, and the refinery. The actors interact to perform the tasks required to take the oil from the tankers and deliver it to the refineries.

Domain Modeling

Modeling in itSIMPLE follows a object-oriented approach using UML diagrams such as *class diagrams*, *state machine diagrams*, and *object diagrams*. The class diagram represents the static structure of the planning domain. It shows the existing entities, their relationships, their features, operators (actions) and constraints. A class's attributes and associations give a visual notion of the semantics of the model.

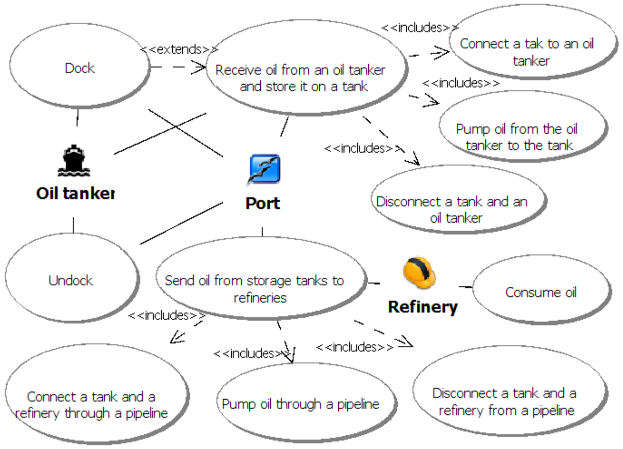
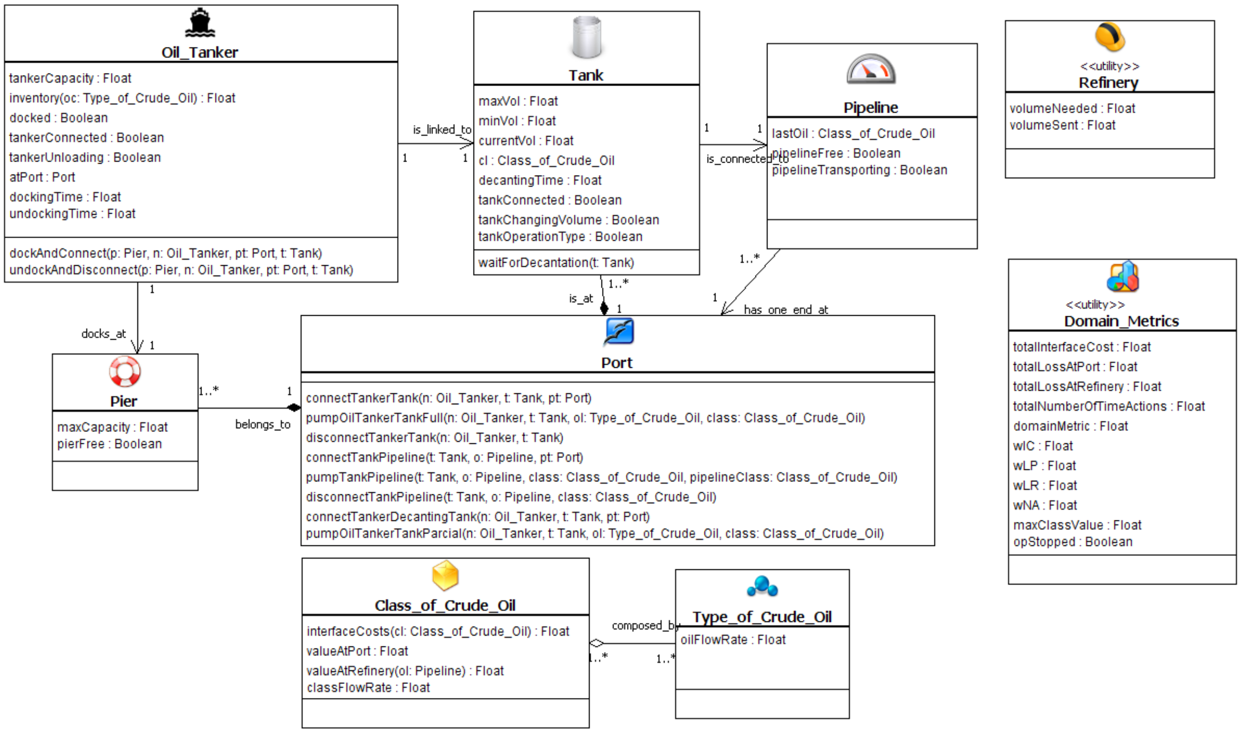

Dock
<<extends>>
Receive oil from an oil tanker and store it on a tank
<<includes>>
Connect a tak to an oil tanker
<<includes>>
Pump oil from the oil tanker to the tank
<<includes>>
Disconnect a tank and an oil tanker
Oil tanker
Port
Undock
Send oil from storage tanks to refineries
Refinery
Consume oil
<<includes>>
<<includes>>
<<includes>>
Connect a tank and a refinery through a pipeline
Pump oil through a pipeline
Disconnect a tank and a refinery from a pipeline

Figure 2: Use case diagram of the Oil Supply domain

Figure 3 shows the class diagram designed for the problem at the São Sebastião terminal. The diagram consists of nine classes: *Oil_Tanker*, *Port*, *Pier*, *Tank*, *Pipeline*, *Refinery*, *Type_of_Crude_Oil*, *Class_of_Crude_Oil*, and *Domain_Metrics*. These classes model all the entities relevant to the real problem.

The *Domain_Metrics* class is a utility class that stores variables that are relevant to all other classes in the model such as interface costs. In this particular case, these variables (corresponding to costs, revenue and time) are used as quality-metrics for the optimization of profit (minimizing losses and costs). The *Refinery* class controls the volume of oil that must be sent to itself (properties *volumeNeed*



Oil_Tanker
tankerCapacity : Float
inventory(oc: Type_of_Crude_Oil) : Float
docked : Boolean
tankerConnected : Boolean
tankerUnloading : Boolean
atPort : Port
dockingTime : Float
undockingTime : Float
dockAndConnect(p: Pier, n: Oil_Tanker, pt: Port, t: Tank)
undockAndDisconnect(p: Pier, n: Oil_Tanker, pt: Port, t: Tank)
is_linked_to
1
1
Tank
maxVol : Float
minVol : Float
currentVol : Float
cl : Class_of_Crude_Oil
decantingTime : Float
tankConnected : Boolean
tankChangingVolume : Boolean
tankOperationType : Boolean
waitForDecantation(t: Tank)
1
1
is_connected_to
Pipeline
lastOil : Class_of_Crude_Oil
pipelineFree : Boolean
pipelineTransporting : Boolean
<<utility>>
Refinery
volumeNeeded : Float
volumeSent : Float
1..*
1
has_one_end_at
1..*
is_at
1
1
docks_at
1
Pier
maxCapacity : Float
pierFree : Boolean
1..*
1
belongs_to
Port
connectTankerTank(n: Oil_Tanker, t: Tank, pt: Port)
pumpOilTankerTankFull(n: Oil_Tanker, t: Tank, ol: Type_of_Crude_Oil, class: Class_of_Crude_Oil)
disconnectTankerTank(n: Oil_Tanker, t: Tank)
connectTankPipeline(t: Tank, o: Pipeline, pt: Port)
pumpTankPipeline(t: Tank, o: Pipeline, class: Class_of_Crude_Oil, pipelineClass: Class_of_Crude_Oil)
disconnectTankPipeline(t: Tank, o: Pipeline, class: Class_of_Crude_Oil)
connectTankerDecantingTank(n: Oil_Tanker, t: Tank, pt: Port)
pumpOilTankerTankParcial(n: Oil_Tanker, t: Tank, ol: Type_of_Crude_Oil, class: Class_of_Crude_Oil)
<<utility>>
Domain_Metrics
totalInterfaceCost : Float
totalLossAtPort : Float
totalLossAtRefinery : Float
totalNumberOfTimeActions : Float
domainMetric : Float
wIC : Float
wLP : Float
wLR : Float
wNA : Float
maxClassValue : Float
opStopped : Boolean
Class_of_Crude_Oil
interfaceCosts(cl: Class_of_Crude_Oil) : Float
valueAtPort : Float
valueAtRefinery(ol: Pipeline) : Float
classFlowRate : Float
composed_by
1..*
1..*
Type_of_Crude_Oil
oilFlowRate : Float

Figure 3: Class diagram of the Oil Supply domain

and *volumeSent*). In fact, the refinery would need to be able to deal with a continuous variable regarding the volume of oil; however, the availability of general planners that handle such domain characteristics constrained the model in this direction. Thus, planners must considered a specific volume needed in the refinery to solve a problem.

The actions of the domain are modeled using two diagrams: the class diagram and the state machine diagram. In the class diagram we can define the name, the parameters and the duration of each operator (in this work we use discrete time). The dynamics of actions are specified in the state machine diagram, in which it is possible to represent the pre- and post-conditions of the operators declared in the class diagram. In the itSIMPLE, pre- and post-conditions are defined using the formal constraint language *Object Constraint Language* (OCL) (OMG 2003).

Usually every class in the class diagram has its own state machine diagram. One state machine diagram does not intend to specify all changes caused by an action. Instead, the diagram details only the changes that the action causes in an object of a specific class. Figure 4 shows the state machine diagram for the class *Tanker*.

In itSIMPLE, the UML object diagrams are used to describe the initial state and goal state of a planning problem. The object diagram represents a picture of the system at a specific state. It can also be seen as an instantiation of the domain structure defined in previous diagrams. This instantiation defines four main aspects: the number and type of objects in the problem; the values of the attributes of each object; and the relationships amongst the objects. In this

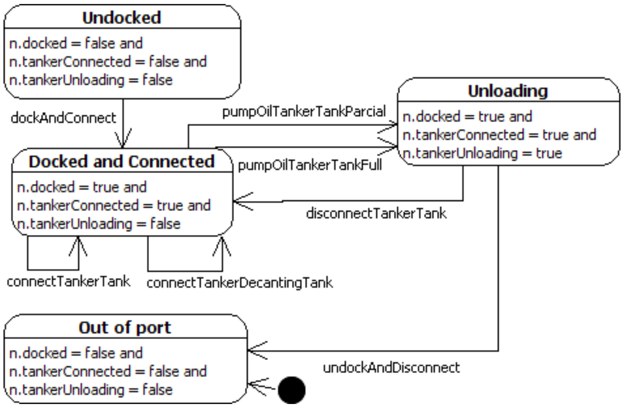
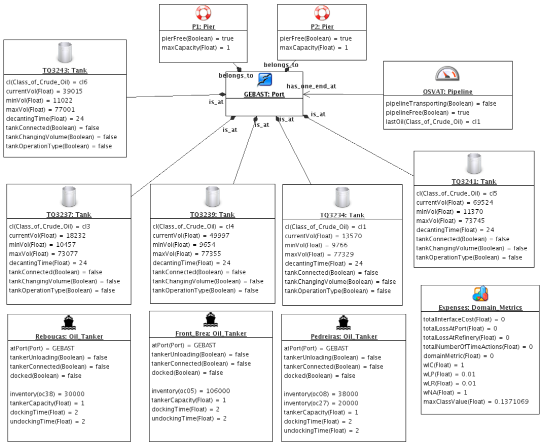


Figure 4: State machine diagram of the Tanker

case the initial state consist of all tankers, tanks, piers, oil types, and oil classes properly declared and defined. The goal state consists of all tankers unloaded (zero inventory) and undocked, while the volume sent to the refinery must satisfy the given volume needed (*volumeSent* >= *volumeNeed*).



P1: Pier
pierFree(Boolean) = true
maxCapacity(Float) = 1

P2: Pier
pierFree(Boolean) = true
maxCapacity(Float) = 1

GEBAST: Port

OSVAT: Pipeline
pipelineTransporting(Boolean) = false
pipelineFree(Boolean) = true
lastOil(Class_of_Crude_Oil) = cl1

TQ3243: Tank
cl(Class_of_Crude_Oil) = cl6
currentVol(Float) = 39015
minVol(Float) = 11022
maxVol(Float) = 77001
decantingTime(Float) = 24
tankConnected(Boolean) = false
tankChangingVolume(Boolean) = false
tankOperationType(Boolean) = false

TQ3237: Tank
cl(Class_of_Crude_Oil) = cl3
currentVol(Float) = 18232
minVol(Float) = 10457
maxVol(Float) = 73077
decantingTime(Float) = 24
tankConnected(Boolean) = false
tankChangingVolume(Boolean) = false
tankOperationType(Boolean) = false

TQ3239: Tank
cl(Class_of_Crude_Oil) = cl4
currentVol(Float) = 49997
minVol(Float) = 9654
maxVol(Float) = 77355
decantingTime(Float) = 24
tankConnected(Boolean) = false
tankChangingVolume(Boolean) = false
tankOperationType(Boolean) = false

TQ3234: Tank
cl(Class_of_Crude_Oil) = cl1
currentVol(Float) = 13570
minVol(Float) = 9766
maxVol(Float) = 77329
decantingTime(Float) = 24
tankConnected(Boolean) = false
tankChangingVolume(Boolean) = false
tankOperationType(Boolean) = false

TQ3241: Tank
cl(Class_of_Crude_Oil) = cl5
currentVol(Float) = 69524
minVol(Float) = 11370
maxVol(Float) = 73745
decantingTime(Float) = 24
tankConnected(Boolean) = false
tankChangingVolume(Boolean) = false
tankOperationType(Boolean) = false

Reboucas: Oil_Tanker
atPort(Port) = GEBAST
tankerUnloading(Boolean) = false
tankerConnected(Boolean) = false
docked(Boolean) = false

inventory(oc38) = 30000
tankerCapacity(Float) = 1
dockingTime(Float) = 2
undockingTime(Float) = 2

Front_Brea: Oil_Tanker
atPort(Port) = GEBAST
tankerUnloading(Boolean) = false
tankerConnected(Boolean) = false
docked(Boolean) = false

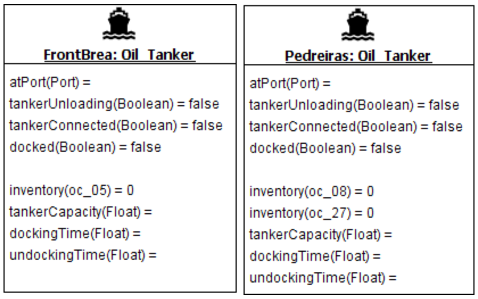
inventory(oc05) = 106000
tankerCapacity(Float) = 1
dockingTime(Float) = 2
undockingTime(Float) = 2

Pedreiras: Oil_Tanker
atPort(Port) = GEBAST
tankerUnloading(Boolean) = false
tankerConnected(Boolean) = false
docked(Boolean) = false

inventory(oc08) = 38000
inventory(oc27) = 20000
tankerCapacity(Float) = 1
dockingTime(Float) = 2
undockingTime(Float) = 2

Expenses: Domain_Metrics
totalInterfaceCost(Float) = 0
totalLossAtPort(Float) = 0
totalLossAtRefinery(Float) = 0
totalNumberOfTimeActions(Float) = 0
domainMetric(Float) = 0
wIC(Float) = 1
wLP(Float) = 0.01
wLR(Float) = 0.01
wNA(Float) = 1
maxClassValue(Float) = 0.1371069

Figure 5: Partial view of an initial state



FrontBrea: Oil_Tanker
atPort(Port) =
tankerUnloading(Boolean) = false
tankerConnected(Boolean) = false
docked(Boolean) = false

inventory(oc_05) = 0
tankerCapacity(Float) =
dockingTime(Float) =
undockingTime(Float) =

Pedreiras: Oil_Tanker
atPort(Port) =
tankerUnloading(Boolean) = false
tankerConnected(Boolean) = false
docked(Boolean) = false

inventory(oc_08) = 0
inventory(oc_27) = 0
tankerCapacity(Float) =
dockingTime(Float) =
undockingTime(Float) =

Figure 6: Goal state of a sample problem

Besides the object diagrams for defining initial and goal states, we also model the metric function to be optimized in every planning situation. A plan objective must considered four main aspects: (1) the cost of oil interfaces; (2) the profit generated by storing oil in the refinery ; (3) the profit resulting from storing oil at the port among the tanks in the terminal; and (4) the total time. In this application, the optimization of these aspects considers four domain variables, with their respective weights (*wLC*, *wLR*, *wLP*, and *wLA*), in a linear equation to be minimized during the planning process. These four variables, from the *Domain_Metrics* class, are the following:

- *totalInterfaceCost*: this variable holds the sum of interface costs during a plan. Every time oil is sent to the refinery through the pipeline the planner must check the interface cost between the last oil type sent and the one that is about to be sent.
- *totalLossAtRefinery*: since we want the best oil type at the refineries, this variable represents the losses related to sending low quality oil to the refineries. When a planner allocates oil to be sent to the refinery, this variable is incremented by the difference between the price of the highest quality oil (a fixed price) and the price of the oil that is about to be sent.
- *totalLossAtPort*: this variable represents the losses related to storing oil in tanks that belongs to a low quality class. Classes of oil must be properly mixed in order to maintain their high quality properties. Every time oil is mixed in a given tank, a price loss is computed based on the difference between the price of the class of oil in the tank and the price of the highest quality class of oil in the system. The variable is incremented with such price loss every time a tank is loaded with oil.
- *totalNumberOfTimeActions*: this variable measures the number of durative-actions as a simplified alternative to total time. However, the model can also use the reserved PDDL variable *totaltime* to compute the plan duration, assuming that the planner is able to deal with that variable.

Neither the cost of docking time of each tanker, nor the cost of overtime docking were considered in this work due to limitation on available general planners in dealing with continuous properties/time. The continuous approach could be used to compute the time that a tanker remains at the pier for its operations, providing the necessary costs to be considered during planning.

Model Testing with Planners and Plan Analysis

itSIMPLE can automatically generate a PDDL model from UML representation. Besides the automated translation process, the tool can communicate with several planners in order to test the domain models in an integrated design environment. In this application the planners must be selected based on the resulting PDDL model requirements that go beyond the classical approaches.

In order to analyze the generated plans, itSIMPLE provides two main support tools for plan analysis: plan simulation and plan validation. Plan simulation is performed by observing a sequence of snapshots (UML object diagrams), state by state, generated by applying the plan from the initial state to the goal state. The tool highlights every change in each state transition as described in (Vaquero et al. 2007).

For the plan analysis, itSIMPLE provides charts that represents the evolution of selected variables of the domain such as those that affect the quality of a plan (metrics).

Experimental Results

We selected the planner SGPlan (Hsu et al. 2006) for solving the planning problems of the oil supply application based on the minimal requirements of the domain model such as numeric properties, durative actions, and metrics. Among 10 planners available in itSIMPLE, SGPlan was the only one that could handle all model properties. In fact, we tested several planners using simplified problems of the domain and SGPlan had outstanding results (most of time only SGPlan was able to provide a solution), as described in (Sette et al.

2008). However, since SGPlan does not treat optimization functions we used the following approach for our application: in each experiment we run SGPlan multiple times and in each iteration we try to find a better solution by adding a request for lower value of the optimization function.

In order to evaluate the domain described in this paper and the generated plans from SGPlan, we created two case studies. These cases reflect real scenarios (with some simplifications) of the daily activities in the port of São Sebastião, São Paulo. The scenarios were based on and inspired by those presented and studied in (Mas and Pinto 2003). The first case investigates a simple situation and the second presents a more realistic scenario. In both scenarios we perform a continuous analysis of the plan produced by SGPlan, even though the presented model was defined with discrete actions.

Case Study 1

This case study represents a simplified scenario in the port; however, real data are used regarding the volumes of oil, types and classes of crude oil, tankers, tanks, costs and pipelines. The planning problem in this case contains one port that receives three tankers: *Reboucas*, *Front_Brea*, and *Pedreiras*. These tankers can be docked in two piers, *P1* and *P2*. The tankers are unloaded using five tanks in the port. The oil stored in the tanks must be sent (respecting the settling period) to a refinery through one pipeline.

The three tankers are loaded with crude oil that must be delivered at the port. The *Reboucas* tanker carries a type of crude oil called *oc38* while *Front_Brea* tanker carries an oil called *oc05*. The *Pedreiras* tanker carries two oil types, *oc08* and *oc27*.

The tanks are available to receive the crude oil. Each one of the tanks store a distinct class of oil. These classes must be considered to maintain the quality of the oil while mixing different types. When loaded, the tanks must remain inoperative for 24 hours waiting the brine to reach the bottom. Finally, the oil must be sent to the refinery, taking the interface cost into account, in order to accomplish the volume need. This problem is illustrated in Figure 7, along with the main oil flow.

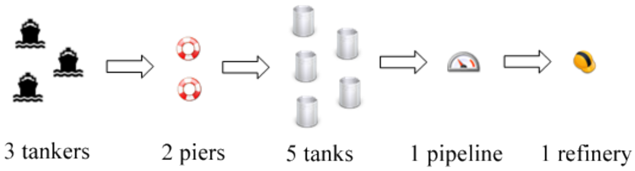


Figure 7: Case study 1 illustration

The scenario was sent to SGPlan which generated its best plan in the second iteration, as shown in Table 1. Figure 8 illustrates the plan.

In order to evaluate the solution, charts were used to check the changes of oil level in the tankers, tanks, and refinery in the continuous time. The charts in Figure 9 represents the evolution of oil levels (m^3) in some of these domain elements. This figure shows the viability of the plan in which all storage level constraints are respected. Due to the required volume (*volumeNeed*), it is possible to see that the refinery maintains its reserve at an adequate level.

	1st iteration	2nd iteration
time	0.17s	233.5s
Number of actions	27	24
Metric value	30.636	26.427

Table 1: Data of the solution for Case 1

```
0.001: (DOCK P2 FRONT_BREA GEBAST) [2.0000]
0.002: (CONNECTTANKPIPELINE TQ3241 OSVAT GEBAST) [0.0000]
0.003: (PUMPTANKPIPELINE TQ3241 OSVAT CL5 CL1) [13.0771]
0.004: (DOCK P1 PEDREIRAS GEBAST) [2.0000]
2.005: (CONNECTTANKERTANK PEDREIRAS TQ3239 GEBAST) [0.0000]
2.006: (PUMPOILTANKERTANKFULL PEDREIRAS TQ3239 OC27 CL4) [3.0874]
5.094: (DISCONNECTTANKERTANK PEDREIRAS TQ3239) [0.0000]
5.095: (CONNECTTANKERTANK PEDREIRAS TQ3237 GEBAST) [0.0000]
5.096: (PUMPOILTANKERTANKFULL PEDREIRAS TQ3237 OC08 CL3) [5.6205]
10.718: (DISCONNECTTANKERTANK PEDREIRAS TQ3237) [0.0000]
10.719: (UNDOCK P1 PEDREIRAS GEBAST) [2.0000]
12.720: (DOCK P1 REBOUCAS GEBAST) [2.0000]
13.090: (DISCONNECTTANKPIPELINE TQ3241 OSVAT CL5) [0.0000]
13.091: (CONNECTTANKERTANK FRONT_BREA TQ3241 GEBAST) [0.0000]
13.092: (PUMPOILTANKERTANKPARCIAL FRONT_BREA TQ3241 OC05 CL5) [8.5457]
14.724: (CONNECTTANKERTANK REBOUCAS TQ3243 GEBAST) [0.0000]
14.725: (PUMPOILTANKERTANKFULL REBOUCAS TQ3243 OC38 CL6) [4.2589]
18.985: (DISCONNECTTANKERTANK REBOUCAS TQ3243) [0.0000]
18.986: (UNDOCK P1 REBOUCAS GEBAST) [2.0000]
21.643: (DISCONNECTTANKERTANK FRONT_BREA TQ3241) [0.0000]
21.644: (CONNECTTANKERTANK FRONT_BREA TQ3234 GEBAST) [0.0000]
21.645: (PUMPOILTANKERTANKFULL FRONT_BREA TQ3234 OC05 CL1) [5.9768]
27.623: (DISCONNECTTANKERTANK FRONT_BREA TQ3234) [0.0000]
27.624: (UNDOCK P2 FRONT_BREA GEBAST) [2.0000]
```

Figure 8: The plan for case study 1

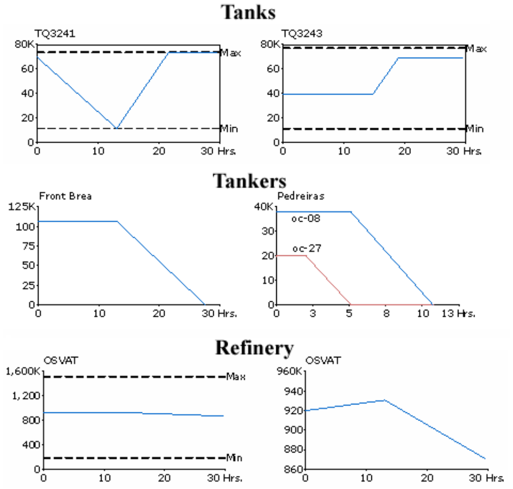


Figure 9: Oil levels evaluation

As mentioned, the current model does not considered the docking period cost and the cost of overtime docking. However, we have analyzed the solution based on these metrics. Figure 10 shows the period ordered for each tanker (48 hours, blue bars) and the de facto time used by tankers in the plan given by SGPlan. The figure shows that the unloading activities of tankers were perform efficiently (the docking time was not exceeded). It also shows that in this case, even with the discrete approach, we still achieved a valid solution concerning the docking costs, surely as a side effect of minimizing the timed actions. Thus, case study 1 shows a promising result for a realistic case.

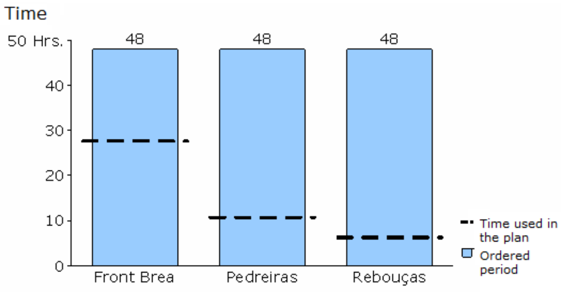


Figure 10: Time used vs. Ordered period of tanker operation

Case Study 2

This case aims to evaluate the model based on a realistic problem encountered daily in the São Sebastião port. In this scenario, seven tankers are considered: *Reboucas*, *Front_Brea*, *Pedreiras*, *Muriae*, *Vergina_II*, *North_Star*, and *Presidente*. These tankers can unload the oil into ten tanks (e.g. *TQ3243*, *TQ3237*, and *TQ3238*). For this problem, four piers are made available (*P1*, *P2*, *P3*, and *P4*). The delivered oil must supply a refinery considering the constraints on the tanks (24-hour inoperative period), the necessary volume, and also the quality-metrics. Figure 11 illustrates the new scenario.

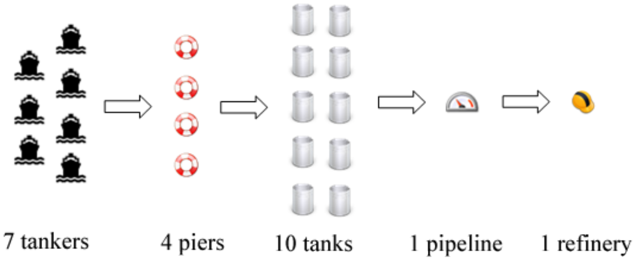
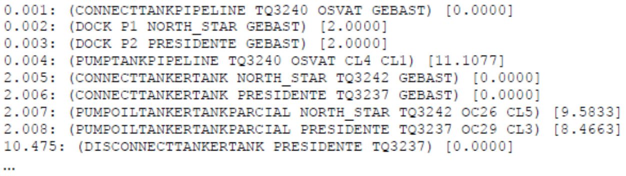


Figure 11: Case study 2 illustration

As in the previous case, the problem was automatically translated to a PDDL model that was then sent to the SGPlan. As opposed to case 1, SGPlan generated a valid plan in the first iteration, but it was unable to find a better solution in a second iteration. Table 2 shows general data about the plan generated by SGPlan. A partial view of the plan is shown in Figure 12.

	1st iteration
time	210.09s
Number of actions	88
Metric value	73.789

Table 2: Data of the solution for Case 2



```
0.001: (CONNECTTANKPIPELINE TQ3240 OSVAT GEBAST) [0.0000]
0.002: (DOCK P1 NORTH_STAR GEBAST) [2.0000]
0.003: (DOCK P2 PRESIDENTE GEBAST) [2.0000]
0.004: (PUMPTANKPIPELINE TQ3240 OSVAT CL4 CL1) [11.1077]
2.005: (CONNECTTANKERTANK NORTH_STAR TQ3242 GEBAST) [0.0000]
2.006: (CONNECTTANKERTANK PRESIDENTE TQ3237 GEBAST) [0.0000]
2.007: (PUMPOILTANKERTANKPARCIAL NORTH_STAR TQ3242 OC26 CL5) [9.5833]
2.008: (PUMPOILTANKERTANKPARCIAL PRESIDENTE TQ3237 OC29 CL3) [8.4663]
10.475: (DISCONNECTTANKERTANK PRESIDENTE TQ3237) [0.0000]
...
```

Figure 12: The plan for case study 2

In order to investigate the oil levels in tankers, tanks and the refinery, charts were also used to check the SGPlan's solution. Figure 13 illustrates some of the analyzed charts.

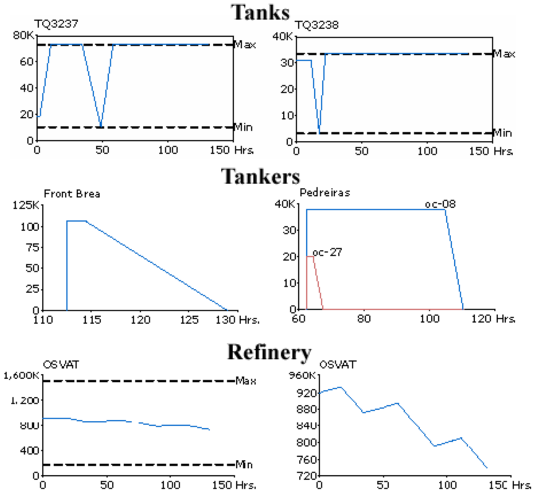


Figure 13: Oil levels evaluation in case 2

We also evaluated the solution based on the docking costs. Figure 14 shows the pre-established period for each tanker compared to the time used by the tankers in the generated plan. This figure emphasizes the efficiency of SGPlan's solution, i.e., most tankers operations remained below the established period. In fact, not all tankers finished their activities at the pier during the estimated 48-hour period; the *Presidente* tanker was the only one that had to remain docked overtime. This affects the final cost of the whole port operation; however, since the other tankers do not completely used the established period, a proper reduction of pre-established docking periods would decrease the total cost of docking operations.

Even with some restriction in the model concerning continuous time, the approach showed satisfactory results in two challenging planning problems. The domain discussed in this paper gathers features that usually challenge recent planners, such as time, large numbers of numeric resources, quality metrics and optimization. Moreover, these are some of features commonly found in real planning problems. As discussed in our previous work on this domain (Sette et al. 2008), few planners can handle domain models that combine all these features, but these applications give a clear roadmap for planning algorithm development.

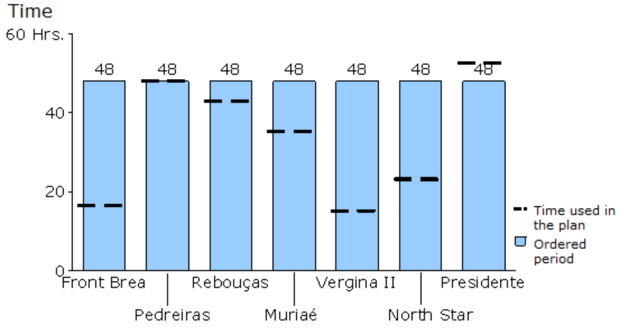


Figure 14: Time used vs. Ordered period of tanker operation in case 2

Conclusion

We have investigated a real planning problem, the planning/scheduling of the daily activities of a crude oil distribution plant, using an AI planning approach. We described the design process used for building a domain model with the KE tool itSIMPLE. As an extension of the work done in (Sette et al. 2008), we studied the domain model for the oil distribution activities in the São Sebastião port (São Paulo) considering time constraints and quality-metrics.

In order to validate the model in real scenarios, two case studies were tested using the SGPlan. The first one considers a semi-realistic scenario and the second brings a realistic case. The planner was chosen based on its capacity of dealing with the domain model requirements (durative-actions, numeric variables, and metrics). The metrics considered in these problems focus on the minimization of different parameters such as losses of mixing different oil types, interface losses in pipelines, and time spent for the activities. Experimental results showed that in both cases SGPlan was able to provide valid solutions. It is important to note that few planners can deal with such combination of PDDL features. Therefore, the resulting PDDL model brings interesting challenges even for the state-of-the-art planners. The model will be made available in order to share our results on this domain.

Experience from this application have motivated the improvement of itSIMPLE towards time-based models. As a future work, we will insert timed-based diagrams of UML into itSIMPLE in order to model continuous time. We plan to create an extended oil supply model with these new features in order to consider all costs.

References

Dahal, K.; Galloway, S.; Burt, G.; McDonald, J.; and Hopkins, I. 2003. Port system simulation facility with an optimization capability. *International Journal of Computational Intelligence and Applications* 3(4):395–410.

Fox, M., and Long, D. 2003. Pddl2.1: An extension of pddl for expressing temporal planning domains. *Journal of Artificial Intelligence Research (JAIR)* 20:61–124.

Hsu, C. W.; Wah, B. W.; Huang, R.; and Chen, Y. X. 2006. New features in sgplan for handling soft constraints and goal preferences in pddl3.0. In *Proc. Fifth International Planning Competition, International Conf. on Automated Planning and Scheduling*.

Li, J.; Wenkai, L.; Karimi, I. A.; and Srinivasan, R. 2005. Robust and efficient algorithm for optimizing crude oil operations. In *American Institute of Chemical Engineers Annual Meeting*.

Mas, R., and Pinto, J. M. 2003. A mixed-integer optimization strategy for oil supply in distribution complexes. *Optimization and Engineering* 4(1-2):23–64.

OMG. 2003. *UML 2.0 OCL Specification m Version 2.0*.

OMG. 2005. *OMG Unified Modeling Language Specification, m Version 2.0*.

Sette, F. M.; Vaquero, T. S.; Park, S. W.; and Silva, J. R. 2008. Are automated planners up to solve real problems? In *Proceedings of the 17th World Congress The International Federation of Automatic Control (IFAC'08), Seoul, Korea*, 15817–15824.

Vaquero, T. S.; Romero, V.; Tonidandel, F.; and Silva, J. R. 2007. itSIMPLE2.0: An integrated tool for designing planning environments. In *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS 2007). Providence, Rhode Island, USA.*