COPPE/UFRJ

MODELOS NÃO-LINEARES CONTÍNUOS PARA A LOGÍSTICA DE PETRÓLEO EM PORTOS E REFINARIAS

Fabio Dias Fagundez

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientadores: Adilson Elias Xavier
João Lauro Dorneles Facó

Rio de Janeiro
Agosto de 2010

MODELOS NÃO-LINEARES CONTÍNUOS PARA A LOGÍSTICA DE PETRÓLEO EM PORTOS E REFINARIAS

Fabio Dias Fagundez

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Adilson Elias Xavier, D.Sc.

Prof. João Lauro Dorneles Facó, Dr-Ing.

Prof. Márcia Helena Costa Fampa, D.Sc.

Prof. Felipe Maia Galvão França, Ph.D.

Prof. Marco Aurélio Cavalcanti Pacheco, Ph.D.

Prof. Eduardo Uchoa Barboza, D.Sc.

RIO DE JANEIRO, RJ – BRASIL

AGOSTO DE 2010

Fagundez, Fabio Dias

Modelos não-lineares contínuos para a logística de petróleo em portos e refinarias/Fabio Dias Fagundez. – Rio de Janeiro: UFRJ/COPPE, 2010.

XI, 179 p.: il.; 29, 7cm.

Orientadores: Adilson Elias Xavier

João Lauro Dorneles Facó

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2010.

Referências Bibliográficas: p. 74 – 84.

1. Scheduling. 2. Logística de Petróleo. 3. Programação Não-Linear. I. Xavier, Adilson Elias *et al.*. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Dedico este trabalho à minha esposa, Carla Hermann, que sempre esteve ao meu lado.

Agradecimentos

Agradeço aos meus orientadores, Professores João Lauro Facó e Adílson Xavier, que tornaram possível este trabalho com sua orientação conjunta e precisa ao longo destes quatro anos; ao Dr. Lincoln Moro (Petrobras) por sua colaboração com a minha pesquisa de doutorado; e aos Professores Felipe França, Priscila Lima e Luiz Fernando Legey, que me acolheram no Laboratório de Otimização Avançada (LOA), onde tive os recursos para desenvolver boa parte desta tese.

Agradeço aos Professores Eduardo Uchoa, Marco Aurélio Pacheco, Márcia Fampa e Felipe França, mais uma vez, pela participação na banca avaliadora.

Agradeço ao Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa em Engenharia (COPPE) da Universidade Federal do Rio de Janeiro (UFRJ) pela oportunidade de desenvolver o trabalho aqui apresentado, bem como ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), pela bolsa de pesquisa que me foi atribuída no período em que estive associado ao LOA.

Agradeço à minha esposa e à minha família, assim como aos meus amigos e colegas, que me ajudaram em diferentes momentos da pesquisa.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

MODELOS NÃO-LINEARES CONTÍNUOS PARA A LOGÍSTICA DE PETRÓLEO EM PORTOS E REFINARIAS

Fabio Dias Fagundez

Agosto/2010

Orientadores: Adilson Elias Xavier
João Lauro Dorneles Facó

Programa: Engenharia de Sistemas e Computação

Empresas da indústria de petróleo operam em um mercado altamente competitivo, sob pressão contínua por menores custos e processos mais eficientes. Nesta pesquisa de doutorado, abordamos problemas de *scheduling* de petróleo e derivados em portos e refinarias, o qual tem sido tratado com frequência sob a forma de modelos misto-inteiros lineares (MILP) e não-lineares (MINLP). Esta tese introduz uma abordagem nova e original de programação não-linear, cuja principal ideia é a de considerar a programação de produção (*schedule*) como um sistema dinâmico que deve ser operado sob certas restrições. Operações de transferência são executadas por vazões de equipamentos de origem para equipamentos de destino, as quais são mapeadas como variáveis de controle, enquanto os conteúdos dos equipamentos são mapeados como variáveis de estado. Decisões do tipo sim-não são modeladas com restrições de complementaridade, portanto possibilitando modelos contínuos não-lineares, equivalentes a modelos MIP (MILP ou MINLP, dependendo do conjunto de restrições adotado), de modo que um ponto viável no NLP possa ser mapeado diretamente a um ponto viável no modelo misto-inteiro. Essa abordagem é ilustrada com exemplos computacionais, obtendo ótimos locais.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

NONLINEAR CONTINUOUS MODELS FOR PETROLEUM LOGISTICS IN PORTS AND REFINERIES

Fabio Dias Fagundez

August/2010

Advisors: Adilson Elias Xavier
João Lauro Dorneles Facó

Department: Systems Engineering and Computer Science

Companies from the petroleum industry operate in a highly competitive market, under constant pressure for lower costs and efficient processes. In this doctorate research, we study the scheduling of crude oil and derivatives in ports and refineries, which has been frequently studied in the literature with mixed-integer models, both linear (MILP) and nonlinear (MINLP). The main idea presented in this thesis is that the schedule is a dynamic system which must operate under certain constraints. Transfer operations are carried out by flows from a source equipment to a destination equipment. Such flows are mapped as control variables, whereas equipment contents are mapped as state variables. Yes-No decisions are modeled with complementarity constraints, thus allowing a continuous nonlinear models, equivalents to MIP models (MILP or MINLP, depending on the chosen set of constraints), in such a way that a NLP-feasible point can be directly mapped to a MIP-feasible point. We illustrate this approach with computational examples, which were solved to local optimality.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Motivação

A indústria de petróleo se consolidou como a indústria mais relevante da economia mundial desde o início do século XX, dado que a principal fonte de energia utilizada em nossa sociedade é o petróleo. Empresas petrolíferas fazem parte de um ambiente de competição acirrada, regulado por normas ambientais cada vez mais restritivas, preços crescentes dos crus e margens reduzidas [1]. Em paralelo, um grande número de fusões e aquisições ocorreram nos últimos anos, consolidando a posição destas empresas como operadores de escala global, com necessidade de integrar refino, distribuição e exploração em diversos locais do mundo. Dentro deste panorama, empresas investem em novas tecnologias e processos como uma forma de tentar obter vantagens frente às concorrentes e se adequar às legislações vigentes. O planejamento corporativo procura otimizar os resultados da empresa como uma entidade integrada, coordenando a operação de cada uma de suas unidade de negócio harmoniosamente. Em sequência, cada unidade procurar otimizar sua operação individual, sujeita às restrições do plano corporativo. A disseminação de ferramentas de simulação, otimização e controle é, portanto, vital para a eficiência dessas empresas. Refinarias, por exemplo, utilizam sistemas comerciais de planejamento, controle avançado e otimização em linha, com diversos graus de maturidade e desempenho [2].

Dentre as atividades que requerem alto grau de coordenação entre seus executores, muitas vezes de empresas distintas, se encontra o transporte marítimo. Como

uma atividade milenar, que detém um quase-monopólio do transporte internacional de bens, o transporte marítimo cresce ano após ano, requerendo uma racionalização do uso de frotas e portos. O mercado de empresas de transporte, assim como o de petróleo, sofreu várias mudanças nos últimos anos, sob a forma de fusões e desregulamentação de mercados, forçando às empresas a se modernizarem [3]. Ademais, empresas de petróleo ou fazem contratos de longo prazo ou possuem empresas de transporte marítimo, de forma a garantir a distribuição de seus produtos (tanto crus quanto derivados) apesar das oscilações do mercado. As duas principais formas de transporte são a marítima e a dutoviária [4], sendo ambas campos férteis para a otimização em planejamento, *scheduling* e operação.

Portanto, dentre as possíveis áreas de interesse para uma pesquisa de doutorado, a otimização da logística de petróleo é uma área que permite o desenvolvimento de trabalhos que possam, efetivamente, trazer benefícios significativos. A programação logística do petróleo é um problema de ordem prática, de grande complexidade computacional, sendo objeto de estudo tanto na academia quanto na indústria. Diferentes escolhas de tanques para estocagem ou de sequências de processamento de petróleos podem gerar ganhos significativos, dado o alto custo operacional (aluguel de navios, consumo energético, particularmente para bombeamento em oleodutos), as multas contratuais envolvidas e as diferentes margens de acordo com a qualidade dos produtos [2]. A indústria procura, portanto, abordagens que permitam soluções de boa qualidade (ótimas ou sub-ótimas) em tempos computacionais aceitáveis, o que é um desafio, dado que o problema apresenta características combinatórias (alocação de atividades e equipamentos) e não-lineares (qualidades e composições resultantes de misturas de petróleos). Programações do dia-a-dia em refinarias e terminais marítimos são tipicamente obtidas de forma manual ou, em alguns casos, com o apoio de sistemas especialistas, simulação ou heurísticas [2]. O uso de modelos matemáticos de otimização se restringe, em geral, ao planejamento de longo prazo das companhias petrolíferas, onde grandes modelos econômicos agregados de Programação Linear são executados periodicamente (e.g. mensalmente) para determinar metas de produção, compra e venda das diferentes unidades da companhia [5]. Ao contrário do planejamento de longo prazo, o problema de curto prazo ainda carece de uma abordagem padrão que permita a sua otimização em tempo com-

putacional aceitável, sendo, portanto, um campo aberto para a pesquisa, conforme observa-se pelo grande número de trabalhos publicados no últimos 20 anos [6], [7]. Esta tese visa acrescentar ao universo de pesquisa uma nova opção a ser explorada.

Em resumo, a pesquisa aqui apresentada foi motivada pelas seguintes características do objeto de estudo:

- Complexidade: problemas de *scheduling* são classsificados como de complexidade NP-Completa ou NP-Difícil [8], portanto desafiadores para novos modelos e algoritmos de solução;
- Importância econômica: dados os altos custos operacionais e financeiros envolvidos em transporte, armazenagem, refino e distribuição de petróleo, pequenos ganhos percentuais obtidos por novas soluções podem ser financeiramente relevantes;
- Tempo de resposta para tomada de decisão: frente a mudanças ou situações inesperadas, um operador precisa, rapidamente, construir um novo plano (*schedule*), respeitando as restrições existentes;
- Ausência de metodologia ou ferramental consolidado: tanto na indústria, quanto na academia, há diferentes abordagens em estudo, de modo que este ainda é um campo propício para novas pesquisas.

1.2 Objetivos e Contribuição

Esta tese tem como objetivo apresentar uma nova abordagem baseada em modelos de otimização contínua não-linear, com restrições de complementaridade e sem variáveis binárias, para o planejamento de curto prazo (*scheduling*) da logística de petróleo, desde os navios-tanque até às unidades de destilação atmosférica (UDA), passando por tanques e dutos, visando a minimização dos custos operacionais, evitando multas por atrasos nos portos (*demurrage*) e respeitando restrições operacionais tais como concentrações máxima e mínima de componentes nas cargas das unidades, vazões máximas entre equipamentos e tempo de descanso nos tanques para separação de impurezas encontradas nos crus (e.g. salmoura).

O trabalho aqui apresentado foi construído a partir das seguintes etapas:

1. Estudo das alternativas existentes para o planejamento de curto prazo (*scheduling*) da logística de petróleo em portos e refinarias;

2. Desenvolvimento de modelos capazes de representar problemas de (*scheduling*) de forma apropriada; e

3. Testes com a modelagem proposta, realizando comparações com outras encontradas da literatura.

O estudo das alternativas existentes nos permitiu identificar que, apesar de métodos de otimização já serem empregados pela indústria do petróleo desde os anos 50, o problema de *scheduling* ainda não se encontra totalmente resolvido [6], estando, portanto, aberto à proposição de novas abordagens, como a que introduzimos nesta tese. A observação de que muitos dos modelos apresentados na literatura são misto-inteiros lineares (MILP), não representando o problema de forma fiel (fazem aproximações das misturas de petróleos) ou não podendo ser resolvidos em tempo computacional adequado (algumas horas para problemas de pequeno porte), nós motivou a propor uma nova abordagem empregando somente variáveis contínuas e tratando as misturas de petróleo de forma rigorosa, a qual, quando aplicada a problemas da literatura, resultou em soluções localmente ótimas em tempos computacionais adequados (segundos ou poucos minutos). Em particular, cabe ressaltar que os modelos encontrados na literatura, em geral, se utilizam de dois conjuntos de variáveis de decisão: (a) variáveis binárias, que indicam se um dado par de equipamentos foi escolhido ou não para uma dada atividade em um dado intervalo de tempo; (b) variáveis contínuas, que representam a vazão empregada ou o volume de petróleo (ou outro material) transferido entre um dado par de equipamentos em um dado intervalo de tempo. O nosso modelo não-linear emprega apenas o conjunto de variáveis contínuas (b), observando que, se uma vazão é positiva entre dois equipamentos, em um dado intervalo de tempo, houve uma decisão de empregá-los na atividade de transferência, enquanto se a vazão é zero, houve a decisão de não empregá-los. O uso de modelos não-lineares totalmente contínuos para problemas de *scheduling* de petróleo é uma inovação, abrindo um novo campo nesta área de pesquisa. A nossa ideia é que, a partir desta tese, outros trabalhos sigam, aprimorando os modelos e métodos de resolução empregados e combinando-os com outros

existentes. Uma primeira combinação possível, apresentada neste texto com exemplos computacionais, é o uso de modelos MILP e NLP de forma alternada, onde o resultado de um modelo é utilizado pelo outro.

1.3 Planejamento na cadeia de suprimentos da indústria de petróleo

O termo *scheduling* é utilizado para representar um processo de tomada de decisões dentro de indústrias de manufatura e produção, visando a alocação de recursos limitados a tarefas ao longo do tempo, a fim de otimizar um ou mais objetivos, tais como custo, inventário, lucro, tempo total de processamento, tempo de espera em filas, entre outros [9]. O *scheduling*, dentro da indústria de petróleo, representa o planejamento de curto prazo de tarefas relacionadas a transporte, processamento e armazenamento de petróleo e derivados dentro de sua cadeia logística. Em português, um termo também utilizado para *scheduling* é "programação de produção", em particular quando relacionado a plantas como refinarias ou petroquímicas [10].

O planejamento de uma empresa petrolífera é dividido em vários níveis, segundo a importância das decisões a serem tomadas, sendo o conceito de importância de planejamento medido conforme a sua abrangência. Quanto mais alto o nível decisório, maior o impacto da decisão sobre a empresa, maior o horizonte de tempo envolvido e menor o detalhamento da execução. O *scheduling* é um dos níveis hierárquicos de decisão mais baixos, lidando com tarefas de curto prazo e impactos locais. A Tabela 1.1 mostra uma classificação para os níveis de planejamento para a indústria de petróleo, adaptada a partir da classificação geral de níveis de planejamento apresentada em Morton e Pentico [9].

Maravelias e Sung [11] associam as responsabilidades de cada nível do planejamento da seguinte maneira: o plano estratégico (longo prazo) determina a estrutura da cadeia de suprimentos (e.g. localização de plantas), enquanto o tático (médio prazo) se concentra na determinação de metas de produção para cada membro da estrutura, bem como nos volumes que devem ser transportados para centros de distribuição. O planejamento de curto prazo (*scheduling*), por sua vez, é revisado diariamente ou semanalmente, para determinar as tarefas que devem ser associadas

Tabela 1.1: Classificação dos níveis de planejamento

Tipo	Plano	Horizonte
Longo prazo	Estratégico	2 a 5 anos
Médio prazo	Tático	2 a 24 meses
Scheduling	Roteiro de atividades	3 a 60 dias
Scheduling reativo	Controle de produção	1 a 3 dias

a cada unidade, bem como a sua sequência de execução. Dadas as interconexões entre os diferentes níveis de planejamento, é importante que os planos sejam integrados. No caso de Maravelias e Sung [11], eles propõem que o *scheduling* seja restrito formalmente pelas metas de médio prazo e que retroalimente as mesmas, indicando possíveis metas que precisariam ser corrigidas. Neiro e Pinto [12] também destacam a importância de um planejamento integrado, onde o plano corporativo mensal e o *scheduling* diário de cada refinaria são revisados (otimizados) simultaneamente, dentro de um *framework* comum que consolida informações de vários níveis (contratos locais e globais, estoques, disponibilidade de equipamentos, datas atualizadas de chegada e partida de navios, parcelas de petróleo em dutos, demandas por produtos, etc.). Entretanto, como o custo computacional para tal otimização integrada ainda é proibitivo, Neiro e Pinto [12] propõem uma abordagem de agregação e decomposição, onde o plano corporativo e os *schedules* são otimizados separadamente em rodadas, com os resultados de um problema alimentando o outro a cada rodada. Apesar de ser computacionalmente mais viável que o planejamento integrado, ainda assim o custo é bastante alto, de modo que, tal tipo de abordagem se restringe à academia. Na indústria, o *scheduling* tem de se adaptar ao que foi determinado no planejamento, dada a hierarquia do tomador da decisão, bem como os momentos em que elas são tomadas. O *planning* (planejamento de longo e médio prazos) geralmente é realizado por um grupo central da empresa, afetando-a como um todo, enquanto que o *scheduling* é feito localmente em cada unidade de execução da empresa [10].

Entre as decisões básicas de scheduling estão [13]:

- designar atividades a equipamentos: necessário quando há mais de um equipamento capaz de realizar uma mesma tarefa, em condições semelhantes ou não.

A decisão chave é descobrir qual o equipamento mais adequado para realizar a tarefa no cenário corrente, frente às outras opções;

- sequenciar a ordem de execução das atividades: em muito casos, é fundamental determinar a melhor sequência de atividades, uma após a outra, para se otimizar um dado objetivo. Isso ocorre, por exemplo, quando há gastos ou tempos diferentes na preparação (*setup*)dos equipamentos entre certas atividades ou entre certos materiais. Em um cenário deste tipo, é importante executar as atividades de forma que os *setups* sejam minimizados, pois diferentes ordens de execução poderão gerar diferentes valores para o objetivo que se queira otimizar;

- temporizar a utilização de recursos: para a montagem de um plano de execução que possa ser utilizado pela equipe que operará os equipamentos da unidade da empresa, deve-se atribuir o instante exato de início e término de cada atividade em cada equipamento, com o devido consumo de recursos ou produção de produtos, bem como as condições operacionais aplicáveis. A partir deste plano, é possível prever-se a variação de estoques, o uso de recursos e se os contratos vigentes serão atendidos dentro dos prazos acordados. Este plano também informa quando e quais equipamentos estariam ociosos, o que pode ser utilizado para a programação da manutenção dos mesmos.

Em resumo, o *scheduling* é uma das áreas chaves de operação de processos. Em geral, lida com a designação de recursos ao longo do tempo para a execução de um conjunto de atividades. No contexto da indústria de petróleo, *scheduling* se refere a estratégias de designação de equipamentos, matérias-primas, utilidades e mão-de-obra para a armazenamento, distribuição ou processamento de um ou mais produtos, através de decisões referentes a designar, ordenar e temporizar tarefas, sempre retritas pelo planejamento mais geral da companhia.

Na literatura, problemas de scheduling são usualmente modelados como problemas discretos (ou misto-inteiros), e podem ser classificados de acordo com configuração de máquinas, uso de recursos e etapas produtivas, presença de tempos de troca etc. Duas classes bastante conhecidas são as de problemas de *job shop* - onde um produto é montado a partir de uma sequência ordenada de tarefas em equipa-

mentos - e de *flow shop* - onde equipamentos podem participar de diferentes estágios de produção de diferentes produtos [9]. Essa classificação, entretanto, geralmente é aplicada à indústria de manufatura discreta, e não consegue representar em detalhes as particularidades da indústria de processos (petroquímicos, químicos e refino) contínuos e em bateladas. Dentro da indústria de processos, destacam-se duas representações, inicialmente somente para processos em batelada, mas depois expandida também para contínuos: STN (State Task Network) [14] e RTN (Resource Task Network) [15], com possíveis derivações (por exemplo, ver [16], [17], [18]).

Na representação STN, os estados (materiais) e as tarefas são representados como nós em uma rede, conectados por arcos se uma tarefa consome ou produz algum estado. Unidades de processo e utilidades, entretanto, são tratados de forma implícita: para cada unidade há um conjunto de tarefas específicas que ela pode executar, bem como, para cada utilidade, há um conjunto específico de tarefas que a consomem. Tanques são dedicados a são associados a materiais, podendo o modelo STN poder ser estendido para tanques compartilhados. Produção e consumo de materiais pelas tarefas são restritos por equações de balanço material ou volumétrico, associadas às vazões nos arcos da rede. Restrições específicas são usadas para restringir as alocações entre unidades e tarefas, bem como os níveis de consumo de utilidades. A representação RTN é uma evolução da STN, explicitando as relações entre tarefa-unidade, tarefa-utilidade e tanque-estado, tratando materiais, unidades de processo e utilidades como recursos consumidos (produzidos) por uma tarefa ao seu início (final). Balanços, alocações de tarefas e consumo de utilidades são expressados como restrições nas vazões da rede. O conceito de tarefa é central a ambas as representações, de modo que quaisquer modificações nas alocações de equipamentos e mudanças de inventário são percebidas ou medidas em inícios ou finais de atividades. Uma solução é viável se todas as restrições são respeitadas nos instantes determinados pelos inícios e finais de atividades. Além disso, nestas representações, não há alterações de estado nos tanques sem a ocorrência de tarefas.

1.4 *Scheduling* de petróleo

A logística de petróleo e derivados é uma tarefa complexa, pois lida com características eminentemente não-lineares, oriundas das misturas de petróleo, e combinatórias, resultante das diversas combinações de origens e destinos para cada parcela de petróleo ou produto. O problema considera equipamentos diversos, tais como tanques, dutos, navios-tanque, pieres de atracação em terminais e portos, todos submetidos a diferentes restrições operacionais [19], [4].

A construção de plano de curta duração (*schedule*), pode ser obtida a partir do reconhecimento de que o sistema logístico é um sistema dinâmico, onde cada ação em um dado instante gera um impacto nos estados futuros do sistema. As ações que podem ser controladas pelo responsável pela programação (*scheduler* ou programador) são as variáveis de controle, enquanto as características do sistema, que podem ser medidas ou calculadas (estimadas), são as variáveis de estado. É possível simular diferentes cenários para o *scheduling* (recebimento, transporte, distribuição, processamento e armazenamento) de petróleo apenas modificando as diferentes vazões entre os equipamentos do sistema logístico: quando um navio atraca, qual tanque deve ser escolhido para receber a carga do navio? Qual a melhor sequência de crus para alimentar uma dada unidade de destilação atmosférica? Essas são perguntas que o *scheduler* tenta responder diariamente, por meio da construção de cenários comparativos. Em cada cenário, diferentes decisões são tomadas, que podem ser refletir em todas os níveis da cadeia de suprimentos.

Como gerar diversos cenários e compará-los manualmente é muito custoso e demorado, a indústria vem procurando soluções automatizadas de *scheduling*, ainda sem um sucesso incontestável [6]. Por ser um problema em aberto, a academia também o estuda, apresentando nos últimos anos diferentes tipos de abordagens. Em especial, uma abordagem bastante encontrada na literatura recente é o uso de modelos de otimização misto-inteira [7], onde se apresentam dois tipos de restrições: as discretas (inteiras) e as contínuas. As do primeiro tipo representam enumerações ou decisões lógicas, como "Escolha tanque A para receber o petróleo P do navio B no instante t", enquanto as do segundo tipo representam limitações mais gerais como "A capacidade máxima de armazenamento do tanque A é 30.000 m^3". Restrições em variáveis discretas regem decisões de alocação e sequenciamento, enquanto as

restrições em variáveis contínuas regem os balanços de massa, energia ou de componentes.

Floudas e Lin, em seu *survey* [7], destacam que a Programação Misto-Inteira Linear (MILP) tem sido bastante utilizada na academia para a modelagem de problemas de *scheduling* de petróleo. A garantia de que existe um ótimo global é considerada a grande vantagem desta abordagem. Entretanto, dada a complexidade do problema, estes modelos sofrem da "maldição da dimensionalidade": quando aplicados a instâncias próximas da realidade, os métodos de otimização consomem memória em demasia ou resultam em tempos computacionais inviáveis, muitas vezes precisando ser abortados antes de convergirem para uma solução aceitável. Uma possível forma de reduzir o tempo computacional é utilizando heurísticas desenvolvidas sob medida, que se aproveitam de estruturas particulares de cada problema, permitindo a obtenção de boas soluções em tempos computacionais viáveis [20], [21]. Outra abordagem é substituir a função-objetivo do *scheduling* por funções alternativas, com melhores propriedades matemáticas [22]. Adicionalmente, a comunidade científica tem trabalhado continuamente em formulações que reduzam as dimensões dos modelos, em particular com formulações de tempo discretizado em intervalos não-uniformes, também chamado de "tempo contínuo"(veja [23] para um levantamento detalhado sobre o assunto).

Fenômenos não-lineares relacionados a operações unitárias e misturas de componentes, para serem modelados de forma minimamente rigorosa, incorrem em modelos recheados de variáveis bilineares, portanto, não-convexos. Por exemplo, o cálculo mais simples de propriedades ou concentrações resultantes em uma mistura é a soma da propriedade de cada componente ponderada pela fração (mássica ou volumétrica) do componente na mistura, uma formulação não-convexa [24]. Outras propriedades (e.g. viscosidade), cujas fórmulas de misturas seriam mais complexas, podem ser convertidas em índices [25], os quais podem ser calculados em uma mistura por meio da soma ponderada dos índices pelas frações. Mesmo misturas tratadas com a fórmula mais simples não podem ser modeladas corretamente em um modelo MILP, sendo então, tratadas por meio de aproximações ou relaxações lineares, ou ainda, removidas do modelo.

A principal ideia apresentada nesta tese é que é possível utilizar modelos total-

mente contínuos, sem qualquer variável discreta, portanto com um menor número total de variáveis. Esses modelos lidam com as decisões de alocação e sequenciamento por meio de restrições de complementaridade aplicadas às variáveis contínuas, sendo, portanto, modelos não-lineares (NLP) não-convexos com soluções ótimas locais. Em compensação, as misturas de petróleo podem ser modeladas de forma rigorosa, sem a necessidade de aproximações lineares. Cabe ressaltar que um ponto viável no modelo NLP é equivalente a um ponto viável em um modelo MILP e vice-versa. Assim, uma solução obtida no NLP é equivalente a uma solução integral do MILP, portanto definindo um limite superior (se resolvendo um problema de minimização) para o MILP. Desta forma, ainda que seja interessante manter a formulação MILP, o NLP pode ser resolvido em paralelo, gerando cortes superiores que podem ser utilizados na resolução do MILP. No caso de um método de *Branch and Bound*, uma solução NLP pode ser utilizada para descartar ramos da árvore de B & B, reduzindo o tempo computacional total para se encontrar o ótimo global.

Adicionalmente à abordagem por complementaridade, ao considerarmos o sistema logístico como um sistema dinâmico, o mesmo pode ser modelado como um problema de controle ótimo. Em indústrias com a de Óleo e Gás, sistemas de controle são construído a partir de modelos dinâmicos de controle ótimo, onde procura-se garantir uma operação contínua e segura, dentro de limites operacionais. Um problema de controle ótimo é equivalente a um problema não-linear [26], apresentando uma Jacobiana com estrutura diagonal em blocos, fortemente separável, o que permite a rápida convergência de métodos comuns de otimização não-linear [27]. Muitos pacotes de otimização (*solvers*) podem se aproveitar desta estrutura separável, diminuindo o custo computacional para a obtenção de um ótimo local.

1.5 Publicações derivadas da tese

A pesquisa desenvolvida nesta tese permitiu a publicação de trabalhos em congressos ou periódicos, conforme listados a seguir:

- Artigo publicado em periódico [28]: FAGUNDEZ, F. D.; XAVIER, A. E.; FACÓ, J. L. D. "Continuous Nonlinear Programming Techniques to Solve Scheduling Problems". Informatica (Vilnius), 2009, v. 20, p. 203-216.

- Capítulo de livro [29]: FAGUNDEZ, F. D.; FACÓ, J. L. D.; XAVIER, A. E. "A Nonlinear Optimal Control Approach to Process Scheduling". In: A. Chinchuluun; P.M. Pardalos; R. Enkhbat; I. Tseveendorj. (Org.). Optimization and Optimal Control: Theory and Applications (Series in Springer Optimization and Its Application), Springer, 2010, v. 39, p. 409-422.

- Artigo submetido à periódico [30]: FAGUNDEZ, F. D.; FACÓ, J. L. D.; XAVIER, A. E. "A novel nonlinear optimal control approach for crude oil and derivatives scheduling". Journal of Scheduling, Springer. (submetido em 2010)

- Artigo publicado em anais de congresso (FOCAPO 2008): FAGUNDEZ, F. D.; XAVIER, A. E.; FACÓ, J. L. D. "A hybrid NLP-MILP scheme for scheduling problems". In: Fifth International Conference on Foundations of Computer-Aided Operations - FOCAPO 2008, 2008, Cambridge. Fifth International Conference on Foundations of Computer-Aided Operations, 2008. v. 1. p. 417-420.

- Outros artigos ou resumos publicados nos seguintes anais de congresso:

  - European Conference on Operational Research (EURO 2010, 2009 e 2007);
  - 8th EUROPT Workshop "Advances on Continuous Optimization"(EUROPT 2010);
  - ALIO-INFORMS Joint International Meeting (ALIO-INFORMS 2010);
  - Congresso Ibero-Latino-Americano de Métodos Computacionais em Engenharia (CILAMCE 2009);
  - International Symposium on Mathematical Programming (ISMP 2009 e 2006);
  - International Conference "Approximation and Optimization in the Caribbean"(APPOPT 2008);
  - EURO Mini-Conference "Continuous Optimization and Knowledge-Based Technologies"(EUROPT 2008);
  - International Conference on Operations Research (ICOR 2008);

- International Conference on Engineering Optimization (ENGOPT 2008);
- International Workshop on Harbor, Maritime & Multimodal Logistics Modelling and Simulation (HMS 2008);
- Operational Research Peripatetic Postgraduate Programme (ORP3 2007);
- Multidisciplinary International Scheduling Conference: Theory & Applications (MISTA 2007);
- Simpósio de Pesquisa Operacional e Logística da Marinha (SPOLM 2007);
- American Institute of Chemical Engineers Annual Meeting (AIChE 2007).

1.6 Estrutura do texto

Esta tese continua em mais quatro capítulos, adicionais à introdução apresentada no Capítulo 1, os quais são:

- Capítulo 2: onde uma revisão da literatura recente no assunto é apresentada;
- Capítulo 3: onde propomos uma modelagem não-linear contínua para o problema e discutimos sua implementação;
- Capítulo 4: onde apresentamos resultados computacionais obtidos com a modelagem proposta;
- Capítulo 5: onde consolidamos nossas conclusões e fazemos sugestões para pesquisas futuras, a partir da abordagem introduzida neste trabalho.

Capítulo 2

Scheduling de petróleo em portos e refinarias

2.1 Descrição do problema

Operações de logística desempenham um papel importante na indústria de petróleo, em particular os transportes por dutos e marítimo. O transporte por dutos é responsável pela quase totalidade do transporte continental de fluidos, sendo frequentemente utilizado por empresas de Óleo e Gás, petroquímicas, farmacêuticas, dentre outras. Stephenson [32], em 1987, ressaltou que aproximadamente 25% de todo o transporte de materiais entre cidades dos EUA já era feito por meio de dutos na época. O transporte marítimo, por sua vez, é detentor de quase um monopólio no transporte entre continentes [3]. Companhias de transporte de carga mantêm o máximo de sua capacidade ocupada, evitando que seus navios fiquem parados, vazios, em portos, enquanto os contratantes procuram garantias de que os navios alugados irão atender suas necessidades dentro dos prazos necessários. Um dos maiores gargalos logísticos, no entanto, é o tratamento dos navios em portos: um navio de atracar, ser carregado ou descarregado e desatracar dentro de uma janela contratual. Se algum atraso ocorrer, são multadas as companhias detentoras das cargas ou que alugaram os navios (multas por sobreestadia, *demurrage*). No Brasil, os custos de transporte marítimo atingiram USD 7 bilhões em 2006, sendo USD 1,5 bilhões gastos somente com pagamento de multas por sobreestadia em portos [31].

Dentre vários problemas de *scheduling* na indústria de petróleo, trabalharemos

com o problema de distribuição de crus no sistema logístico formado por portos ou terminais marítimos, centro de distribuição e refinarias, considerando navios-tanque, pieres, tanques, dutos e unidades de destilação atmosférica.

Em linhas gerais, o *scheduling* de petróleo é definido como o problema de determinar: (a) ordem de atracação dos navios nos píeres; (b) bombeio de cargas (operações de transferência) entre navios e tanques, tanques e dutos e tanques e unidades de destilação; e (c) sequência de crus transportados nos dutos ou processados nas unidades de destilação, de forma otimizar uma função-objetivo, sujeita a restrições operacionais, físico-químicas e econômicas. Como as dificuldades desse problema são grandes, em muitas indústrias o objetivo é tão somente a obtenção de *schedules* viáveis [19].

O sistema logístico associado ao problema pode ser dividido em três subsistemas complementares [4]: porto ou terminal marítimo, centros de distribuição e planta (refinaria), todos conectados por meio de dutos, conforme ilustrado na Figura 2.1. Em muitos casos, a refinaria se comunica diretamente com o terminal marítimo, sem o subsistema de distribuição. No Brasil temos exemplos com os centros de distribuição, como a infraestrutura do estado de SP [4], e exemplos com refinarias ligadas ao terminal marítimo por meio de dutos, como a REFAP no estado do RS [33]. É possível, ainda, considerar-se um quarto subsistema, a frota de navios, com suas cargas, características físicas e datas estimada de chegada aos portos (ETA - Estimated Time of Arrival) e máxima de estadia, firmadas em contrato.

A tancagem do porto funciona como um *buffer* de segurança para o suprimento de petróleo: mantém estoque suficiente para atender sem interrupções a demanda de cru para refino em caso de atrasos nas chegadas dos navios. Assim, a programação do porto precisa atender a demanda das refinarias e manter a operação de oleodutos com o menor número possível de interrupções. Já no caso dos produtos finais, os estoques são bem mais baixos, por razões como a natureza do produto ou como o fato de serem necessários produtos apenas quando houver contratos de entrega, atrelados a navios específicos [34]. Os tanques no porto são segregados em relação ao tipo de produto estocado, ou seja, há tanques para diesel, gasolina, várias classes de cru etc. e um tanque de um tipo de material não deve ser utilizado para armazenar outros tipos de materiais (e.g. tanques de diesel não podem armazenar crus). Os

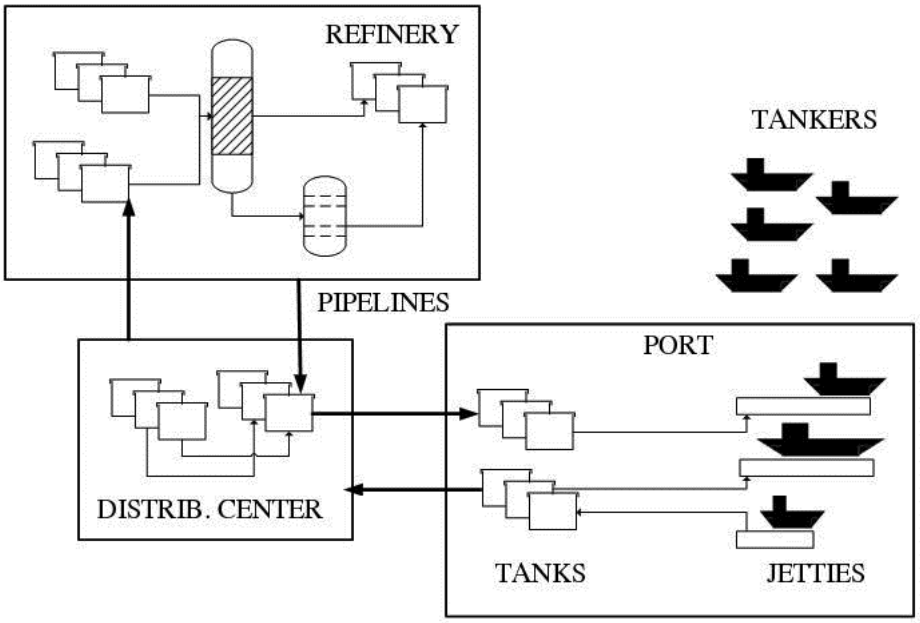

Figura 2.1: Sistema Logístico

tanques podem ser de terceiros ou da própria empresa de petróleo. Dessa forma, pode-se considerar os custos de tancagem no *scheduling*, apesar de ser um objetivo menos importante, se comparado com a garantia da correta operação dos dutos e com o atendimento dos navios sem atrasos. Todos os tanques têm uma capacidade de armazenamento máxima e mínima (maior que zero em caso de tanques com teto flutuante).

A correta operação de dutos é fundamental: eles são os meios de conexão entre os portos e as refinarias (e centros de distribuição), devendo operar constantemente, sem interrupções, sob a pena de gerar prejuízos nas refinarias que se programaram para o recebimento de certos crus. Garantir essa operação é um dos maiores objetivos do *scheduling*, ao lado do correto atendimento dos navios. As parcelas de materiais transportados devem obedecer as demandas e restrições operacionais. Além disso, idealmente deve-se escolher uma sequência de parcelas de crus com qualidades semelhantes, diminuindo perdas na interface entre um par de parcelas dentro do duto.

Os navios são contratados para entrega ou recebimento de cargas nos portos, cumprindo contratos que prevêem altas multas por atrasos. Alguns navios são multi-compartimentalizados (podendo ter cargas diferentes), enquanto outros apre-

sentam apenas um compartimento (carga única). Em geral, a estimativa da data de chegada (ETA) é incerta, e varia quase que diariamente, de acordo com as condições marítimas, atrasos em outros portos, problemas mecânicos etc. Assim, a programação precisa ser revista diariamente, considerando a ETA mais atual como entrada no modelo [35]. Uma forma de evitar revisões frequentes, é utilizar técnicas de otimização que considerem incertezas nos parâmetros de entrada (usando programação paramétrica ou cenários estocásticos, por exemplo) na construção de um *schedule* mais robusto [36]. Entretanto, considerando que o tempo computacional para resolver um problema com incertezas é normalmente maior que quando sem incertezas, e que esta ainda é uma área em que nem mesmo existe um consenso em que métricas utilizar numa avaliação de robustez [37], vamos abordar o problema considerando que nossos dados de entrada são confiáveis. Atrasos na liberação do navio acarretam em multas diárias (de sobrestadia ou *demurrage*) e devem ser evitados ao máximo. Segundo Collyer [31], o Brasil gastou em 2006 mais de USD 8 bilhões em afretamento de navios, e a incrível cifra de USD 1,5 bilhão em multas por sobreestadia. Assim, deve-se considerar que o objetivo mais importante no *scheduling* em portos é garantir que a companhia contratante (no nosso caso, a de petróleo) consiga ter todos os seus navios atendidos em tempo, sem incorrer em despesas de *demurrage*.

Um porto é composto por píeres - com extensão, profundidade e pontos de atracação definidos - e bombas que podem ser restritivas em relação ao fluido bombeado. Dessa forma, cada navio pode atracar somente em um certos pieres, de acordo com as suas características e com as de sua carga [19]. É necessário considerar o tempo despendido para atracação e desatracação dos navios nos pieres. Somente após um navio ter desatracado totalmente é que o pier se torna disponível para a atracação de outro navio. Após o bombeio de petróleo de um navio para um tanque, este deve permanecer em repouso durante um tempo, para separar certas impurezas do óleo (e.g. salmoura e areia), especialmente as que possam aumentar a corrosão nos dutos. Somente após esse tempo de "descanso"("preparo", "sedimentação") é que o tanque pode ser utilizado para bombear óleo para um oleoduto [4]. No caso de um tanque ser destino de um produto final, ele é selado e certificado por análise laboratorial, e, somente após o resultado de certificação o produto pode ser enviado

a navios (ou dutos e tanques de transporte em caminhões ou trens). Esse tempo de análise laboratorial é tratado da mesma forma que o tempo de "descanso"no modelo matemático do porto. Note que a presença do tempo de "descanso"impede que os tanques no porto operem de forma "pulmão", recebendo e enviando material ao mesmo tempo. Uma outra regra possível na prática é evitar misturas em linha: somente um tanque deve ser empregado por vez, em cada operação de transferência.

2.2 Revisão Bibliográfica

A literatura é rica em abordagens para problemas de *scheduling* na cadeia de suprimentos da indústria de petróleo: simulação, modelos caixa-preta, heurísticas, meta-heurísticas, programação matemática etc. Um resumo comparativo é encontrado em [6]. Nesta revisão, nos restringimos a trabalhos recentes de programação matemática.

Em 1996, duas formulações de programação MILP para o *scheduling* de petróleo foram introduzidas [38], [19], as quais se tornaram fundamentais para diversos trabalhos à sequência no campo. Lee et al. [38] propuseram um modelo MILP detalhando a alocação de tanques de cru em refinarias, os quais recebiam itens de petróleo de dutos e alimentavam as unidades de destilação atmosférica (UDA). Shah [19], por sua vez, propôs uma formulação distinta para o *scheduling* de petróleo, desde os navios-tanques até as UDA, baseada em dois modelos complementares: (i) o modelo da refinaria ("downstream problem"), o qual definia a alocação de tanques na refinaria e a sequência de itens de petróleo nos dutos; (ii) o modelo de porto ("upstream problem") que definia a alocação dos navios nos pieres e de suas cargas nos tanques, de forma a atender a sequência de itens de petróleo definida no problema (i). Cada problema é modelado como um MILP: são utilizadas variáveis binárias para alocações de cru, navios e tanques, e contínuas para as quantidades de cru transportadas em cada intervalo de tempo. A discretização do tempo é feita em intervalos de duração fixa. Shah reportou que um exemplo simples foi resolvido em "alguns minutos"e que outras regras operacionais (e.g. misturas de petróleo, *demurrage*, tempo de "descanso/preparo") teriam de ser adicionadas ao modelo para que pudesse vir a ter algum uso prático.

Antes das publicações de Shah [19] e Lee et al. [38], desde os anos 50, a literatura de planejamento de operações de petróleo era composta, em grande parte, por modelos lineares (LP) [39], [40], [41] e, até mesmo, por alguns modelos não-lineares (NLP) [42], [43], [44], que se restringiam às quantidades e tipos de petróleo, aditivos e componentes que deveriam ser processados ao longo das próximas semanas ou meses para o atendimento dos objetivos comerciais estabelecidos, não tratando de alocações de equipamentos, sequências de atividades no tempo ou restrições operacionais (tempo de preparo/sedimentação, uso paralelo ou sequencial de tanques, locais para atracação de navios etc.). Os modelos NLP geralmente eram resolvidos por uma abordagem SLP (Sequential Linear Programming) [43], [45], ou seja, por aproximações lineares sucessivas, ou, em alguns casos, por métodos não-lineares como o GRG (Gradiente Reduzido Generalizado) [46]. Esses modelos evoluíram e ainda são utilizados em empresas de petróleo para planejamento de médio prazo de sua produção. *Softwares* deste tipo são, por exemplo, o Aspen PIMS e o Haverly GRTMPS, que utilizam modelos lineares (LP) para determinar a composição ideal de crus que deve ser utilizada em cada unidade de destilação de acordo com o tipo de produto cuja produção quer-se maximizar [33].

Magalhães e Shah [33] estendem o modelo anterior de Shah [19], considerando operações de "descanso/preparo", uma função-objetivo que mede o desvio do *schedule* em relação ao planejamento corporativo e discutem a discretização em "tempo-contínuo", um tópico ativo neste campo [23]. Casos da refinaria brasileira REFAP e de seu terminal marinho são exibidos. Os autores ressaltam que o crescimento das variáveis binárias pode tornar o problema intratável e que uma abordagem possível é fixarem-se certas decisões *a priori*, por serem usuais ao operador humano ou por representarem atividades já programadas quando da realização do *schedule*. Os autores também discutem que o modelo precisaria ser modificado para evitar decisões que podem ser consideradas por um ser humano como sub-ótimas, tais como trocas de tanques em uma operação de transferência, pequenos volumes transferidos, além de considerar custos de sobreestadia.

Floudas e Lin [7] apresentam uma revisão detalhada de diversos modelos MILP não apenas para a logística de petróleo, mas para a indústria de processos em geral. Alguns desafios que eles anteveem para o futuro próximo são a redução dos *gaps* de

integralidade e o uso de MILP para resolver problemas de médio prazo, distribuição e aplicações mais complexas. Wu et al. [6] apresentam outra revisão sobre logística de petróleo, onde destacam que, apesar de ser um tópico extensivamente estudado na academia ao longo das últimas decadas, ainda há um hiato significativo entre a teoria e as aplicações do mundo real. Os autores acreditam que o foco primário deveria ser na garantia da obtenção de *schedules* viáveis. Recentemente, Wu et al. [47], [48], [49], [50] exploraram esta ideia, desenvolvendo abordagens com *petri net* que permitem a obtenção de *schedules* viáveis, considerando diversas restrições operacionais complexas.

Karimi, Srinivasan e colaboradores [51], [52], [36], [53] desenvolveram abordagens MILP que são capazes de lidar com incertezas no suprimento de cru, além de, segundo os autores, serem os primeiros a consertar um erro que persistia em diversos modelos de *scheduling* de petróleo, desde a formulação original de Lee et al. [38]: a correta computação da concentração de cada componente após uma mistura de petróleos diferentes. Mais recentemente, estes modelos foram revisitados por Pan et al. [54], que desenvolveram uma nova formulação MILP, considerando um conjunto de regras heurísticas para operações de petróleo (descarga de navios e carga de unidades), que reduzem a complexidade do problema e substituem diversos termos bilineares. Eles testaram seu modelo nos exemplos de [51], [52] e [53]. No capítulo 4 deste texto, testamos o modelo não-linear proposto aqui nesta tese nestes mesmos problemas.

Neiro e Pinto [55] desenvolveram um modelo estocástico MINLP baseado em cenários com diversas probabilidades modelando incertezas nos preços de petróleo, produtos e demanda, de modo a conseguir robustez nos planos de refino. Mais recentemente, Li and Ierapetritou [56] também estudaram este assunto, discutindo o uso de formulações MILP de contra-parte para obtenção de *schedules* robustos.

Pinto e colaboradores [57], [5], [58], [12], [55] desenvolveram uma série de modelos MILP e MINLP para *planning* e *scheduling* de petróleo para aplicações reais da Petrobras, a partir da contribuição de Pinto em [38]. Em particular, Más e Pinto [4] apresentam um modelo MILP para 2 subsistemas: porto e centro de distribuição (refinarias são tratadas pelos outros modelos de Pinto e colaboradores). Os autores afirmam que o problema se torna intratável para casos reais de grande porte, e ilustram

esta afirmativa por meio de um limite superior para o número de variáveis binárias associadas ao problema. Operações de "descanso/sedimentação"e segregação de tanques de acordo com classes de petróleo são consideradas. A resolução dos modelos ocorre de forma hierárquica: primeiro o problema do porto é resolvido, e, em seguida, o problema do centro de distribuição, restrito pela programação do duto definida no primeiro problema. Refinarias são modeladas como centros consumidores de petróleo, com demanda média conhecida ao longo do cenário. O complexo da Petrobras em São Paulo é utilizado como problema-exemplo, sendo composto por um porto (GEBAST), dois centros de distribuição (SEBAT e SEGUA), conectados por dois dutos (OSVAT e OSBAT). Um problema de porto com dados reais (13 navios e 4 pieres) foi modelado em GAMS e resolvido com o solver CPLEX em aproximadamente 25 minutos.

Lee et al. [59] apresentaram um modelo MINLP para *scheduling* de nafta, bastante similar ao de petróleo. Considera-se um sistema logístico com porto e planta de produção de etileno, conectados por um duto. Quer-se determinar a alocação dos tanques e as sequências de atracação dos navios nos pieres, respeitando-se restrições operacionais e otimizando custos. A nafta estocada no porto é enviada por duto para a planta. O MINLP é resolvido por meio da resolução de aproximações sucessivas de problemas MILP.

Blanco et al. [60] discutem o problema de planejamento de terminais de distribuição de petróleo para refinarias (por oleodutos) e comércio marítimo (por meio de navios-tanque), possuidores de tancagem própria e misturadores, onde são feitas misturas dos crus recebidos. As datas de chegada dos navios são conhecidas *a priori*, bem como a demanda das refinarias pelos oleodutos. O modelo é resolvido com métodos de otimização global, por ser altamente não-convexo, com operações de mistura rigorosas, com termos bilineares. O objetivo é maximizar o lucro por meio da relação ótima entre os crus de baixa e alta qualidade nas misturas. É importante notar que os autores optaram por um modelo de maior grau de dificuldade, porém de melhor aderência aos fenômenos físico-químicos.

O problema de descarregamento de navios petroleiros com navios auxiliares realizando operações de "sangria"(*lightering*) foi abordado em [61] em um modelo MILP com tempo-contínuo, com uso de eventos. Cinco casos de estudo de dificuldade cres-

cente foram modelados em GAMS: o maior deles continha 12 petroleiros e 7 navios de lightering, gerando 114 variáveis binárias, 369 contínuas e 1326 restrições e foi resolvido pelo CPLEX em 100 minutos.

Uma abordagem de simulação é apresentada por Paolucci et al. [35]. Ao invés de procurar uma solução ótima, os autores propõem um sistema de apoio à decisão que permite ao usuário definir operações de transporte e observar as variações de estoque e qualidades. Um problema-exemplo é apresentado, considerando um sistema com um único ponto de atracamento, navios em fila, 4 tanques no porto, 5 tanques na refinaria e 3 crus. Outra abordagem baseada em simulação é apresentada por van Asperen et al. [62], onde estudam-se os efeitos nos estoques de um porto a partir dos diferentes processos de chegada de navios para carregamento de produtos e descarregamento de matéria-prima no porto. O sistema é semelhante ao de petróleo: os estoques funcionam como um *buffer*, mantendo um fluxo ininterrupto de matéria-prima para a planta mesmo com atrasos nos navios.

O problema da escolha dos pontos de atracação, de forma a otimizar a utilização dos equipamentos nos portos (*berth allocation problem*) é estudado, por exemplo, em [63]. Embora seja um problema importante para operação de portos de cargas em containers, ele não é um problema fundamental para a logística de petróleo, onde os terminais marítimos e portos apresentam alternativas bem definidas e poucas para atracação.

Poucos autores abordam o *scheduling* de produtos derivados de petróleo em portos, pois na maioria dos casos os produtos finais de uma refinaria são transportados por polidutos ou caminhões. Entretanto, Duarte [34] demonstrou em sua dissertação que exportações de produtos finais podem ser feitas pela via marítima, de forma semelhante à logística de petróleo.

Apesar das diferentes abordagens apresentadas, certos pontos são comuns: ETA são entradas para o *schedule*, mesmo se incertas; pieres são restritivos; tanques são segregados de acordo com produtos; misturas só podem ser modeladas rigorosamente em modelos não-lineares; em instâncias reais, modelos MILP sofrem de crescimento exponencial de acordo com os intervalos de tempo. Em resumo, a literatura é bastante escarsa em modelos de Programação Não-Linear contínuos para o importante problema do *scheduling* de petróleo.

2.3 Transporte Marítimo

Ronen constata em sua revisão em 1983 [64] que poucos trabalhos relativos à otimização do transporte oceânico por navios (*fleet scheduling*) foram produzidos entre os anos 50 (quando os primeiros trabalhos na área apareceram) e os anos 80 (época de seu artigo). Segundo o autor, as razões para essa escassez seriam: (i) tradicionalismo das companhias de transporte marítimo (uma indústria milenar); (ii) tamanho de instâncias reais (normalmente inviáveis para os solvers da época); (iii) relativa pouca importância do modo de transporte marítimo nos EUA; (iv) presença maciça de incertezas de difícil modelagem (greves em terra e em alto-mar, mudanças climáticas e problemas ambientais), entre outras. Em uma segunda revisão em 1993 [65], Ronen constata que o crescimento da produção acadêmica sobre transporte por navios durante o período 1983-93 ainda era aquém do esperado, considerando-se os avanços computacionais da década, e ainda muito menor que a literatura de outros problemas de transporte, como o de roteamento de veículos terrestres. Em 2004, Christiansen et al. realizam um terceiro *survey* [3], onde nota-se um avanço significativo da produção acadêmica na área, em particular nos países nórdicos. Segundo os autores, a desregulamentação do mercado de transporte marítimo e as grandes fusões entre companhias operadoras de navios na década de 90 levaram à redução de margens, aumento da concorrência e racionalização do uso da frota. Assim, os métodos tradicionais não seriam mais adequados para esse novo cenário, forçando as empresas a se modernizarem por meio de sistemas de apoio à decisão, e adesão de profissionais com formação em engenharia, pesquisa operacional e computação aos seus quadros de pessoal.

Provavelmente, os primeiros artigos com aplicações de programação matemática para o problema de transporte oceânico de petróleo são dois artigos de 1954, de Flood [66] e de Dantzig e Fulkerson [67], ambos baseados em modelos de programação linear (transportes e fluxos em rede, respectivamente). Em 1987, Brown et al. [68] abordaram o problema real de uma grande companhia petrolífera operando no Oriente Médio: a programação de navios petroleiros para entregas a compradores da América do Norte e da Europa, com rotas pelo Cabo da Boa Esperança (quando carregados) ou pelo Canal de Suez (quando vazios). Os autores adotaram uma estratégia de decomposição do problema: um modelo de programação inteira gera

schedules, que são então simulados por um simulador não-linear que considera a relação não-linear entre a velocidade de cruzeiro e o consumo de combustível dos navios. Apesar do modelo apresentar bons resultados em tempo computacional viável, os autores destacam que as soluções são instáveis: basta adicionar uma nova entrega para que o schedule tenha de ser totalmente recalculado. Sherali et al. [69] apresentam um modelo misto-inteiro linear (MILP) para um problema semelhante ao de [68]: a programação das rotas dos petroleiros da KPC (Kwait Petroleum Corporation), do Oriente Médio para clientes na América do Norte e na Europa, podendo navegar pelo Canal de Suez (navios de pequeno/médio porte e vazios) ou pelo Cabo da Boa Esperança (navios carregados ou super-petroleiros). As dimensões do MILP tornaram-no intratável computacionalmente na época, de modo que os autores decidiram adotar uma abordagem decompositora: resolver um modelo agregado, depois um detalhado, restringido pelo agregado. Uma das diferenças ressaltadas pelos autores em relação ao modelo de Brown et al. [68], está no fato de considerarem navios de vários tamanhos e compartimentos com diferentes cargas. Iakovou e Douligeris [70] abordam o problema de transporte de óleo na costa americana sob a óptica do risco ambiental. Eles afirmam possuir um modelo não-linear para minimização dos riscos de derramamento, mas, infelizmente, o mesmo não é apresentado no artigo. Hwang [71] apresenta um modelo MILP para o problema de *scheduling* de navios de materiais líquidos, considerando restrições de estoques. Uma série de restrições são linearizadas, o problema é decomposto em dois modelos lineares, um para navios e outro para portos e duas heurísticas são apresentadas para auxiliar à resolução do problema, uma que primeiro resolve o problema dos navios e depois dos portos, e outra que opera de forma contrária: primeiro os portos e depois os navios. Mais recentemente, Li et al. [72] discutiram o planejamento de uma frota heterogênea de navios, usada para transportar matérias-primas de diversos produtores para consumidores da indústria química. Eles desenvolveram um modelo MILP para o problema, considerando restrições de estoques nos produtores e nos consumidores, restrições operacionais nos equipamentos, transporte de múltiplas cargas em um mesmo navio, diferentes sequências de carga e descarga de materiais, e diferentes possibilidades de pontos de atracação nos portos. O modelo foi testado em duas pequenas instâncias no artigo.

Em resumo, o problema de transporte de petróleo por navios oceânicos é um problema aberto na literatura, geralmente de grande porte, para o qual ainda se buscam modelos eficientes para instâncias reais. Desta forma, é um problema tão ou mais complexo do que a logística de petróleo em terra, a partir dos terminais marítimos ou portos. No próximo capítulo, modelamos o problema da logística de petróleo considerando a previsão das datas de chegada (ETA) como entrada do problema.

Capítulo 3

Modelagem Matemática do Problema

3.1 Modelos para transferências entre equipamentos

A atividade fundamental do *scheduling* de petróleo é a operação de transferência de cru de um equipamento origem para um equipamento destino, com vazão ou volume bem definidos. Modelar a transferência é modelar o problema de *scheduling*, pois o *schedule* pode se resumir a uma sequência de transferências ao longo do tempo.

3.1.1 Exemplo 1: Refinaria

Consideremos o seguinte exemplo (Figura 3.1): uma pequena refinaria, com dois tanques de cru (T1 e T2) e uma unidade de destilação (CDU), onde ambos os tanques podem alimentar a unidade. Este sistema pode ser considerado como um grafo, formado por 3 nós (T1, T2 e CDU) e 2 arcos orientados (T1 - CDU) e (T2 - CDU). Podemos associar a cada arco uma variável que represente a vazão (ou volume) de petróleo do nó origem para o nó destino. Desta forma, a vazão de petróleo (por exemplo, em m^3) de T1 para CDU em um instante t é representada pela variável contínua não-negativa $u_1(t)$, enquanto a vazão de T2 para CDU no instante t é representada pela variável $u_2(t)$ e o vetor $u(t) = [u_1(t), u_2(t)]^T$ representa todas as vazões deste sistema no instante t. Por exemplo, suponhamos que se queira indicar que uma

transferência de 1000 m^3/h de T1 para CDU está ocorrendo no instante t=t_A, com T2 em repouso. Como representamos esta informação no sistema? Representa-se: $u(t_A) = [u_1(t_A), u_2(t_A)]^T = [1000, 0]^T$.

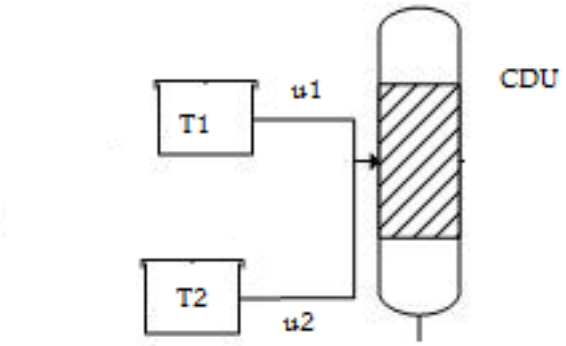

Figura 3.1: Exemplo simples de refinaria

O *schedule* desta refinaria, desde um instante inicial t_0 até um instante final T_f pode ser entendido como a definição da sequência de vetores $u(t)$, de $u(t_0)$ até $u(T_f)$. Suponhamos que se queira determinar um *schedule* para um cenário onde a CDU deve processar 2000 m^3 em 4 horas, ou seja, de $t_0 = 0$h até $T_f = 4$h. Uma possível solução seria definir um vetor u constante ao longo do cenário inteiro, tal que $u(t) = [500, 0]^T$, ou seja, definir uma única atividade de transferência de 4 horas, entre T1 e CDU. Outra solução poderia ser $u(t_1) = [500, 0]^T$, para $0 <= t_1 <= 2$ e $u(t_2) = [0, 500]$ para $2 < t_2 <= 4$, definindo duas atividades de transferência (a primeira de T1 para CDU, seguida por outra de T2 para CDU). Agora, suponhamos que seja importante forçar a seguinte restrição: "a CDU não pode ser alimentada por dois tanques em paralelo". Como representamos essa restrição? Apresentamos na Tabela 3.1 três formulações de Programação Matemática:

1. Formulação mista-inteira: introduzida independentemente por Shah [19] e Lee et al. [38] e adotada em vários modelos da literatura recente. Este modelo requer a adição de variáveis binárias, uma para cada arco do grafo (b_1 e b_2), que são utilizadas para representar a escolha entre T1 ou T2 como único tanque de carga da CDU em um dado instante. Se $b_1(t_A) = 1$, então, pela desigualdade $b_1(t_A) + b_2(t_A)$ ¡= 1, $b_2(t_A) = 0$, o que implica em $u_2(t_A) = 0$. Ora, isso significa que há uma vazão de T1 para CDU e nenhuma de T2 para CDU, respeitando

Tabela 3.1: Modelos para a refinaria exemplo

Modelo	Equações
(1) Mista-Inteira	$b_1(t_i) + b_2(t_i) <= 1$
	$b_1(t_i) * u_1^{MIN}(t_i) <= u_1(t_i) <= b_1(t_i) * u_1^{MAX}(t_i)$
	$b_2(t_i) * u_2^{MIN}(t_i) <= u_2(t_i) <= b_2(t_i) * u_2^{MAX}(t_i)$
	$Vol_{T1}(t_{i+1}) = Vol_{T1}(t_i) - (t_{i+1} - t_i) * u_1(t_i)$
	$Vol_{T2}(t_{i+1}) = Vol_{T2}(t_i) - (t_{i+1} - t_i) * u_2(t_i)$
	$b_1(t_i)$ e $b_2(t_i)$ binários, $u_1(t_i)$ e $u_2(t_i) \in \mathbf{R}$
(2) NLP não-suave	$u_1(t_i) + u_2(t_i)$ - max$\{u_1(t_i), u_2(t_i)\} = 0$
	$0 <= u_1(t_i) <= u_1^{MAX}(t_i)$
	$0 <= u_2(t_i) <= u_2^{MAX}(t_i)$
	$Vol_{T1}(t_{i+1}) = Vol_{T1}(t_i) - (t_{i+1} - t_i) * u_1(t_i)$
	$Vol_{T2}(t_{i+1}) = Vol_{T2}(t_i) - (t_{i+1} - t_i) * u_2(t_i)$
	$u_1(t_i)$ e $u_2(t_i) \in \mathbf{R}$
(3) NLP	$u_1(t_i) * u_2(t_i) = 0$
	$0 <= u_1(t_i) <= u_1^{MAX}(t_i)$
	$0 <= u_2(t_i) <= u_2^{MAX}(t_i)$
	$Vol_{T1}(t_{i+1}) = Vol_{T1}(t_i) - (t_{i+1} - t_i) * u_1(t_i)$
	$Vol_{T2}(t_{i+1}) = Vol_{T2}(t_i) - (t_{i+1} - t_i) * u_2(t_i)$
	$u_1(t_i)$ e $u_2(t_i) \in \mathbf{R}$

a restrição dada. Analogamente, se $b_2(t_A) = 1$, então, o T2 alimenta CDU em t_A e T1 está em repouso. Os volumes de T1 e T2 variam de acordo com as equações de balanço volumétrico;

2. Formulação NLP não-suave: apresentada na dissertação de mestrado do autor [73]. Se $u_1(t_A) > 0$, então, pela igualdade $u_1(t_i) + u_2(t_i)$ - max$u_1(t_i), u_2(t_i)$ $= 0$, concluí-se que $u_2(t_A) = 0$. Ora, isso significa que há uma vazão de T1 para CDU e nenhuma de T2 para CDU, respeitando a restrição dada. Analogamente, se $u_2(t_A) > 0$, então, o T2 alimenta CDU em t_A e T1 está em repouso. Esta formulação não requer variáveis binárias, mas apresenta uma dificuldade numérica devido à presenção da função max, cuja derivada

é descontínua, pois métodos usuais de otimização não-linear utilizam-se de gradientes, o que exige derivadas contínuas, pelo menos nas proximidades da solução. Cabe ressaltar, entretanto, que é possível aproximar a função max por funções suavizadoras (com derivada contínua), como as apresentadas em [74], [75] e [76]. Os volumes de T1 e T2 variam de acordo com as equações de balanço volumétrico;

3. Formulação NLP: proposta desta tese. Esta formulação só utiliza variáveis contínuas e funções totalmente diferenciáveis, sendo uma restrição uma equação de complementaridade, podendo ser resolvida por métodos de otimização não-linear (conforme [76], [77], [78], [79]). Se $u_1(t_A) > 0$, então, pela igualdade $u_1(t_i) * u_2(t_i) = 0$, conclui-se que $u_2(t_A) = 0$. Ora, isso significa que há uma vazão de T1 para CDU e nenhuma de T2 para CDU, respeitando a restrição dada. Analogamente, se $u_2(t_A) > 0$, então, o T2 alimenta CDU em t_A e T1 está em repouso. Os volumes de T1 e T2 variam de acordo com as equações de balanço volumétrico

As abordagens (2) e (3) usam o valor zero como limite inferior para todas as vazões. Em teoria, isto poderia permitir uma vazão irrealisticamente pequena, muito próxima de zero, na solução. Entretante, em problemas reais, isto não ocorre, pois uma vazão muito baixo acaba atrasando uma ou mais atividades, cujos atrasos incorrem em custos. Desta forma, é aceitável utilizar o limite inferior como zero.

3.1.2 Exemplo 2: Porto

Estudemos outro exemplo: um sistema logístico formado por um porto, com 3 píeres, 5 tanques (3 para produtos finais e 2 para petróleos), 2 dutos conectados a uma refinaria (um para petróleo, o outro para produtos finais) e 3 navios que devem ser atendidos. Neste exemplo, há algumas restrições físicas que se refletem no grafo de equipamentos e vazões:

- Todos os navios podem atracar no pier P2;
- Navio N1 não consegue atracar no pier P3, por causa de suas dimensões físicas (comprimento e calado);

- Navio N3 não atraca no pier P1, por causa de suas dimensões físicas (comprimento e calado);
- Tanques T1, T2 e T3 são utilizados para produtos finais, que recebem da refinaria pelo duto D1;
- Tanques T4 e T5 são utilizados para petróleo, que enviam para a refinaria pelo duto D2;
- Pier P1 é utilizado apenas para carregar navios de produtos e é conectado somente aos tanques T1, T2 e T3;
- Pier P2 é utilizado apenas para carregar navios de petróleo e é conectado aos tanques T4 e T5;
- Pier P3 é utilizado apenas para descarregar navios de petróleo e é conectado aos tanques T4 e T5;
- Navio N1 é um navio que deve receber uma carga de diesel para exportação;
- Navios N2 e N3 são navios com cargas de petróleo, que devem ser descarregadas no porto.

Neste exemplo, as vazões possíveis são determinadas não apenas pela existência de linhas de bombeamento entre pieres, tanques e dutos, mas também pelas cargas dos navios. Cabe ressaltar que, como o pier não armazena qualquer material, ele pode ser considerado apenas como formador do arco entre navios e tanques. Apesar de N2 poder atracar em P1, neste cenário isto nunca ocorreria, pois N2 é um navio de petróleo, e P1 só tem ligações com os tanques de produto. Desta maneira, existem arcos de T1, T2 e T3 para N1, via pier P1; e de N2 e N3 para T4 e T5, via pier P3. Como não há navio de petróleo para ser descarregado neste cenário, não há arcos passando por P2. A Figura 3.2 ilustra o sistema, apresentando em destaque três variáveis associadas a arcos: $u_{T2-P1-N1}$, ligando o tanque T2 ao navio N1; $u_{T5-P3-N3}$, ligando o navio N3 ao tanque T5; e u_{T4-D2} ligando o tanque T4 ao duto D2. As outras variáveis neste sistema são: $u_{T1-P1-N1}$, $u_{T3-P1-N1}$, u_{T1-D1}, u_{T2-D1}, u_{T3-D1}, $u_{T4-P3-N3}$ e u_{T5-D2}.

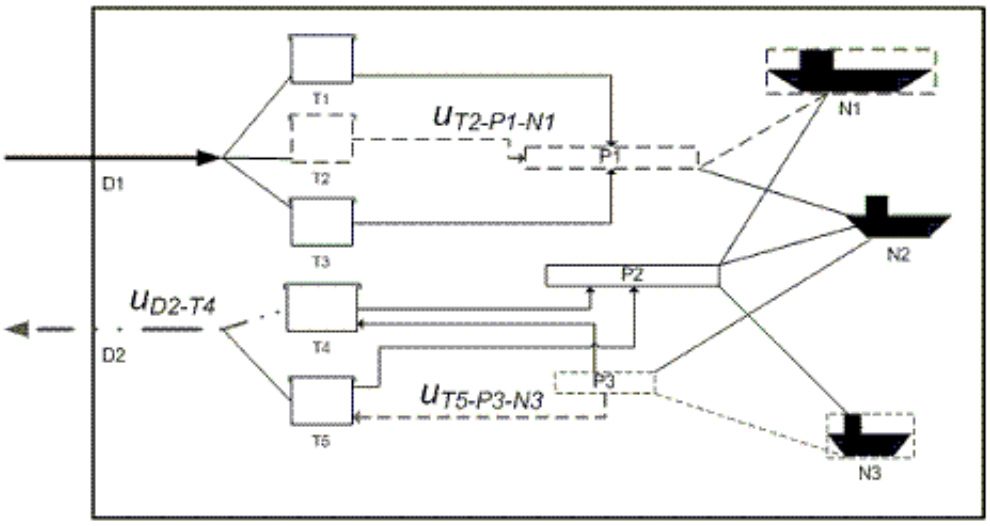

Figura 3.2: Porto como um Grafo

3.1.3 Formulação genérica da atividade de transferência

A partir dos exemplos, chegamos às equações genéricas para transferências entre equipamentos (Tabela 3.2), considerando balanço de volume e as seguintes restrições:

- Não é permitido que um equipamento opere em modo pulmão, enviando e recebendo material em um mesmo instante de tempo;
- Não é permitido que um equipamento receba material em paralelo, de dois outros equipamentos;
- Não é permitido que um equipamento envie material para dois outros equipamentos, simultaneamente.

Cada equipamento N é um nó do sistema e apresenta um conjunto A_N composto por vazões de entrada e saída do equipamento u_j, as quais são associadas aos arcos conectados a N. Cada vazão é restrita por limites máximo e mínimo. Pelas restrições dadas anteriormente, somente uma vazão u_j pode ser diferente de zero para cada equipamento N, em um dado instante t_i. No caso do modelo MILP, são adicionadas variáveis binárias b_j para cada u_j.

Analisando os modelos:

- O volume do equipamento N é calculado conforme a equação de balanço volumétrico nos 3 modelos.

Tabela 3.2: Modelos para transferências entre equipamentos

Modelo	Equações
(1) MILP	$vol_N(t_i) = vol_N(t_{i-1}) + \sum_{j \in A_N} u_j(t_i)\Delta t$
	$\sum_{j \in A_N} b_j(t_i) <= 1$
	$b_j(t_i) * u_j^{MIN}(t_i) <= u_j(t_i) <= b_j(t_i) * u_j^{MAX}(t_i), j \in A_N$
	$b_j(t_i)$ binário, $u_j(t_i) \in \mathbf{R}$
(2) Não suave	$vol_N(t_i) = vol_N(t_{i-1}) + \sum_{j \in A_N} u_j(t_i)\Delta t$
	$\sum_{j \in A_N} u_j(t_i) - max\{u_j(t_i) \mid j \in A_N\} = 0$
	$0 <= u_j(t_i) <= u_j^{MAX}(t_i), j \in A_N$
	$u_j(t_i) \in \mathbf{R}$
(3) NLP contínua	$vol_N(t_i) = vol_N(t_{i-1}) + \sum_{j \in A_N} u_j(t_i)\Delta t$
	$\sum_{j \in A_N} \sum_{k > j \in A_N} u_j(t_i) * u_k(t_i) = 0$
	$0 <= u_j(t_i) <= u_j^{MAX}(t_i), j \in A_N$
	$u_j(t_i) \in \mathbf{R}$

- No modelo (1), o somatório das variáveis binárias $b_j(t_i)$ é menor ou igual a um: somente uma variável binária associada aos arcos do nó N pode ter valor diferente de zero em t_i, implicando que somente uma vazão $u_j(t_i)$ associada ao equipamento N poderá ser diferente de zero em t_i;

- No modelo (2), o somatório das vazões $u_j(t_i)$ é igual à maior de todas as vazões $u_j(t_i)$, o que é verdade se todas as vazões menores ou iguais à vazão máxima forem zero em t_i. Ora, isto implica que somente uma vazão $u_j(t_i)$ associada ao equipamento N poderá ser diferente de zero em t_i;

- No modelo (3), o somatório dos termos bilineares $u_j(t_i) * u_k(t_i)$ é zero, o que é verdade se todas as vazões menores ou iguais à vazão máxima forem zero em t_i. Ora, isto implica que somente uma vazão $u_j(t_i)$ associada ao equipamento N poderá ser diferente de zero em t_i;

Em qualquer modelo de Programação Matemática, um passo crucial é a determinação das variáveis de otimização do problema sendo formulado. No caso do modelo NLP aqui apresentado, referimo-nos à definição das variáveis u_j, associa-

das aos arcos do grafo do sistema a ser otimizado, onde os nós são equipamentos (tanques, navios, dutos e unidades de processo) e os arcos são as conexões permitidas entre eles. Para determinarem-se estas conexões, deve-se proceder da seguinte forma:

1. Para cada tanque, verifique a existência de uma linha de tubulação conectando o tanque em questão a outros equipamentos (pieres, dutos ou unidades). Se o tanque for segregado para um tipo específico de produto ou classe de petróleo, ignore as conexões com equipamentos que não podem armazenar, bombear ou processar tais tipos de material. Por exemplo, se um tanque armazena cru pesado, e uma unidade de destilação não processa cru pesado, mesmo que haja alguma tubulação entre estes dois equipamentos, ela não poderá ser utilizada no *schedule*;

2. Para cada navio, verifique em quais pieres eles podem ser atracados (comprimento e calado compatíveis), bem como a compatibilidade da estrutura de bombeamento no pier. Para cada pier que pode ser utilizado por aquele navio, verifique quais tanques conectados ao pier em questão armazenam materiais compatíveis com as cargas do navio. Para cada par compatível Tanque-Navio, passando por aquele pier, crie um arco. Observação: se um mesmo navio puder ser bombeado de/para um tanque por mais de um pier, é necessário considerar um arco para cada pier, pois representam opções diferentes de atracação no *schedule*.

3. Cada arco identificado determina uma componente do vetor u de vazões. O limite máximo desta vazão é dado pela capacidade máxima de bombeamento no trecho entre os dois nós.

3.2 Modelo não-linear de controle ótimo

3.2.1 Problema de controle ótimo

Um problema de controle ótimo é definido matematicamente como o seguinte problema:

$$\text{Minimizar: } J = f(u, x, t)$$

$$\text{Sujeito a:}$$

$$u^{MIN} <= u(t_i) <= u^{MAX},\ t_0 <= t_i < t_F$$

$$x^{MIN} <= x(t_i) <= x^{MAX},\ t_0 <= t_i <= t_F$$

$$x(t_i) = g(x(t_{i-1}), u(t_{i-1}))\ ,\ t_0 < t_i <= t_F$$

$$x \in \mathbf{R}^n,\ u \in \mathbf{R}^m$$

O vetor $u(t_i)$ é o vetor de controle, cujas componentes são os valores das variáveis de controle no instante t_i, enquanto o vetor $x(t_i)$ é o vetor de estado, cujas componentes são os valores das variáveis de estado no instante t_i. A solução do problema é a sequência de vetores $u(t_i)$ e $x(t_i)$ ao longo do tempo, de forma a minimizar a função-objetivo J. Todas as variáveis apresentam limites inferiores e superiores, e os valores das variáveis de estado em t_i são calculados por equações de estado, a partir dos valores destas mesmas variáveis em t_{i-1}, bem como dos valores das variáveis de controle, também em t_{i-1}.

No modelo da logística de petróleo, as variáveis de controle são as vazões entre os equipamentos, enquanto as variáveis de estado são os estados dos equipamentos (volume, composição, propriedades físico-químicas).

3.2.2 O modelo proposto

O modelo não-linear de controle ótimo proposto modela o vetor de vazões u como o vetor de variáveis de controle (Eq. 3.2), e os vetores de volumes v e o de composição e propriedades p como subvetores do vetor de estado x (Eq. 3). Todas as variáveis estão restritas por limites superiores e inferiores. Os limites podem variar ao longo dos intervalos de tempo, de acordo com condições diversascomo altura das marés em certos períodos do dia, data de chegada de navios (um navio não pode ser utilizado antes de seu ETA) e manutenção programada de equipamentos. Deve-se ressaltar, no entanto, que todas os limites são determinados com a formulação do problema. As equações de estado foram desenvolvidas a partir do balanço volumétrico (Eq. 3.4) e da mistura de petróleos (Eq. 3.5) em base volumétrica. A função-objetivo (Eq. 3.1) é uma soma de diferentes custos, os quais podem ser priorizados pela

manipulação dos pesos w_{custo}.

$$MinimizarF(u,x,t) = \sum_{custo} w_{custo} * C_{custo}(u,x,t) \tag{3.1}$$

$$0 <= u(t_i) <= u^{MAX}(t_i) \tag{3.2}$$

$$x^{MIN} <= x(t_i) = [v(t_i), p(t_i)]^T <= x^{MAX} \tag{3.3}$$

$$v(t_i) = v(t_{i-1}) + Uu(t_{i-1})\Delta t \tag{3.4}$$

$$p_{N,q}(t_i) = (v_N(t_{i-1}) * p_{N,q}(t_{i-1}) + \sum_{j \in A_N} u_j(t_{i-1}) * p_{j,q}(t_{i-1}) * \Delta t)/v_N(t_i) \tag{3.5}$$

Equação 3.4 apresenta uma matriz de incidência U, cujas entradas são {0, 1, -1} para representar se a vazão implica em entrada de material (valor 1), saída de material (valor -1) ou se não há relação entre a vazão e o volume em questão (valor 0). A equação 3.5 calcula densidade, concentração (% vol.) de componentes como enxofre e a própria composição de petróleos em uma mistura: cada $u_j(t_{i-1})$ é uma vazão de entrada em N no instante t_{i-1}, enquanto $p_{j,q}(t_{i-1})$ é o valor da propriedade q nesta vazão de entrada. A equação de complementaridade (Eq. 3.6) garante que apenas um equipamento está alimentando N de fato, de modo que apenas um $u_j(t_{i-1})$ é maior que zero.

As equações seguintes modelam as decisões de *scheduling*: apenas uma origem e destino em cada transferência (Eq.3.6), tempo de preparo/descanso deve ser observado para segregar impurezas após o recebimento de uma carga e antes de um envio subsequente (Eq. 3.7), tempo de atracação para navios em um pier (Eq. 3.8), vazão constante para alguns equipamentos (Eq. 3.9). Todas definem novas variáveis de estado r_N, z_N, s_N, e q_N, cada uma associada a uma equação de complementaridade. Na próxima seção, estas variáveis serão utilizadas para relaxar e penalizar o problema.

$$r_N(t_i) = \sum_{j \in A_N} \sum_{k > j \in A_N} u_j(t_{i-1})u_k(t_{i-1}) = 0 \tag{3.6}$$

$$z_N(t_i) = \sum_{t'=t_{i-1}-\Delta t_N^{DESCANSO}}^{t_{i-1}} \sum_{j \in A_N} \sum_{k \in A_N^{SAIDA}} u_j(t') u_k(t_{i-1}) = 0 \tag{3.7}$$

$$s_N(t_i) = \sum_{t'=t_{i-1}-\Delta t_N^{ATRACACAO}}^{t_{i-1}} \sum_{j \in A_N} \sum_{k \in A_{K<>N}} u_j(t') u_k(t_{i-1}) = 0 \tag{3.8}$$

$$q_N(t_i) = u_{N,0} - \sum_{j \in A_N} u_j(t_i) = 0 \tag{3.9}$$

A Equação 3.6 restringe que apenas uma vazão pode estar alimentando o nó N no instante t_{i-1}: toda transferência tem apenas uma origem e um destino em t_{i-1}. A (Eq. 3.7) modela que N só pode enviar para outro equipamento se o tempo de preparo/descanso necessário $\Delta t_N^{DESCANSO}$ tiver sido respeitado. A (Eq. 3.8) modela o tempo de atracação $\Delta t_N^{ATRACACAO}$ para navios. Eq. 3.9 força uma vazão constante $u_{N,0}$ para um dado equipamento (geralmente um duto ou uma unidade de processo) N - esta restrição pode ser modificada facilmente para permitir vazão variável, mas limitada, se necessário.

Alguns custos foram considerados no modelo: custo de *demurrage* (Eq. 3.10), custo de não-atendimento da demanda planejada (Eq. 3.11), custo de estoques (3.12) e de variação de vazão (Eq. 3.13). Nem todos os custos precisam ser considerados em cada problema, sendo uma decisão do operador selecionar os custos mais relevantes, bem como as suas importâncias relativas (dadas pelos pesos).

$$C_{demurrage}(x,t) = \sum_{N \in Navios} \sum_{t_i > t_N^{saida}} c_N^{demurrage} (Carga_N - v_N(t_i))^2 \tag{3.10}$$

A Eq. 3.10 lida com o custo de *demurrage*: não empregamos a formulação clássica da demurrage [33], mas uma formulação alternativa, que gera um custo proporcional ao quadrado do volume da carga que ainda deve ser descarregada (ou carregada) em cada navio N, a partir do momento em que ele estiver atrasado, ou seja, a partir do momento em que não esteja mais em sua janela contratual (cujo tempo máximo é t_N^{saida}). O volume $v_N(t_i)$ representa o total de volume carregado ou descarregado do navio até o instante t_i. A carga $Carga_N$ é o volume total final que o navio deve conter ao final do *schedule*. Se o navio tiver de ser descarregado,

$Carga_N = 0$. Para navios de múltiplos compartimentos, considera-se uma carga para cada compartimento.

$$C_{demanda}(u,x,t) = \sum_{D \in Dutos} \sum_{M \in Materiais} c_D^{demanda}(Demanda_{D,M} - \sum_{t_i} u_{D,M}(t_i) * \Delta t)^2 \tag{3.11}$$

A Eq. 3.11 lida com o custo de não atendimento da demanda: cada duto (ou unidade de processo) D apresenta uma demanda $Demanda_{D,M}$ de material M que deve ser atendida pelo *schedule*, geralmente definida no planejamento de médio prazo da companhia. Utilizamos uma fórmula em que o custo é calculado ao final do cenário, dado pelo quadrado da diferença da demanda $Demanda_{D,M}$ menos a soma de todas as transferências ($u_{D,M}$) de M em D ao longo do cenário.

$$C_{estoque}(x,t) = \sum_{t_i} \sum_{Tq \in Tanques} c_{Tq}^{estoque} v_{Tq}(t_i) \tag{3.12}$$

$$C_{vazao}(u,t) = \sum_{t_i} \sum_{j \in Vazoes} (u_j(t_i) - u_j(t_{i-1}))^2 / u_j^{MAX}(t_{i-1}) \tag{3.13}$$

A Eq. 3.12 mede o custo de material imobilizado nos tanques (estoque) ao longo do cenário. Este custo, em geral, deve influenciar pouco na solução final, pois é mais importante, para o *scheduling* de petróleo, atender à demanda e não atrasar os navios. A Eq. 3.13 mede o custo de mudança de vazão, ao longo do cenário. Este custo visa evitar trocas desnecessárias de tanques nas atividades de transferência. Entretanto, assim como o custo de estoque, ele deve ter um peso pequeno na solução, para não incorrer em atrasos ou não atender à demanda. Outros mecanismos podem ser utilizados para reduzir as trocas de tanques: (a) forçar vazões constantes em certos arcos em certos intervalos de tempo (ex: por 12h ou 24h); (b) forçar a condição de que uma vazão em um arco só pode ser maior ou igual que a vazão anterior neste mesmo arco, dentro de certos intervalos de tempo (ex: por 12h ou 24h). Estas restrições diminuem a liberdade na escolha dos equipamentos envolvidos nas atividades.

O modelo proposto se aplica tanto a portos, quanto a refinarias ou a centros de distribuição. As equações físicas são as mesmas (Eq. 3.2 a 3.5). As restrições operacionais também são as mesmas: equações 3.6 e 3.7 são as mesmas, enquanto

a 3.8 deve ser aplicada apenas se houver navios para serem atracados na instância em questão. Já a Eq. 3.9, ela pode ser usada tanto para garantir a carga constante em uma unidade de processo quanto em um duto. Variáveis de estado relativas à carga de uma unidade de processo podem ser adicionadas ao modelo, se for necessário monitorar esta carga. Todas as equações relativas às restrições operacionais (Eq. 3.6 a 3.9)ou custos (Eq. 3.10 a 3.13) podem ser relaxadas ou retiradas do modelo, de acordo com as características da instância a ser modelada. Em todas as equações de custp, os parâmetros de custo c^{custo} são arbitrários e independentes, sendo determinados em cada problema.

3.3 Método de Resolução

Problemas de Controle Ótimo são equivalentes a problemas não-lineares e, portanto, podem ser resolvidos por típicos métodos de otimização não-linear [26], [80], [81]. Além disso, resultados recentes [76], [82], [77], [83] têm mostrado que problemas com restrições de complementaridade podem ser resolvidos eficientemente por métodos de otimização não-linear, particularmente com o uso de relaxações e penalidades nas equações de complementaridade. Assim, a abordagem que empregamos para a resolução do modelo não-linear de controle ótimo com equações de complementaridade será o uso de *solvers* não-lineares eficientes, que empregam métodos bem estabelecidos na literatura, aplicados a versões do modelo com as equações de complementaridade relaxadas e penalizadas na função-objetivo. Como a formulação é não-convexa, com diversos ótimos locais, a inicialização do problema influencia na qualidade da solução obtida.

3.3.1 Relaxação e Penalização

A formulação NLP original, com todas as restrições (Eq. 3.2 a 3.9) define uma região viável bastante restrita, especialmente por causa das equações de complementaridade. De fato, observamos na prática que várias instâncias, dependendo do ponto inicial escolhido, apresentavam dificuldades para explorar a região viável. Se um bom ponto inicial viável era escolhido, o método NLP convergia rapidamente para um ótimo local próximo. Se o ponto inicial era inviável, o método, em vários casos,

não conseguia convergir para uma solução viável.

De modo a tornar mais eficiente a busca por um ótimo local, relaxamos as restrições operacionais (Eq. 3.6 a 3.9) e os adicionamos como penalidades exatas na função-objetivo, ou seja, penalidades que, na região viável, apresentam valor zero. As restrições físicas (Eq. 3.2 a 3.5) foram mantidas como restrições. Isso gerou um problema relaxado de região viável maior, onde a única equação não-linear é a Eq. 3.4. É resolvida, então, uma sequência de problemas relaxados, até que as equações de complementaridade sejam atendidas. A função-objetivo do problema relaxado é:

$$G(u,x,t) = F(u,x,t) + \mu \sum_{t} (\rho_1 e^T r(t) + \rho_2 e^T s(t) + \rho_3 e^T z(t) + \rho_4 e^T q(t)) \quad (3.14)$$

, onde e é o vetor unário e ρ_i é um peso individual associado a cada restrição relaxada, que pode ser atualizado ou não a cada problema que resolvido. O parâmetro μ também pode ser determinado de forma iterativa, ao longo da resolução da sequência de problemas relaxados, ou pré-fixado *a priori*, como um valor suficientemente grande.

Alternativamente, para problemas em que as penalidades variam de forma muito desproporcional, outra função pode ser utilizada:

$$G(u,x,t) = F(u,x,t) + \mu \sum_{t} [ln(1+\rho_1||r(t)||), ln(1+\rho_2||s(t)||), ln(1+\rho_3||z(t)||), ln(1+\rho_4||q(t)||)]^T \quad (3.15)$$

3.3.2 Inicialização

Como o modelo é não-convexo, o ponto inicial pode ter um grande impacto na solução encontrada. Propomos, então, dois esquemas para inicialização:

- Inicialização em "pontos triviais", ou seja nos pontos onde o vetor de controle u é zero para todos os intervalos de tempo. Esses pontos são construídos de forma imediata, porém ou não são viáveis na formulação original ou apresentam custos máximos de *demurrage* e não-atendimento de demanda. Entretanto, estes pontos são sempre viáveis na versão relaxada do problema e definem uma

direção de descida que aponta para a minimização das penalidades, movendo-se para a região viável. Assim, o uso de "pontos triviais"é a escolha mais simples para a inicialização do problema;

- Inicialização em ponto construído por heurística: propomos uma heurística simples, que na maioria dos casos gera pontos viáveis no problema original, porém, em alguns casos, pode gerar pontos inviáveis tanto no modelo original quanto no modelo relaxado. A heurística pode ser resumida nos seguintes passos:

    1. Para cada duto ou unidade de processo, encontre um tanque disponível no instante inicial t_0, com material adequado, e o utilize para alimentação (ou recebimento) do duto ou da unidade;
    2. Ordene os navios por suas datas estimadas de chegada (ETA);
    3. Para cada navio, encontre um pier disponível e um tanque disponível na data superior mais próxima do seu ETA. Se não houver, estenda o horizonte de tempo até que uma data seja encontrada.

Na próxima seção, realizamos experimentos com ambas as inicializações.

Capítulo 4

Resultados Computacionais

4.1 Estratégia de validação do modelo

Visando a validação do modelo apresentado nos capítulos anteriores, foram executados diferentes casos de teste, agrupados em três grandes blocos:

1. Problemas 1 a 4: foram construídos de forma *ad hoc* para validar se as soluções obtidas eram *schedules* significativas e lógicas, respeitando todas as restrições operacionais e minimizando os custos de operações, em particular, o de *demurrage*. Estes resultados foram consolidados no artigo [30], submetido para publicação em periódico.

2. Comparativos MILP e NLP: versões simplificadas do problema, sem equações de mistura não-convexas, foram modeladas como NLP e MILP, para fins de comparação e teste da abordagem híbrida NLP-MILP. O objetivo destes testes era verificar que número de iterações e nós visitados para garantir-se o ótimo global do MILP era reduzido ao se inicializar o problema com a solução obtida pelo NLP, mesmo que a mesma fosse um ótimo local. Estes resultados foram publicados no capítulo de livro [29].

3. Casos da literatura: foram selecionados quatro problemas da literatura recente para validação do modelo NLP: um problema referente a uma refinaria brasileira (a REVAP) [58] e os três referentes aos que são considerados os melhores resultados atualmente obtidos pela abordagem MILP, conforme publicado em 2009 [54]. Os resultados do primeiro problema foram publicados no artigo [28],

enquanto os dos três outros problemas no artigo submetido para publicação [30].

É importante ressaltar que, em diversas áreas de pesquisa, problemas de teste estão disponíveis para realização de *benchmarks* e uso científico em geral, tanto na internet quanto na literatura formal. Infelizmente, este não é caso dos problemas de *scheduling* de petróleo, até por causa da natureza sigilosa de muitas aplicações. Obter modelos codificados em GAMS, AMPL ou outra linguagem de casos publicados na literatura para testes depende da receptividade de seus autores. No caso desta tese, foi possível obter de seus autores os arquivos GAMS do problema apresentado por Moro e Pinto [58], os quais serviram de base para construção do modelo NLP, codificado em AMPL, apresentado no Anexo A desta tese. Os outros modelos AMPL, também anexados à esta tese, foram construídos a partir das informações publicadas na literatura [54], enquanto os outros casos de teste foram construídos de forma *ad hoc*.

4.2 Problemas de Teste 1 a 4

O *solver* não-linear empregado nos testes de 1 a 4 foi o *solver* comercial LSGRG2 (Large Scale GRG2), desenvolvido a partir da implementação do método do Gradiente Reduzido Generalizado (GRG) por Lasdon e colaboradores para problemas de grande porte [84], e comercializado pela empresa Frontline Systems [85]. O *hardware* utilizado nestes testes foi um computador AMD Athlon XP 2.6 GHz, com 512 MB de RAM e sistema operacional Windows XP.

O método do (GRG) se adequa à estrutura do problema de controle ótimo [26], convergindo para um ótimo local e particionando o conjunto de variáveis do problema em dois: variáveis básicas e variáveis independentes. As dimensões do problema são reduzidas, pois somente o conjunto de variáveis independentes é manipulado iterativamente pelo método, enquanto as básicas são simplesmente computadas a partir dos valores das independentes. Ademais, Facó [27] mostra que a Jacobiana das restrições do problema de controle ótimo (Figura 4.1) é bem esparsa, em uma estrutura diagonal em blocos, a qual pode ser facilmente explorada para acelerar a convergência e reduzir o consumo de memória.

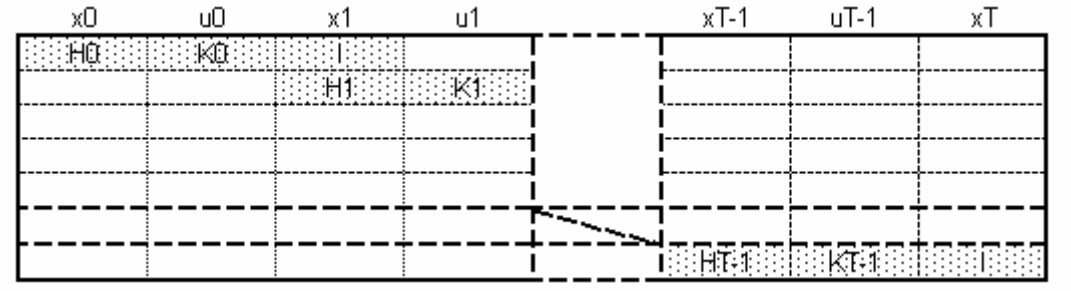

Figura 4.1: Jacobiana das restrições de um problem de controle ótimo

Cada caso apresentou uma dificuldade específica. Em todos os casos, a função-objetivo foi (3.14), exceto no problema 4, onde a função-objetivo foi (3.15). Em todos os casos os fatores de penalidade w_i iniciaram-se em zero, sendo atualizados para 1 no segundo problema relaxado, e, a partir de então, multiplicados por 10 a cada problema relaxado subsequente. Todos os casos foram inicializados com as variáveis de controle zeradas (o chamado "ponto trivial") ou determinadas pela rotina de heurística. Em geral, os casos inicializados pela heurística convergiram para soluções melhores que as dos inicializados no ponto trivial. Em todos os casos, a solução obtida é pelo menos um ótimo local, com pouco ou nenhum atraso no tratamento dos navios, baixo uso de tancagem e vazões estáveis, sem muitas variações.

4.2.1 Caso de Teste 1

O Problema 1 apresenta um cenário de 48 horas com 2 tanques, 1 pier de atracação e 2 navios que devem ser descarregados: 25.000 m^3 e 35.000 m^3 de um dado tipo de petróleo. A vazão de bombeamento máxima dos navios para os tanques é de 3.000 m^3/h. O ETA do primeiro navio é 0h, enquanto o do segundo é 12h. As datas máximas de partida dos navios são 24h e 36h, respectivamente. O tempo de atracação no terminal é de 3h, e o tempo de preparação do petróleo nos tanques é irrelevante para este problema. Ambos os tanques tem capacidade de armazenamento igual a 50.000 m^3 e podem ser utilizados indistintamente para as cargas dos navios. O primeiro tanque inicia o cenário com 15.000 m^3, enquanto o segundo com 10.000 m^3. Os parâmetros utilizados neste cenário são: $C_{ESTOQUE} = 0.05$ para ambos os tanques e $C_{DEMURRAGE} = 8$ para ambos os navios. Os pesos da função-objetivo foram $w_1 = 0.9$, $w_2 = 0.01$ e $w_3 = 0.09$.

Tabela 4.1: Caso de Teste 1

Caso	Estrutura	Variáveis	Iteração	F	G
1(a)	Tanques: 2	Controle: 48	Inicial	8.5 * 10^5	8.5 * 10^5
Tempo: $<$ 1s.	Navios: 2	Estado: 56	1	6331.79	6338.82
Inicial.: trivial	Píeres: 1	Não-linear: 2	2	4.77	10.05
Horizonte: 48h	Dutos: 0	Linear: 54	3	0.92	3.53
	Δ t = 4h		4	1.65	1.65

O problema foi inicializado no ponto trivial (u = 0), ou seja, sem descarregar os navios nos tanques. O *solver* convergiu para uma solução em menos de 1 segundo. A solução encontra conseguiu descarregar os dois navios sem qualquer atraso, portanto sem incorrer em custos de *demurrage*. A solução encontrada é um mínimo local, pois é possível melhorá-la por meio de ajustes nas variações de vazão. Entretanto, este benefício é desprezível frente ao obtido no custo de demurrage, que foi zero na solução encontrada. Um resumo é apresentado na Tabela 4.1.

4.2.2 Caso de Teste 2

O Problema 2 apresenta um cenário de 39 horas, com 3 tanques, 1 duto, 1 pier de atracação e 3 navios que devem ser descarregados: 35000 m^3, 10000 m^3 e 25000 m^3 de um tipo de petróleo. A vazão de bombeamento máxima dos navios para os tanques é de 3.000 m^3/h. O ETA do primeiro navio é 0h, o do segundo 8h e o do terceiro 12h. As datas máximas de partida dos navios são 24h, 27h e 36h, respectivamente. Tempos de atracação e de preparo (um tanque que recebeu petróleo só pode bombear para o duto se respeitar o tempo de preparo) são de 3 horas. Todos os tanques tem capacidade de armazenamento igual a 50.000 m^3 e podem ser utilizados indistintamente para as cargas dos navios. O primeiro tanque inicia o cenário com 15.000 m^3, o segundo com 10.000 m^3 e o terceiro com 45.000 m^3. Os parâmetros utilizados neste cenário são: $C_{ESTOQUE} = 0.05$ para os três tanques e $C_{DEMURRAGE} = 1$ para os três navios. A refinaria conectada ao duto tem uma demanda de 80.000 m^3 dentro deste cenário. A vazão de bombeamento máxima no duto é 3.000 m^3/h. Os pesos da função-objetivo foram $w_1 = 0.9$, $w_2 = 0.01$ e $w_3 = 0.09$.

Tabela 4.2: Caso de Teste 2

Caso	Estrutura	Variáveis	Iteração	F	G
2(a)	Tanques: 3	Controle: 126	Inicial	$6 * 10^5$	$6 * 10^5$
Tempo: 10s.	Navios: 3	Estado: 92	1	0.23	4.71
Inicial.: trivial	Píeres: 1	Não-linear: 4	2	4.72	4.82
Horizonte: 39h	Dutos: 1	Linear: 88	3	5.33	5.34
	Δ t = 3h				
2(b)	Tanques: 3	Controle: 126	Inicial	4.07	4.07
Tempo: 5s.	Navios: 3	Estado: 92	1	1.18	1.49
Inicial.: heuris.	Píeres: 1	Não-linear: 4	2	1.31	1.34
Horizonte: 39h	Dutos: 1	Linear: 88			
	Δ t = 3h				

O problema foi resolvido a partir de dois pontos iniciais diferentes:

- Caso (a): inicializado no ponto trivial, convergiu em 3 relaxações;
- Caso (b): inicializado a partir de um ponto obtido pela heurística, convergiu em 2 relaxações para uma solução melhor.

Em ambos os casos, todos os navios foram descarregados no prazo e a demanda da refinaria foi atendida, recebendo os 80.000 m^3 de cru dentro do cenário. Ademais, a importância de uma boa inicialização fez-se clara quando os dois casos são comparados (Tabela 4.2). Ambas as soluções são ótimos locais, sendo válidos para este problema os mesmos comentários feitos para o Problema 1. Em particular, o fato de a solução encontrada ser afetada pelo ponto inicial é algo esperado, dado que o problema é não-convexo e apresenta diversos ótimos locais. No futuro, uma estratégia de inicialização *multi-start* poderia vir a ser utilizada para obter soluções ainda melhores.

4.2.3 Caso de Teste 3

O Problema 3 é o Problema 2 com a adição de um novo pier e um novo navio. O quarto navio está vazio no início do cenário e precisa ser carregado com com 15000

Tabela 4.3: Caso de Teste 3

Caso	Estrutura	Variáveis	Iteração	F	G
3(a)	Tanques: 3	Controle: 196	Inicial	$9 * 10^5$	$9 * 10^5$
Tempo: 15s.	Navios: 4	Estado: 316	1	$1.6 * 10^3$	$1.6 * 10^3$
Inicial.: trivial	Píeres: 2	Não-linear: 232	2	43.05	48.48
Horizonte: 39h	Dutos: 1	Linear: 84	3	2.64	6.53
	Δ t = 3h		4	7.94	7.99
3(b)	Tanques: 3	Controle: 196	Inicial	4.31	4.31
Tempo: 30s.	Navios: 4	Estado: 316	1	4.00	4.07
Inicial.: heuris.	Píeres: 2	Não-linear: 232			
Horizonte: 39h	Dutos: 1	Linear: 84			
	Δ t = 3h				

m^3 de petróleo, na janela entre 21h e 33h. Todos os outros parâmetros são iguais aos do Problema 2.

O problema foi resolvido a partir de dois pontos iniciais diferentes:

- Caso (a): inicializado no ponto trivial, convergiu em 4 relaxações;
- Caso (b): inicializado a partir de um ponto obtido pela heurística, convergiu em 1 relaxação para uma solução melhor.

Em ambos os casos, todos os navios foram descarregados no prazo e a demanda da refinaria foi atendida dentro do cenário (Figuras 4.2 e 4.3). Como no Problema 2, observou-se que a importância de uma boa inicialização do problema: o ponto inicial do caso 3(b) já era uma boa solução, que foi ainda melhorada, convergindo para um ótimo local próximo (Tabela 4.3).

4.2.4 Caso de Teste 4

O Problema 4 representa uma situação onde os recursos disponíveis não são suficientes para evitar o atraso no atendimento dos navios. Em situações reais, é necessário decidir que navios poderão ser atrasados e quais terão de ser atendidos na janela contratual, de forma a minimizar o custo de *demurrage*. Problema 4 apresenta um

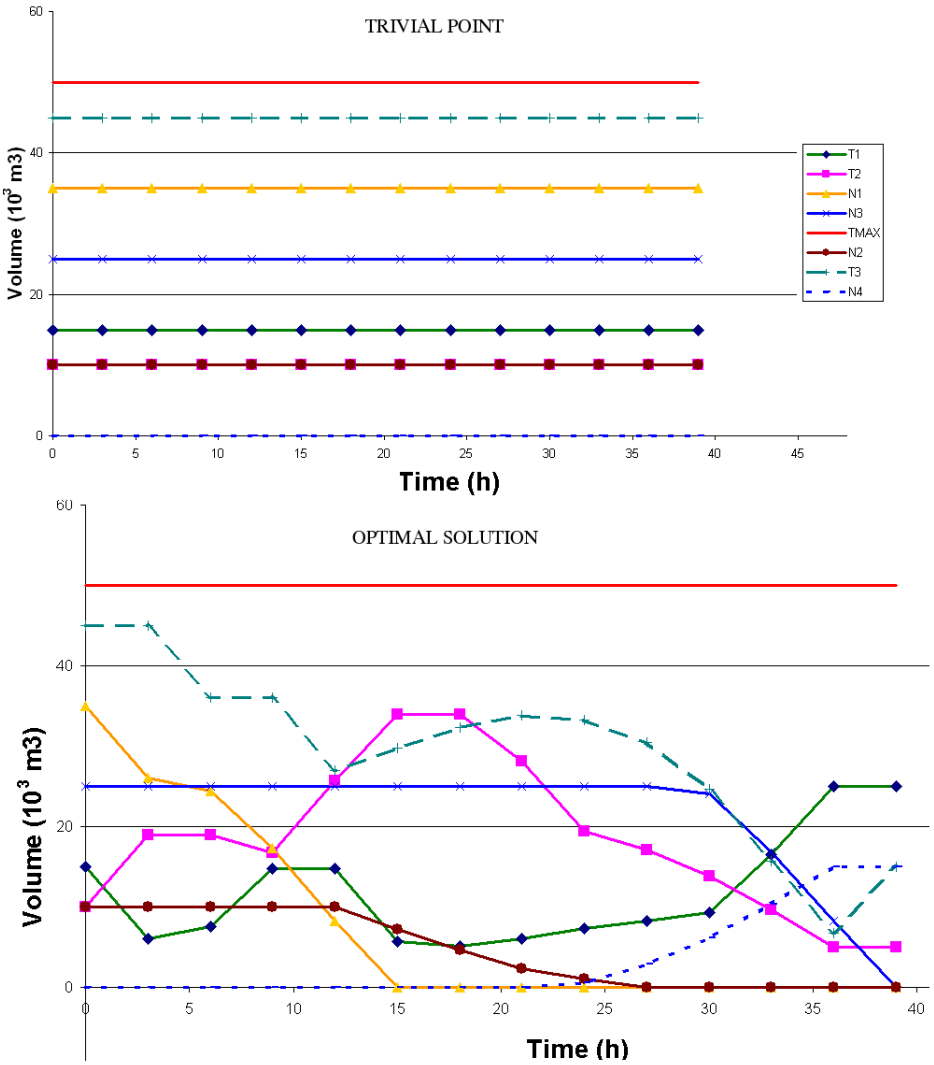

Figura 4.2: Volumes dos Tanques no Caso 3(a)

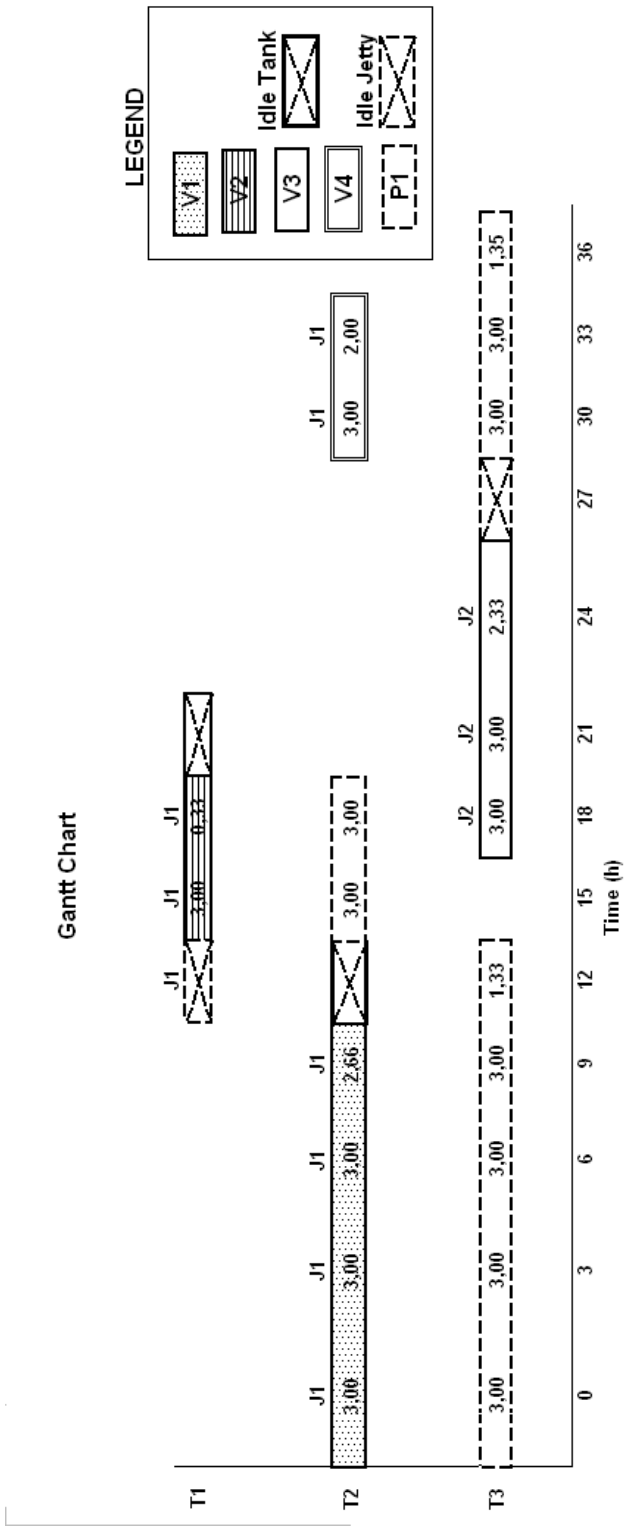

Figura 4.3: Gráfico de Gantt do Caso 3(a)

Tabela 4.4: Caso de Teste 4

Caso	Estrutura	Variáveis	Iteração	F	G
4	Tanques: 3	Controle: 650	Inicial	165.5	175.77
Tempo: 60s.	Navios: 8	Estado: 774	1	0.08	9.32
Inicial.: heuris.	Píeres: 2	Não-linear: 576	2	0.75	5.41
Horizonte: 72h	Dutos: 1	Linear: 198	3	5.24	5.25
	Δ t = 3h				

cenário de 3 dias com 3 tanques, 1 duto, 2 pieres e 8 navios. Navios 1, 2, 3, 6 e 7 devem ser descarregados de suas cargas de petróleo, respectivamente de volumes 35000 m^3, 10000 m^3, 25000 m^3, 35000 m^3 e 35000 m^3. Navios 4, 5 e 8 devem ser carregados com 15000 m^3, 18000 m^3 e 10000 m^3 de petróleo. A vazão de bombeamento máxima dos navios para os tanques é de 3.000 m^3/h. Os ETAs dos navios 1 a 8 são, respectivamente: 0h, 8h, 12h, 21h, 27h, 33h, 33h e 51h. As datas máximas de saída dos navios 1 a 8 são, respectivamente: 24h, 27h, 36h, 37h, 42h, 47h, 47h e 62h. Tempos de atracação e de preparo: 3 horas. Cada tanque pode armazenar até 50.000 m^3 de cru, e podem ser utilizados pelos 8 navios. Os volumes iniciais dos tanques 1, 2 e 3 são, respectivamente: 15000 m^3, 10000 m^3 and 45000 m^3. Os parâmetros utilizados neste cenário são: $C_{ESTOQUE} = 0.05$ para os três tanques e $C_{DEMURRAGE} = 8$ para os oito navios. A refinaria conectada ao duto tem uma demanda de 100.000 m^3 dentro deste cenário. A vazão de bombeamento máxima no duto é 3.000 m^3/h. A função objetivo é a (3.15).

O problema foi resolvido a partir de um ponto inicial inviável obtido pela heurística, que violava as capacidades dos tanques (Figura 4.4). O *solver* convergiu para uma boa solução, com custo de *demurrage* maior que zero, mas atrasando apenas um navio (Navio 6). Este atraso foi necessário para respeitar o tempo de preparo do tanque 2, que foi utilizado para atender o navio. A Figura 4.5 mostra a solução obtida. A Tabela 4.4 mostra a evolução da função objetiva e da penalidade ao longo das relaxações do problema.

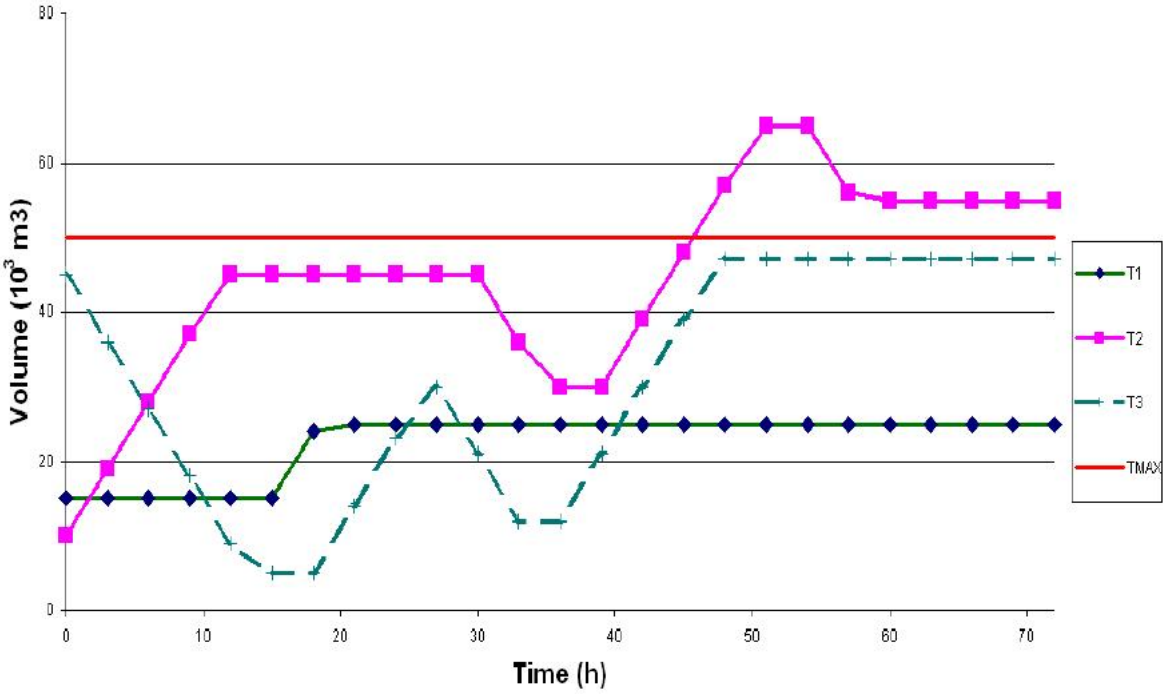

Figura 4.4: Volumes dos tanques no ponto dado pela heurística no Caso 4

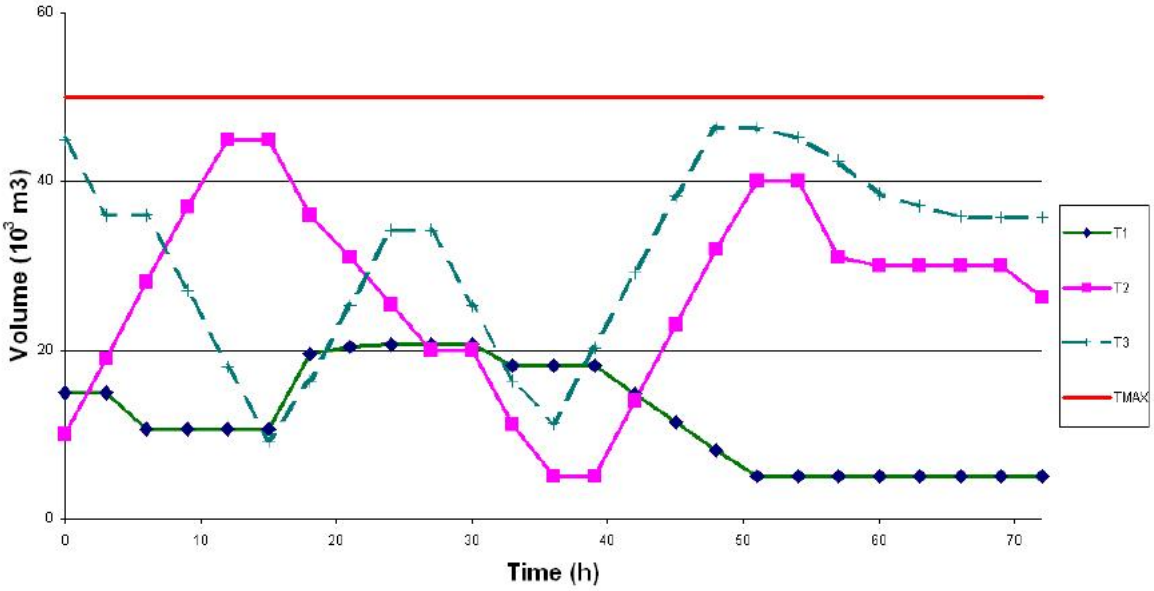

Figura 4.5: Volumes dos tanques na solução do Caso 4

4.3 Comparação entre MILP e NLP

4.3.1 Modelo MILP para comparação

Os quatro problemas da seção anterior visavam mostrar que o modelo não-linear é sólido, no sentido de ser capaz de gerar bons *schedules*, em tempo computacional aceitável. Entretanto, é importante comparar desempenho e qualidade de solução em relação a modelos MILP de porte semelhante. Para gerar uma comparação justa, os modelos MILP e NLP devem ter as mesmas funções-objetivo lineares (soma direta de custos) e um conjunto de restrições semelhante (limites máximo e mínimo de capacidade dos tanques e tempos de atracação e de preparo). Retiramos, então, as restrições de misturas de petróleo, pois um modelo MILP não conseguiria resolvê-las diretamente. Essas serão utilizadas na próxima seção, quando onde são resolvidos problemas da literatura. Ainda no intuito de realizar uma comparação justa, decidimos utilizar a linguagem de modelagem AMPL [86] e resolver os problemas com *solvers* comerciais.

4.3.2 Casos de Teste MILP e NLP

Devido a uma restrição da licença do LSGRG2, que limita o número total de variáveis do modelo NLP, decidimos utilizar as licenças acadêmicas gratuitas dos *solvers* CPLEX (v. 10.1.0) [87], SNOPT (v. 6.1) [88] e MINOS (v. 5.5) [89]. Os três pacotes são integrados ao ambiente computacional do AMPL, sendo o primeiro voltado para otimização de problemas MILP e os outros dois para otimização não-linear contínua.

Empregamos dois *solvers* não-lineares para constatar se os resultados obtidos seriam muito diferentes de acordo com o método de otimização empregado. O SNOPT [88] foi desenvolvido para problemas não-lineares convexos ou não-convexos com restrições de igualdade ou desigualdade de grande porte, preferencialmente esparsas. Seu algoritmo se baseia em um método do tipo SQP (Sequential Quadratic Programming), o que significa que aproxima o problema original por uma sequência de subproblemas quadráticos (QP), minimizando uma aproximação quadrática do Lagrangeano aumentado sujeito a restrições lineares. Ele converge para ótimos locais a partir de qualquer ponto inicial dado, sendo mais eficiente se poucas restrições

Tabela 4.5: Dimensões dos Problemas de Teste NLP e MILP

Modelo	Caso	Variáveis Binárias	Variáveis Contínuas	Restrições
NLP	1(A)	0	31	25
	1(B)	0	31	30
	2	0	111	87
	3(A)	0	169	103
	3(B)	0	135	97
MILP	1(A)	12	25	31
	1(B)	12	25	36
	2	34	82	111
	3(A)	93	158	265
	3(B)	93	158	275

não-lineares estiverem ativas e a maioria das lineares estiverem ativas. As funções e variáveis devem ser todas contínuas. O MINOS [89], por sua vez, também foi desenvolvido para problemas não-lineares convexos e não-convexos, com restrições de igualdade e desigualdades, preferencialmente esparsas. Ele também trabalha com uma sequência de subproblemas, onde a função-objetivo é uma aproximação linear do Lagrangeano aumentado, sujeito a restrições lineares. Ele converge para ótimos locais, dependendo do ponto inicial. A convergência não é garantida para qualquer ponto.

O CPLEX [87] é considerado com um *solver* no estado-da-arte para problemas misto-inteiros, implementando um algoritmo extremamente eficiente de *Branch and Bound* com diversas heurísticas embutidas e uso de planos de corte (*cutting planes*) para acelerar a convergência e garantir a otimalidade global da solução.

Foram executados 5 casos de teste, representando um terminal marítimo conectado a uma refinaria por meio de um oleoduto, inicializados no ponto trivial (u = 0). Todos foram resolvidos em tempo computacional de aproximadamente 1 segundo em um computador Intel Core Duo T2250 1.73GHz, com 1 GB de RAM e sistema operacional Linux OpenSUSE 10.1. Os resultados estão consolidados nas Tabelas 4.5 e 4.6.

- Caso 1 (A): terminal com dois tanques de cru e um oleoduto para uma refinaria, cuja demanda deve ser atendida. O oleoduto pode parar de bombear em alguns intervalos de tempo. Ambos os tanques têm capacidade máxima de 50.000 m^3 e mínima de 10.000 m^3. Inicialmente, o tanque 1 tem 40.000 m^3, enquanto o tanque 2 tem 20.000 m^3. A demanda da refinaria é de 30.000 m^3. A vazão máxima de bombeamento no duto é de 8.000 m^3. A função objetivo é minimizar a soma do custo de não atendimento da demanda da refinaria com o custo de armazenamento dos tanques ao longo do cenário. Os parâmetros são: $C^{T_1}_{ESTOQUE} = 5$, $C^{T_2}_{ESTOQUE} = 10$, $C_{DEMANDA} = 10$, $w_{ESTOQUE} = 1$ e $w_{DEMANDA} = 1$. Os 3 *solvers* convergiram para o ótimo global.

- Caso 1 (B): idêntico ao 1 (A), exceto pelo fato de que o duto deve ser bombeado continuamente, sem interrupções ao longo do cenário. Os 3 *solvers* convergiram: CPLEX e SNOPT para ótimo global, MINOS para ótimo local.

- Caso 2 : terminal com dois tanques de cru, um pier e dois navios que devem ser descarregados. Ambos os tanques têm capacidade máxima de 50.000 m^3 e mínima de 5.000 m^3. Inicialmente, o tanque 1 tem 15.000 m^3, enquanto o tanque 2 tem 10.000 m^3. As cargas dos navios são 35.000 m^3 e 25.000 m^3. O navio 1 deve ser descarregado entre 1h e 7h, enquanto o navio 2 entre 4h e 10h. O tempo de atracação é de 1h. A vazão máxima de bombeamento no pier é de 12.000 m^3. A função objetivo é minimizar a soma dos custos de *demurrage*, armazenamento dos tanques e variação de vazão ao longo do cenário. Os parâmetros são: $C^{T_1}_{ESTOQUE} = 0.05$, $C^{T_2}_{ESTOQUE} = 0.05$, $C_{DEMURRAGE} = 8$ para os dois navios, $C_{VAZAO} = 1$, $w_{ESTOQUE} = 0.01$, $w_{DEMURRAGE} = 0.9$ e $w_{VAZAO} = 0.09$. Os 3 *solvers* convergiram para o ótimo global.

- Caso 3 (A): terminal com três tanques de cru, um pier, três navios que devem ser descarregados, e um duto conectado a um refinaria, cuja demanda deve ser atendida. O oleoduto pode parar de bombear em alguns intervalos de tempo. Os três tanques têm capacidade máxima de 50.000 m^3 e mínima de 5.000 m^3. Inicialmente, o tanque 1 tem 15.000 m^3, o tanque 2 tem 10.000 m^3 e o tanque 3 tem 45.000 m^3. As cargas dos navios são 35.000 m^3, 10.000 m^3 e 25.000 m^3. O navio 1 deve ser descarregado entre 1h e 8h, o navio 2 entre 4h e 9h e o

Tabela 4.6: Resultados Comparativos NLP x MILP

Modelo	Caso	Solução	Iterações	Ótimo Global
NLP (SNOPT)	1(A)	1460	51	Sim
	1(B)	1600	13	Sim
	2	0.33	12	Sim
	3(A)	0	812	Sim
	3(B)	18.27	544	Não
NLP (MINOS)	1(A)	1460	17	Sim
	1(B)	1625	5	Não
	2	0.33	191	Sim
	3(A)	0	411	Sim
	3(B)	0	472	Sim
MILP (CPLEX)	1(A)	1460	14	Sim
	1(B)	1600	13	Sim
	2	0.33	63	Sim
	3(A)	0	324 (8 BB nós)	Sim
	3(B)	0	397 (25 BB nós)	Sim

navio 3 entre 5h e 11h. O tempos de atracação e de preparo iguais a 1h. A vazão máxima de bombeamento no duto e no pier é 12.000 m^3. A demanda da refinaria é de 80.000 m^3. A função objetivo é minimizar a soma dos custos de *demurrage* com não atendimento da demanda da refinaria ao longo do cenário. Os parâmetros são: $C_{DEMURRAGE} = 1$ para os três navios, $C_{DEMANDA} = 1$, $w_{DEMURRAGE} = 0.5$ e $w_{DEMANDA} = 0.5$. Os 3 *solvers* convergiram para o ótimo global.

- Caso 3 (B): idêntico ao 3 (A), exceto pelo fato de que o duto deve ser bombeado continuamente, sem interrupções ao longo do cenário. Os 3 *solvers* convergiram: CPLEX e MINOS para ótimo global, SNOPT para ótimo local.

4.3.3 Abordagem híbrida NLP-MILP

Como o modelo não-linear adotado é não convexo, por causa das equações de complementaridade, os *solvers* MINOS e SNOPT convergiram para ótimos locais em alguns casos, o que não ocorre no modelo misto-inteiro linear, cujo ótimo sempre é o global. Entretanto, como o modelo não-linear é mais compacto, com menor número de variáveis e restrições, ele pode ser utilizado para acelerar a convergência do MILP. Uma solução NLP é equivalente a uma solução MILP, dado que as restrições de capacidade, tempo de atracação, tempo de preparo e unicidade origem-destino nas transferências são respeitadas, as demandas atendidas e os navios descarregados a contento. Dessa maneira, é possível inicializar o problema MILP com uma solução viável e integral obtida pelo NLP, a qual gerará um limite superior para a função objetivo do MILP.

Desta forma, pode-se desenvolver o seguinte esquema híbrido (Figura 4.6): resolver o NLP contínuo, obtendo um solução local, que servirá de ponto inicial para o MILP, fornecendo um limite para a função-objetivo. Este esquema pode ser estendido indefinidamente, conforme os nós da árvore de B&B forem sendo visitados, tendo suas variáveis inteiras são fixadas em 0 ou 1 (mecanismo de *branching*) e gerando problemas com menor grau de liberdade, os quais podem ser novamente convertidos em NLP, potencialmente gerando novos limites superiores para a função-objetivo do MILP original.

A Tabela 4.7 mostra o número de iterações empregadas pelo CPLEX para garantir a convergência para o ótimo global em cada um dos exemplos. A solução originalmente obtida em cada NLP é convertida para um ponto MILP pela simples adição de variáveis binárias, e pela substituição das restrições de complementaridade pelas restrições sobre as variáveis binárias. Para cada vazão positiva no NLP, a correspondente variável binária recebe o valor 1, enquanto para cada vazão zero no NLP, a binária correspondente recebe o valor zero. A realização deste procedimento uma única vez, realizando a simples inicialização do MILP com uma solução obtida pelo modelo não-linear, já foi suficiente para reduzir significativamente o número de iterações e nós visitados na resolução do MILP.

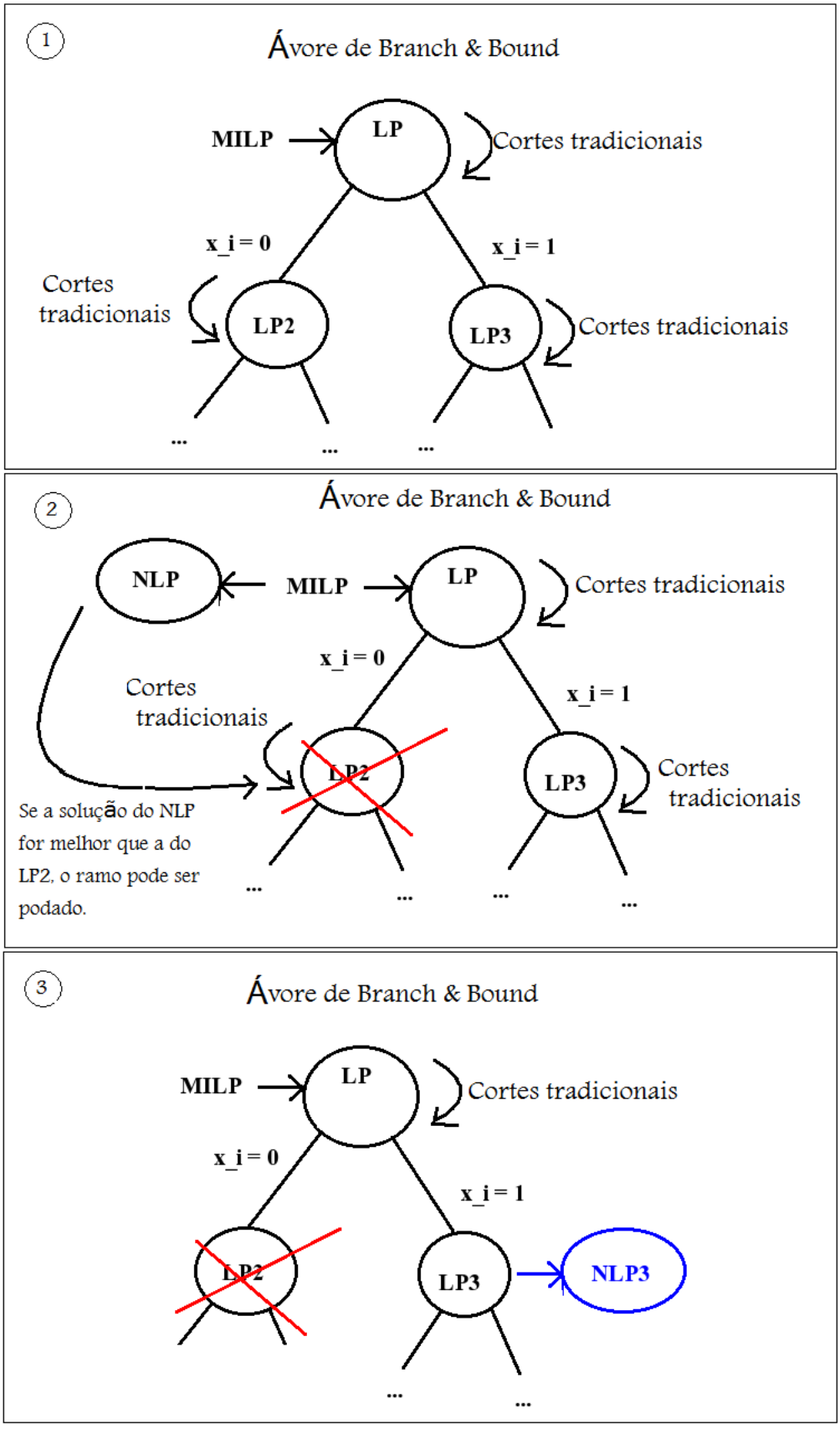

Figura 4.6: Desenho esquemático do procedimento híbrido NLP-MILP

Tabela 4.7: Resultados MILP a partir de diferentes inicializações

Caso	Ponto Inicial (x, u)	Iterações
1(A)	$(x^0, 0)$	14
	$(x, u)^{SNOPT}$	13
	$(x, u)^{MINOS}$	13
1(B)	$(x^0, 0)$	13
	$(x, u)^{SNOPT}$	4
	$(x, u)^{MINOS}$	4
2	$(x^0, 0)$	63
	$(x, u)^{SNOPT}$	47
	$(x, u)^{MINOS}$	40
3(A)	$(x^0, 0)$	324 (8 BB nós)
	$(x, u)^{SNOPT}$	215
	$(x, u)^{MINOS}$	215
3(B)	$(x^0, 0)$	397 (25 BB nós)
	$(x, u)^{SNOPT}$	265 (6 BB nós)
	$(x, u)^{MINOS}$	215

4.4 Resolvendo problemas da literatura

Em adição aos testes anteriores, o modelo não-linear foi aplicado aos seguintes problemas da literatura:

1. Problema de *scheduling* de petróleo em uma refinaria, apresentado por Moro e Pinto [58];

2. Três problemas de *scheduling* de petróleo em um sistema porto-refinaria estudado por Reddy et al. [51], [52] e revisitado por Pan et al. [54].

Os problemas foram modelados em AMPL (ver Anexo A desta tese), com alguns milhares de variáveis e restrições, e resolvidos remotamento com o *solver* não-linear SNOPT, disponibilizado para uso gratuito pela internet no sítio do NEOS Server [90]. Por causa das dimensões destes problemas, eles não puderam ser resolvidos

com as licenças acadêmicas utilizadas nos problemas da seção anterior. Como a configuração de *hardware* do NEOS Server nos é desconhecida, o tempos computacionais reportados pelos autores não podem ser comparados diretamente com os tempos obtidos na resolução de nossos modelos. Entretanto, é importante registrar esta informação para cada problema resolvido, de modo a evidenciar que os modelos propostos puderam ser resolvidos em tempos aceitáveis, o que era um dos objetivos desta proposta.

4.4.1 Caso de Teste 5: Programação de petróleo em uma refinaria (Moro e Pinto, 2004)

Moro e Pinto [58] apresentaram um problema baseado na área de cru de uma refinaria real da Petrobras, a REVAP, localizada em São José dos Campos, interior do estado de São Paulo. O objetivo deste cenário é programar o recebimento de 4 itens de petróleo nos 6 tanques de cru da refinaria, ao longo de 5 dias, mantendo a operação contínua da unidade de destilação atmosférica (UDA), preferencialmente em carga máxima.

- Os itens de petróleo são de 3 tipos diferentes e chegam à refinaria por meio de um oleoduto: Item 1 (60000 m^3, 100% Bonito, 8h - 20h), Item 2 (40000 m^3, 100% Bonito, 48h - 58h), Item 3 (1000 m^3, 100% Marlim, 58h - 59h) e Item 4(60000 m^3, 100% RGN, 100h - 112h).

- Nenhum tanque pode operar em operação pulmão (enviando e recebendo simultaneamente petróleo) e o tempo de preparo é de 24h.

- Todos os tanques podem alimentar a UDA individualmente ou em par com outro tanque. A carga na UDA deve ser de até 1500 m^3/h, com no máximo 50% de Marlim em sua composição.

- Estado inicial dos tanques: T1 (40000 m^3, 50% Marlim, 50% Bonito), T2 (50000 m^3, 100% Marlim), T3 (15000 m^3, 30% RGN, 70% Bonito), T4 (50000 m^3, 100% Marlim), T5 (20000 m^3, 60% Marlim, 40% RGN) e T6 (15000 m^3, 60% Marlim, 30% Bonito, 10% RGN).

Moro e Pinto desenvolveram dois modelos para o problema, um MILP e um MINLP, codificados em GAMS [91]. O modelo MILP empregava uma aproximação linear das equações de mistura, utilizadas para calcular as composições de petróleos nos tanques e na carga da UDA. O modelo MINLP empregava equações não-lineares de mistura (soma ponderada das parcelas) para calcular as composições de petróleos nos tanques e na carga da UDA. Em ambos os modelos, a função-objetivo era maximizar a quantidade de óleo processado pela UDA. Em seu artigo, eles reportaram que o modelo MINLP foi resolvido com o *solver* DICOPT [92] em 30 minutos em um Pentium 700 MHz, obtendo uma solução capaz de processar o máximo de petróleo possível, operando a UDA em vazão máximo ao longo dos 5 dias de cenário. Já o modelo MILP foi executado no *solver* CPLEX, também em um Pentium 700 MHz, porém precisou ser abortado após 105 minutos de execução do *solver*, por estourar o tempo máximo permitido para sua execução. A solução MILP obtida foi subótima, pois não processava o máximo de petróleo possível (a UDA trabalhava em uma vazão inferior à máxima).

A solução do nosso modelo não-linear foi obtida em 30 segundos (job 1701755 no host shepherd.mcs.anl.gov) e satisfez todos os requisitos: processou o máximo de petróleo possível, mantendo a UDA em operação constante com carga máxima e respeitou todas as restrições (composição na carga, capacidade dos tanques, tempo de preparo, etc.). Uma característica interessante da nossa solução foi que o uso de tanques também foi otimizado: apenas 5 tanques precisaram ser utilizados na operação da refinaria, ficando o tanque T3 em repouso ao longo dos 5 dias.

Os resultados são apresentados nas tabelas 4.8, 4.9 e 4.10 e nas figuras 4.7, 4.8 e 4.9. Apesar de os tempos computacionais obtidos nos 3 modelos não poderem ser comparados diretamente, dada o desconhecimento do *hardware* empregado pelo NEOS Server, o baixo número de iterações no NLP é um indicativo que o modelo proposto é uma opção válida para o problema de *scheduling* de petróleo.

4.4.2 Programação de petróleo para uma refinaria com porto (Pan et al., 2009)

Os casos de teste 6, 7 e 8 se baseiam em uma refinaria típica estudada por Reddy et al. [52] e Pan et al. [54], incluindo 8 tanques (T1 - T8), 3 UDAs (CDU1 - CDU3)

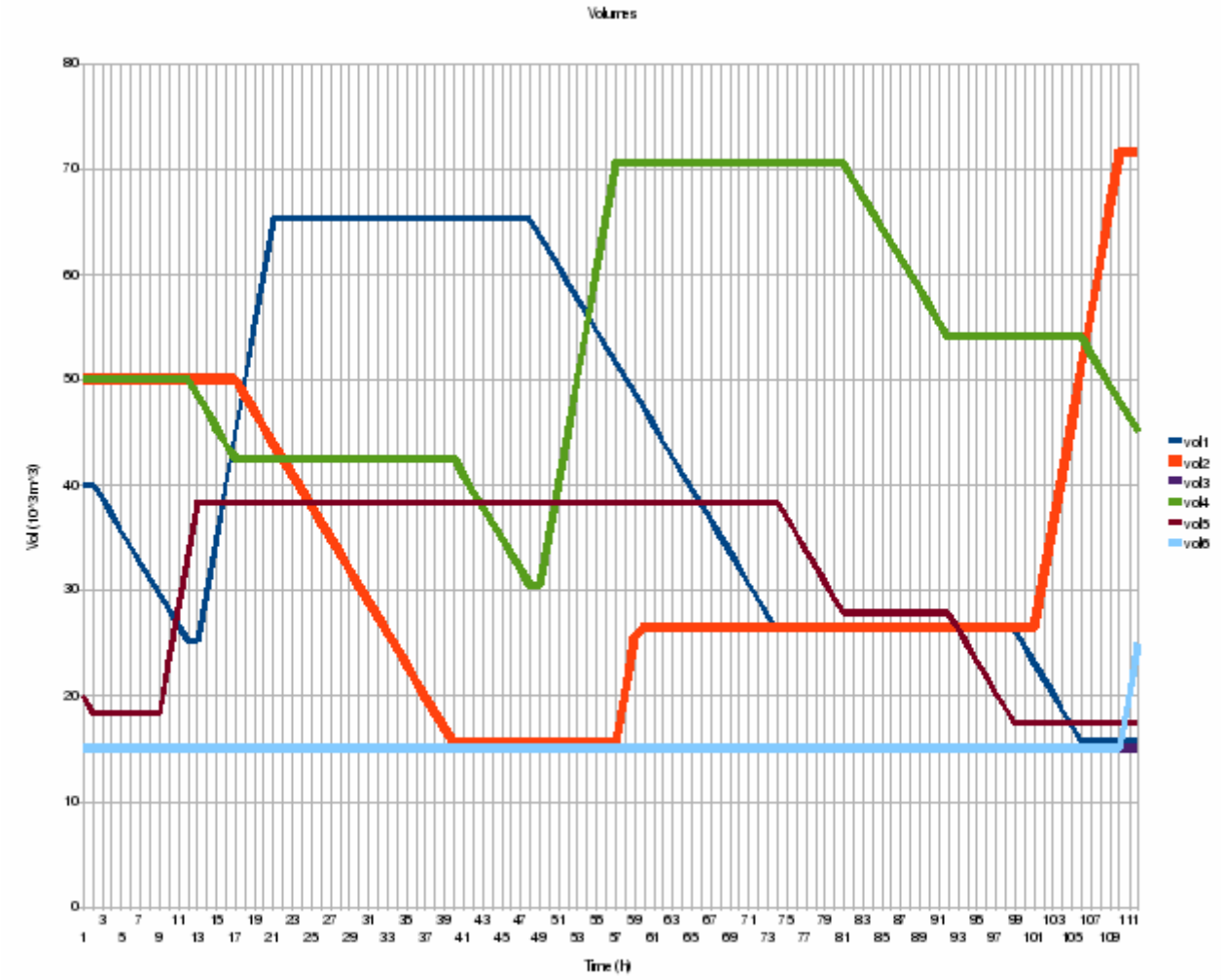

Figura 4.7: Volumes dos tanques na solução do Caso 5

Tabela 4.8: Comparação das dimensões dos modelos

Modelo	Caso	Variáveis Binárias	Variáveis Contínuas	Restrições
NLP	5	0	3727	3108
MILP	5	912	4246	5175
MINLP	5	228	2226	2634

Tabela 4.9: Resultados

Modelo	Caso	Solução (Vol. de cru processado (m^3))	Iterações
NLP (SNOPT)	5	168000 (UDA com carga máxima)	33536
MILP (CPLEX)	5	143109	856297
MINLP (DICOPT)	5	168000 (UDA com carga máxima)	330370

e 8 tipos de petróleo (C1 - C8), agregados em 2 classes de petróleo (P1 e P2). As tabelas 4.11 e 4.12 apresentam os dados disponibilizados por Pan et al. em seu artigo [54], os quais são os mesmos utilizados por Reddy et al. em [52].

- A classe P1 é composta pelos crus C1 a C4, enquanto a classe P2 é composta pelos crus C5 a C8;
- Os tanques T1, T6, T7 e T8 e a unidade CDU3 podem processar os petróleos da classe P1, enquanto os demais os petróleos da classe P2;
- Cada unidade de processo tem limites operacionais específicos, podendo ser alimentada por até dois tanques simultaneamente, respeitando-se suas vazões máxima e mínima de carga, bem como limites de concentração de um componente-chave nas cargas (e.g. % enxofre na carga);
- Tanques T2, T3, T4 e T5 podem alimentar simultaneamente as unidades CDU2 e CDU3;

A Tabela 4.13 apresenta as estatíticas e medidas de desempenho para os problemas 6, 7 e 8. Como tanto *hardware* quanto *software* empregados são diferentes, o objetivo desta avaliação não é comparar os tempos computacionais, mas apenas

Tabela 4.10: Caso de Teste 5

Caso	Estrutura	Variáveis	Iteração	F	G
5	Tanques: 6	Controle: 672	Inicial	10^8	10^8
Tempo: 30s.	Navios: 0	Estado: 3055	1	475	480
Inicial.: trivial	Píeres: 0	Não-linear: 2023	2	440	442.7
Horizonte: 120h	Dutos: 1	Linear: 1704	3	420	421.5
	Δ t = 1h		4	360	363
			5	270	270

mostrar que a abordagem não-linear proposta é válida nos problemas testados, convergindo para boas soluções (ótimos locais) em tempo computacional aceitável. Os modelos NLP foram resolvidos no NEOS Server (jobs 1931447, 1931626 e 1931640 no host newton.mcs.anl.gov) e comparados com 3 dos mais eficientes modelos da literatura, conforme relatado por Pan et al. [54]: o modelo heurístico deles - resolvido em um computador Intel Core 2 CPU 2.16GHz, com 1 GB de RAM com o *solver* CPLEX - e os modelos RKS(a) e os modelos MILP RKS(b) de Reddy et al. [51], [52] - resolvidos em uma estação Sun Single Ultra SPARC II 400 MHz, com 2 GB de RAM, também com o *solver* CPLEX. O lucro (10^3 USD) obtido em cada solução foi calculado conforme a fórmula definida em [54]. Cabe ressaltar que as soluções NLP apresentaram lucros menores por não ser o lucro a função-objetivo do modelo não-linear.

Caso de Teste 6

O objetivo do caso de Teste 6 é programar o recebimento de 4 cargas de petróleo em 8 tanques da refinaria ao longo de 3 dias, mantendo as unidades de destilação da refinaria operando continuamente.

- Cada unidade deve processar 300 kbbl (10^3 barris de petróleo) até o final do cenário;
- A refinaria recebe os itens de petróleo de um navio do tipo VLCC ("Very Large Crude Carrier"), que chega ao terminal às 15h: Item 1 (10 kbbl, 100% C2),

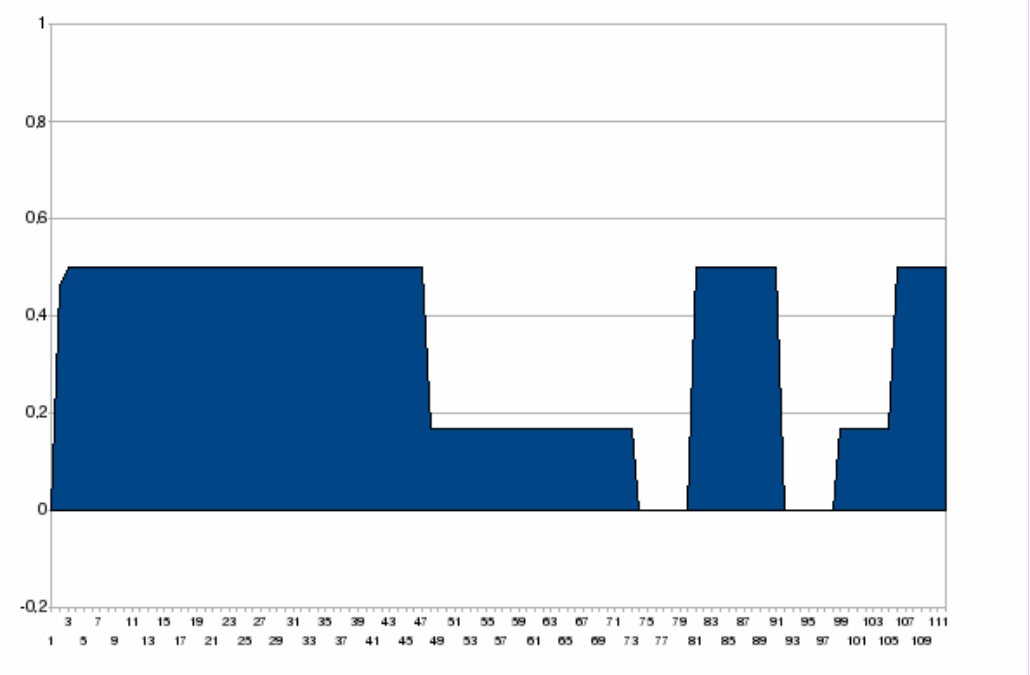

Figura 4.8: Percentagem de Marlim na carga da UDA na solução do Caso 5

Item 2 (250 kbbl, 100% C3), Item 3 (300 kbbl, 100% C4) e Item 4 (190 kbbl, 100% C5);

- O tempo de preparo nos tanques é 8h.
- Cada UDA pode ser alimentada por até dois tanques, com vazão máxima de 6 kbbl/h, desde que a concentração do "componente-chave"na carga (e.g. % enxofre) seja maior que 0.10 e menor que 1.40 (CDU1), 1.30 (CDU2) ou 0.40 (CDU3).

Os resultados computacionais do modelo não linear (NLP) são apresentados na Tabela 4.13 ao lado dos reportados em [54] (Heurística, RKS(a) e RKS(b)). A solução do modelo não linear empregou apenas 7 tanques (tanque 5 ficou ocioso). O tempo foi discretizado em intervalos de 1h. As figuras 4.10, 4.11 e 4.12 apresentam, respectivamente, os volumes dos tanques ao longo cenário, a concentração (% vol) do "componente-chave"na carga de cada UDA e o gráfico de Gantt do *schedule* montado.

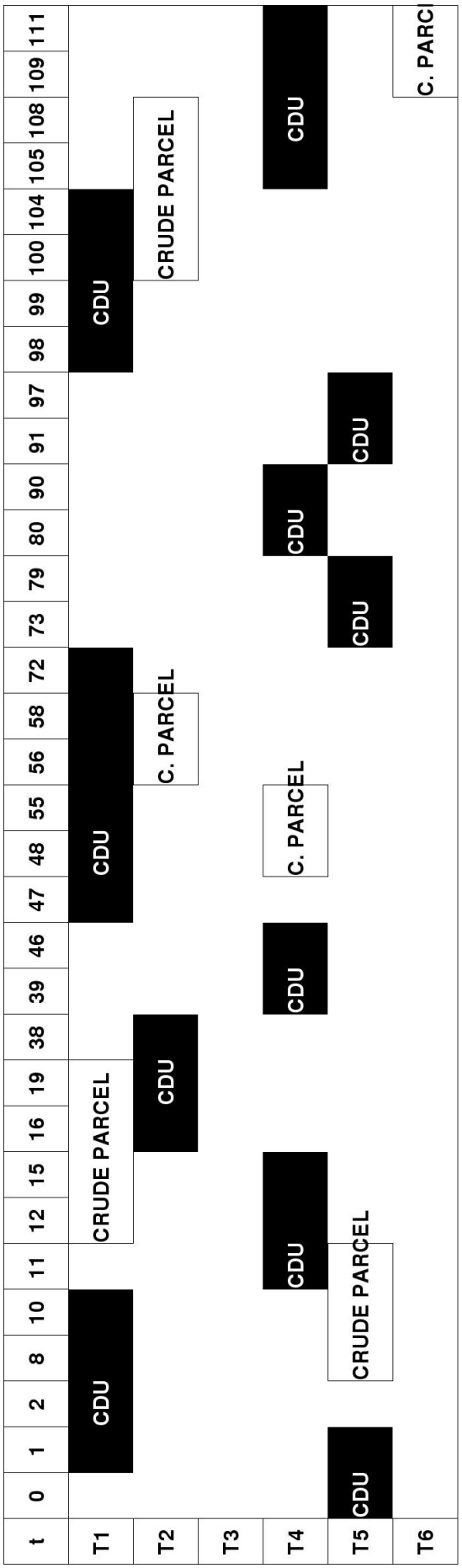

Figura 4.9: Gráfico de Gantt na solução do Caso 5

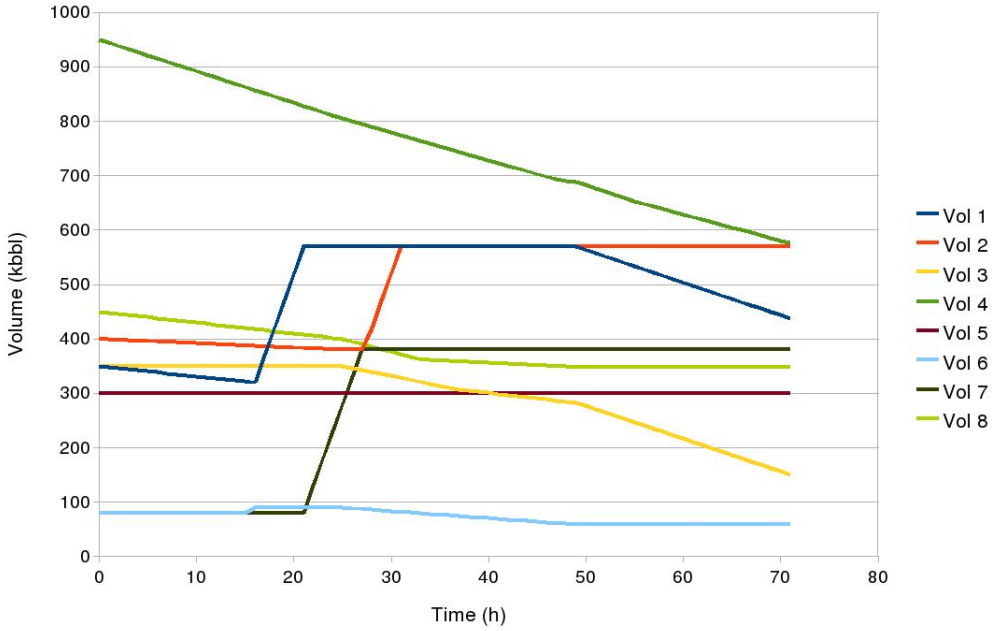

Figura 4.10: Volumes dos tanques na solução do Caso 6

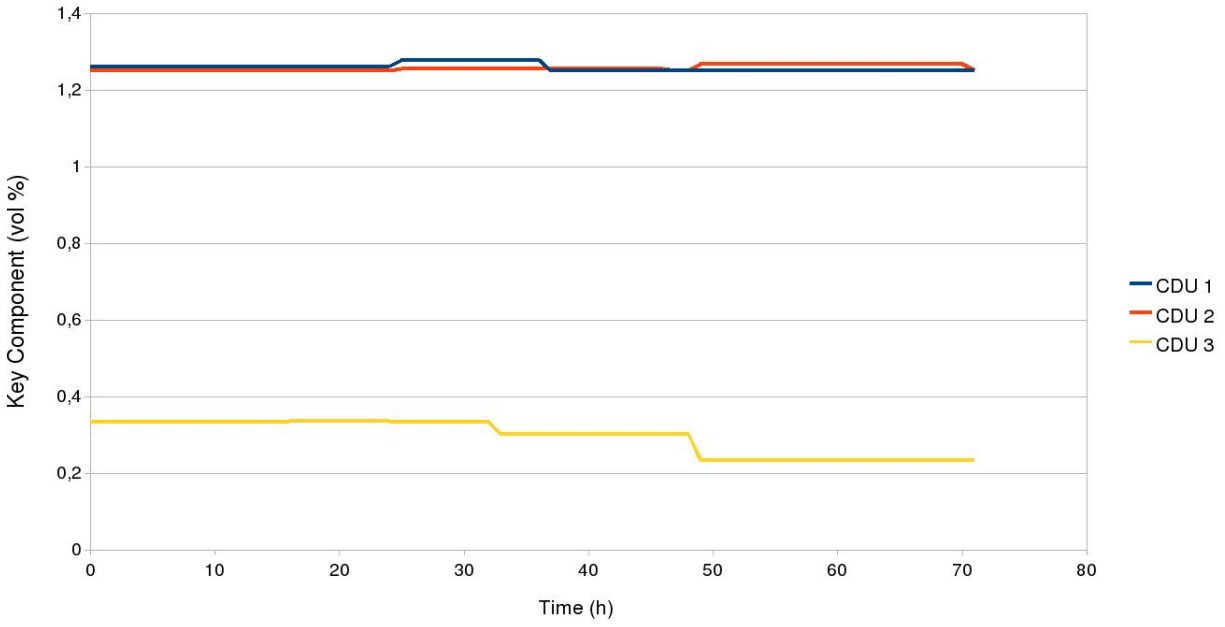

Figura 4.11: Concentração (% vol) do componente chave nas cargas das UDA's no Caso 6

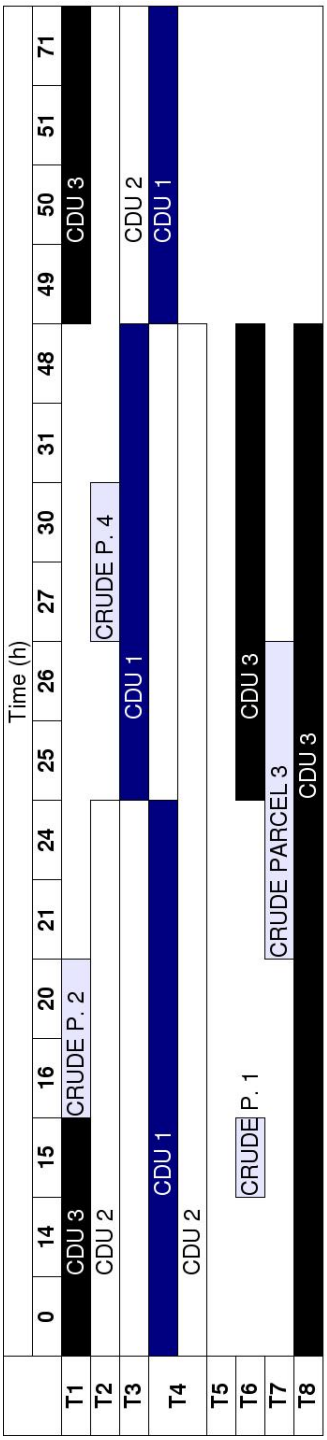

Figura 4.12: Gráfico de Gantt do Caso 6

Tabela 4.11: Especificações de Crus para os Casos de Teste 6, 7 e 8

Cru	Caso de Teste	Componente Chave (% vol.)
C1	6 e 8	0.20
	7	0.25
C2	6 e 8	0.25
	7	0.25
C3	6 e 8	0.15
	7	0.40
C4	6 e 8	0.60
	7	0.20
C5	6 e 8	1.20
	7	1.00
C6	6 e 8	1.30
	7	1.50
C7	6 e 8	0.90
	7	1.40
C8	6 e 8	1.50
	7	1.10

Caso de Teste 7

O objetivo do Caso de Teste 7 é programar o recebimento de 4 cargas de petróleo em 8 tanques da refinaria ao longo de 80h, mantendo as unidades de destilação da refinaria operando continuamente.

- As unidade devem processar até o final do cenário: 375 kbbl (CDU1), 375 kbbl (CDU2) e 400 kbbl (CDU3);
- A refinaria recebe os itens de petróleo de um navio do tipo VLCC ("Very Large Crude Carrier"), que chega ao terminal às 8h: Item 1 (10 kbbl, 100% C2), Item 2 (500 kbbl, 100% C4), Item 3 (500 kbbl, 100% C3) e Item 4 (440 kbbl, 100% C5);
- O tempo de preparo nos tanques é 8h.

Tabela 4.12: Estados iniciais dos tanques nos Casos de Teste 6, 7 e 8

Tanque	Vol.Min.(kbbl)	Vol.Max.(kbbl)	Caso	Vol.Inicial(kbbl)
T1	60	570	6 e 8	C1(50)/C2(100)/C3(100)/C4(100)
			7	C1(50)/C2(150)/C3(150)/C4(50)
T2	60	570	6 e 8	C5(100)/C6(100)/C7(100)/C8(100)
			7	C5(200)/C6(50)/C7(50)/C8(150)
T3	60	570	6 e 8	C5(100)/C6(100)/C7(50)/C8(100)
			7	C5(100)/C6(50)/C7(50)/C8(100)
T4	110	980	6 e 8	C5(200)/C6(250)/C7(200)/C8(300)
			7	C5(200)/C6(200)/C7(200)/C8(300)
T5	110	980	6 e 8	C5(100)/C6(100)/C7(50)/C8(50)
			7	C5(100)/C6(50)/C7(50)/C8(50)
T6	60	570	6 e 8	C1(20)/C2(20)/C3(20)/C4(20)
			7	C1(30)/C2(150)/C3(150)/C4(30)
T7	60	570	6 e 8	C1(20)/C2(20)/C3(20)/C4(20)
			7	C1(30)/C2(50)/C3(50)/C4(10)
T8	60	570	6 e 8	C1(100)/C2(100)/C3(100)/C4(150)
			7	C1(150)/C2(210)/C3(210)/C4(90)

- Cada UDA pode ser alimentada por até dois tanques, com vazão máxima de 6 kbbl/h, desde que a concentração do "componente-chave"na carga (e.g. % enxofre) seja maior que 0.10 e menor que 1.35 (CDU1), 1.20 (CDU2) ou 0.35 (CDU3).

Resultados computacionais são apresentados na Tabela 4.13. O tempo foi discretizado em intervalos de 1h.

Caso de Teste 8

O objetivo do Caso de Teste 8 é programar o recebimento de 8 cargas de petróleo em 8 tanques da refinaria ao longo de 160h, mantendo as unidades de destilação da refinaria operando continuamente.

Tabela 4.13: Estatísticas dos Casos de Teste 6, 7 e 8

Caso	Modelo	CPU(s)	Lucro (10^3 USD)	Var. Bin.	Var. Cont.	Restrições
6	NLP	27	1334.6	0	1734	2808
	Heur.	4.1	1409.3	135	1103	3438
	RKS(a)	68	1409.3	115	1358	2668
	RKS(b)	49	1409.3	136	1235	2323
7	NLP	139	1661.8	0	1995	2210
	Heur.	10.4	1771.7	154	1135	3593
	RKS(a)	815	1766.4	160	1778	3471
	RKS(b)	556	1771.7	162	1395	2601
8	NLP	217	2982.0	0	3834	6083
	Heur.	5.5	3272.5	160	1780	5422
	RKS(a)	3242	3256.8	242	2893	5510
	RKS(b)	3389	3255.3	301	2759	5041

- Cada unidade deve processar 700 kbbl (10^3 barris de petróleo) até o final do cenário;
- A refinaria recebe os itens de petróleo de dois navios do tipo VLCC ("Very Large Crude Carrier"), que chegam ao terminal às 15h e às 100h.
  - VLCC1: Item 1 (10 kbbl, 100% C2), Item 2 (250 kbbl, 100% C3), Item 3 (300 kbbl, 100% C4) e Item 4 (190 kbbl, 100% C5);
  - VLCC2: Item 5 (10 kbbl, 100% C5), Item 6 (250 kbbl, 100% C6), Item 7 (240 kbbl, 100% C8) e Item 8 (240 kbbl, 100% C3);
- O tempo de preparo nos tanques é 8h.
- Cada UDA pode ser alimentada por até dois tanques, com vazão máxima de 6 kbbl/h, desde que a concentração do "componente-chave"na carga (e.g. % enxofre) seja maior que 0.10 e menor que 1.30 (CDU1), 1.25 (CDU2) ou 0.30 (CDU3).

Resultados computacionais são apresentados na Tabela 4.13. A solução do modelo não linear empregou apenas 7 tanques (tanque 5 ficou ocioso). O tempo foi discretizado em intervalos de 1h. As figuras 4.13 e 4.14 apresentam, respectivamente, os volumes dos tanques ao longo cenário e a concentração (% vol) do "componente-chave"na carga de cada UDA.

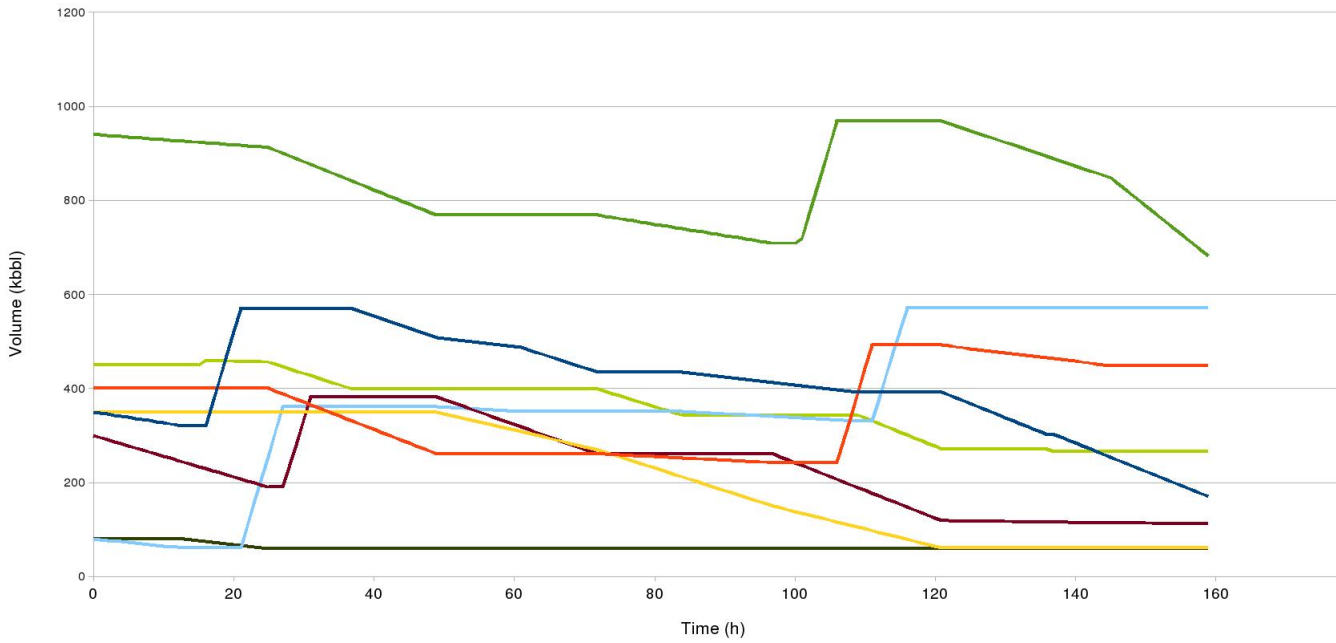

Figura 4.13: Volumes dos tanques na solução do Caso 8

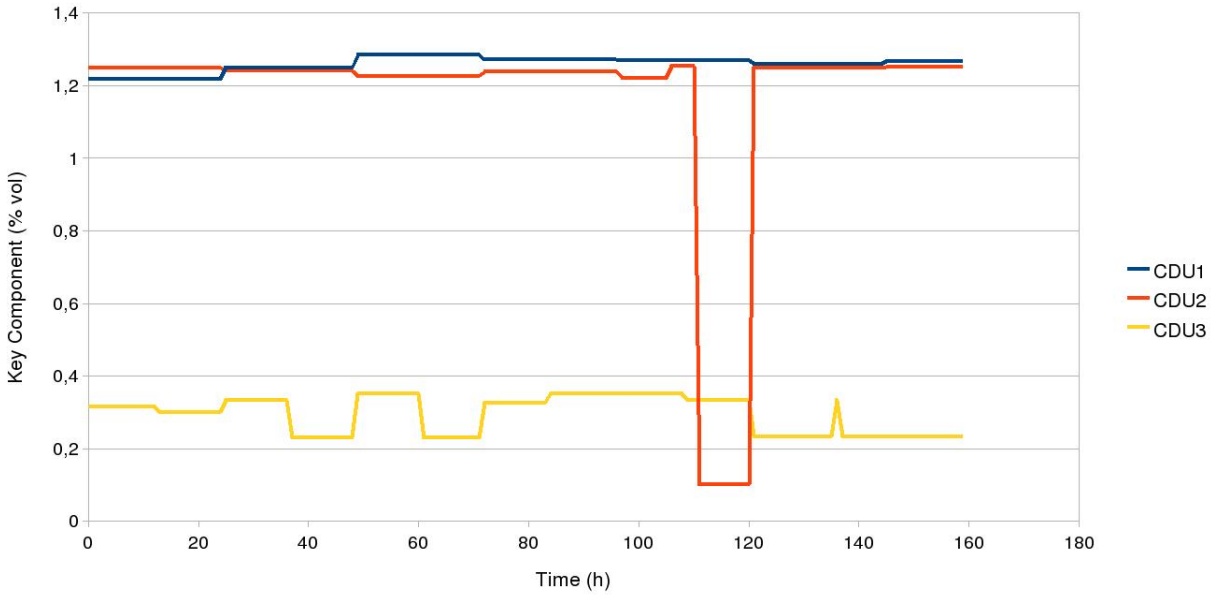

Figura 4.14: Concentração (% vol) do componente chave nas cargas das UDA's no Caso 8

Capítulo 5

Conclusões

5.1 Conclusões

Nesta tese apresentamos uma abordagem original para solução de problemas de *scheduling* de petróleo em portos e refinarias, considerando o sistema logístico como um sistema dinâmico, com diversos estados que variam ao longo do tempo, limitados por restrições (físico-químicas, operacionais e financeiras) e influenciados por variáveis de controle, que nada mais são que as vazões entre os equipamentos. Desta forma, o problema de *scheduling* nada mais é que um problema de controle ótimo, portanto equivalente a um problema de programação não-linear.

Ao contrários de diversos modelos da literatura, que empregam variáveis discretas para representar decisões do tipo sim-não para a alocação das atividades nos equipamentos, o modelo aqui apresentado utiliza apenas variáveis contínuas, conseguindo, por meio de restrições de complementaridade, modelar decisões do tipo "Sim-Não"sem a introdução de variáveis binárias no problema. Desta forma, métodos comuns de otimização não-linear contínua podem ser utilizados para resolver o problema. Ademais, cabe ressaltar que, apesar de testado em problemas de *scheduling* de petróleo, as equações fundamentais do modelo, relacionadas às atividades de transferência de produtos entre dois equipamentos, foram desenvolvidas a partir dos balanços de volume e de massa nos equipamentos, podendo ser utilizadas em outros problemas de *scheduling*, não sendo exclusivas para o problema da logística de petróleo.

Uma desvantagem observada da abordagem apresentada é que o modelo não-

linear é não-convexo e, portanto, suscetível à convergência em ótimos locais. Dentro do *framework* de relaxação e penalização proposto para solução do modelo, observou-se que a escolha do ponto inicial, bem como a evolução dos parâmetros de penalidade, podem influenciar consideralmente a qualidade da solução encontrada.

Os resultados computacionais apresentados neste trabalho, baseados tanto em problemas construídos para a tese, quanto em problemas obtidos na literatura recente, se mostraram encorajadores, pois soluções ótimas ou sub-ótimas foram encontradas em poucos minutos, mesmo para problemas com milhares de variáveis e restrições. Além disso, foi observado que é possível combinar a abordagem aqui apresentada com métodos mais tradicionais de programação mista-inteira, reduzindo potencialmente o número de iterações necessárias para se comprovar a otimalidade de uma solução MILP.

5.2 Sugestões para novos trabalhos

Considerando este um trabalho que introduz o uso de modelos de programação matemática com equações de complementaridade para *scheduling* de petróleo, há diversos pontos que podem ser explorados e evoluídos em trabalhos seguintes à pesquisa aqui apresentada. Seguem algumas sugestões:

- Utilizar métodos de inicialização *multi-start* ou metaheurísticas para gerar diferentes soluções de boa qualidade, procurando obter diferentes ótimos locais;
- Desenvolver procedimentos de redução de variáveis e discretização do tempo, visando menor número de trocas de tanques;
- Estudar relaxações convexas para o problema.

Referências Bibliográficas

[1] GARY, J. H., HANDWERK, G.E., *Petroleum Refining Technology and Economics Fourth Edition*, New York, Marcel Dekker Inc., 2001.

[2] MORO, L.F.L., "Process technology in the petroleum refining industry: current situation and future trends", *Computers & Chemical Engineering*, 27, 8, pp. 1303-1305, 2003.

[3] CHRISTIANSEN, M., FAGERHOLT, K., RONEN, D., "Ship routing and scheduling: status and perspectives", *Transportation Science*, 38, 1, pp. 1-18, 2004.

[4] MÁS, R., PINTO, J.M., "A mixed-integer optimization strategy for oil supply in distribution complexes", *Optimization and Engineering*, 4, 1, pp. 23-64, 2003.

[5] JOLY, M., MORO, L.F.L. e PINTO, J.M., "Planning and scheduling for petroleum refineries using mathemathical programming", *Brazilian Journal of Chemical Engineering*, 19, 2, pp. 207-228, 2002.

[6] WU, N., ZHOU, M.C., CHU, F., "Short-term Scheduling for Refinery Process: Bridging the Gap between Theory and Applications", *International Journal of Intelligent Control and Systems*, 10, 2, pp. 162-174, 2005.

[7] FLOUDAS, C. A., LIN, X., "Mixed integer linear programming in process scheduling: modeling, algorithms, and applications", *Annals of Operations Research*, 139, pp. 131-162, 2005.

[8] BRUCKER, P., KNUST, S. *Complex Scheduling*, Berlin/Heidelberg, Springer-Verlag, 2006.

[9] MORTON, T.E., PENTICO, D.W., *Heuristic Scheduling Systems with Applications to Production Systems and Project Management*, New York, John Wiley & Sons, 1993.

[10] ALLE, A., *Técnicas de programação mista-inteira aplicadas ao scheduling de plantas químicas contínuas*, Tese de D.Sc., Universidade de São Paulo, São Paulo, 2003.

[11] MARAVELIAS, C.T., SUNG, C., "Integration of production planning and scheduling: overview, challenges and opportunities", *Proceedings Foundations of Computer-Aided Process Operations (FOCAPO 2008)*, pp. 13-23, 2008.

[12] NEIRO, S.S.M., PINTO, J.M., "A general modeling framework for the operational planning of petroleum supply chains", *Computers and Chemical Engineering*, 28, pp. 871-896, 2004.

[13] REKLAITIS, G.V., "Overview of scheduling and planning of batch process operations", *Proceedings of the NATO Advanced Study Institute on Batch Processing Systems Engineering: Current Status and Future Directions*, pp. 660-705, 1992.

[14] KONDILI, E., PANTELIDES, C., SARGENT, R., "A general algorithm for short-term scheduling of batch operations - 1. Mixed Integer Linear Programming formulation", *Computers and Chemical Engineering*, Supp. 17, pp. 211-227, 1993.

[15] SCHILLING, G., PANTELIDES, C., "A simple continuous-time process scheduling formulation and a novel solution algorithm", *Computers and Chemical Engineering*, 20, pp. 1221-1226, 1996.

[16] GIMÉNEZ, D.M., HENNING, G.P., "An efficient global-event based continuous-time formulation for the short-term scheduling of multipurpose batch plants", *Computer Aided Chemical Engineering*, 24, pp. 661-667, 2007.

[17] GIMÉNEZ, D.M., HENNING, G.P., MARAVELIAS, C.T., "A novel network-based continuous-time representation for process scheduling: Part I. Main

concepts and mathematical formulation", *Computers and Chemical Engineering*, 33, pp. 1511-1528, 2009.

[18] GIMÉNEZ, D.M., HENNING, G.P., MARAVELIAS, C.T., "A novel network-based continuous-time representation for process scheduling: Part II. General Framework", *Computers and Chemical Engineering*, 33, pp. 1644-1660, 2009.

[19] SHAH, N., "Mathematical Programming Techniques for Crude Oil Scheduling", *Computers and Chemical Engineering*, 20 supl., pp. S1227-S1232, 1996.

[20] AL-YAKKOB, S.M., SHERALI, H.D., "Multiple Shift Scheduling of Hierarchical Workforce with Multiple Work Centers", *Informatica*, Vol. 18, No. 3, pp. 325-342, 2007.

[21] JANIAK, A., KOVALYOV, M.Y., "Job Sequencing with Exponential Functions of Processing Times", *Informatica*, Vol. 17, No. 1, pp. 13-24, 2006.

[22] DZEMYDA, G., "Mean squared load criteria for scheduling independent tasks", *International Journal of Applied Mathematics and Computer Science*, 9, 4, pp. 101-116, 1999.

[23] FLOUDAS, C.A., LIN, X., "Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review", *Computers and Chemical Engineering*, 28, pp. 2109-2129, 2004.

[24] SIMÃO, L.M., PACHECO, M.A.C., VELLASCO, M.M.B.R., "Otimização da mistura de produtos na indústria do petróleo utilizando algoritmos genéticos", *Anais do VI Simpósio Brasileiro de Automação Inteligente*, Bauru, 2003.

[25] ASTM, *ASTM Standard D-2270: Standard Practice for Calculating Viscosity Index from Kinematic Viscosity at 40o and 100o Celsius*, 1991.

[26] ABADIE, J. (1970). Application of the GRG algorithm to Optimal Control problems. In: J. Abadie (Ed.), *Integer and Nonlinear Programming*, North-Holland.

[27] FACÓ, J.L.D. (1990). A Generalized Reduced Gradient Algorithm for Solving Large-scale Discrete-time Nonlinear Optimal Control Problems. In: H.B. Siguerdidjane and P. Bernhard (Ed.), *Control Applications of Nonlinear programming and Optimization*, Pergamon Press: Oxford.

[28] FAGUNDEZ, F.D., FACÓ, J.L.D., XAVIER, A.E., "Continuous Nonlinear Programming Techniques to Solve Scheduling Problems", *Informatica*, 20, pp. 203-216, 2009.

[29] FAGUNDEZ, F.D., FACÓ, J.L.D., XAVIER, A.E. (2010). An optimal control approach to process scheduling. In: *Optimization and Optimal Control: Theory and Applications*, A. Chinchuluun, P.M. Pardalos, R. Enkhbat e I. Tseveendorj (Ed.), v.39, Springer, p.409-422.

[30] FAGUNDEZ, F.D., FACÓ, J.L.D., XAVIER, A.E., "A novel nonlinear optimal control approach for crude oil and derivatives scheduling", *Journal of Scheduling*, Springer, submetido em 2010.

[31] COLLYER, W., "Sobreestadia de navios: a regra 'once on demurrage, always on demurrage'", *Jus Navigandi*, 1166, 2006. Web: http://jus2.uol.com.br/doutrina/texto.asp?id=8889 (Último acesso em Julho de 2008).

[32] STEPHENSON JR., F.J., *Transportation U.S.A*, Reading, Addison-Wesley, 1987.

[33] MAGALHÃES, M.V., SHAH, N., "Crude Oil Scheduling", *Proceedings Foundations of Computer-Aided Process Operations (FOCAPO 2003)*, pp. 323-326, 2003.

[34] DUARTE, D.A.L., *Aplicação da modelagem de um sistema de apoio à decisão para o planejamento das operações logísticas de produtos especiais*, Dissertação de M.Sc., Universidade Federal de Santa Catarina, Florianopólis, 2002.

[35] PAOLUCCI, M., SACILE, R., BOCCALATTE, A., "Allocating crude oil supply to port and refinery tanks: a simulation-based decision support system", *Decision Support Systems*, 33 supp., pp. 39-54, 2002.

[36] LI, J., KARIMI, I.A., SRINIVASAN, R., "Robust Scheduling of Crude Oil Operations under Demand and Ship Arrival Uncertainty", *AIChE Annual Meeting Abstracts*, paper 662f, 2006.

[37] KARRI, B., SRINIVASAN, R., KARIMI, I.A., "Measures and approaches for a priori analysis of schedule robustness", *AIChE Annual Meeting Abstracts*, paper 372d, 2007.

[38] LEE, H., PINTO, J.M., GROSSMANN, I.E., PARK, S., "Mixed-integer linear programming model for refinery short term scheduling of crude oil unloading with inventory management", *Industrial & Engineering Chemistry Research*, 35, pp. 1630-1641, 1996.

[39] SYMONDS, G.H., "Linear Programming Solves Gasoline Refining and Blending Problems", *Industrial and Engineering Chemistry*, 48, pp. 394-401, 1956.

[40] MANNE, A.S., *Scheduling of Petroleum Refinery Operations*, Cambridge, Harvard University Press, 1956.

[41] FOSTER, B.L., "Maximization in the Oil Industry-A Survey", *Management Technology*, 4, 1, pp. 26-46, 1964.

[42] GRIFFITH, R.E., STEWART, R.A., "A Nonlinear Programming Technique for the Optimization of Continuous Processing Systems", *Management Science*, 7, pp. 379-392, 1961.

[43] BAKER, T.E., LASDON, L.S., "Successive Linear Programming at Exxon", *Management Science*, 31, 3, pp. 264-274, 1985.

[44] WITT, C.W.D., LASON, L.S., WAREN, A.D., BRENNER, D.A., MELHEM, S.A., "OMEGA: An Improved Gasoline Blending System for TEXACO", *Interfaces*, 19, pp. 85-101, 1989.

[45] ZHANG, J., KIM, N.H., LASDON, L.S., "An Improved Successive Linear Programming Algorithm", *Management Science*, 31, 10, 1312-1331, 1985.

[46] MCFARLAND, J.W., LASDON, L.S., LOOSE, V., "Development Planning and Management of Petroleum Reservoirs Using Tank Models and Nonlinear Programming", *Operations Research*, 32, 2, pp. 270-289, 1984.

[47] WU, N.Q., CHU, F., CHU, C.B., ZHOU, M.C., "Schedulability analysis of short-term schedule for crude oil operations using Petri nets", *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pp. 3481-3486, 2007.

[48] WU, N.Q., CHU, F., CHU, C.B., ZHOU, M.C., "Short-term schedulability analysis of crude oil operations in refinery with oil residency time constraint using Petri net", textitIEEE Transactions on Systems, Man, and Cybernetics, Part C, 38, 6, pp. 765-778, 2008.

[49] WU, N.Q., CHU, F., ZHOU, M.C., "A Petri net based heuristic algorithm for realizability of target refining schedule for oil refinery", textitIEEE Transactions on Automation Science and Engineering, 5, 4, pp. 661-676, 2008.

[50] WU, N.Q., CHU, F., CHU, C.B., ZHOU, M.C., "Short-term schedulability analysis of multiple distiller crude oil operations in refinery with oil residency time constraint", *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 39, 1, pp. 1-16, 2009.

[51] REDDY, P.C.P., KARIMI, I.A., SRINIVASAN, R., "A New Continuous Time Formulation for Scheduling Crude Oil Operations", *Chemical Engineering Science*, 59, 6, pp. 1325-1341, 2004.

[52] REDDY, P.C.P., KARIMI, I.A., SRINIVASAN, R., "A Novel Solution Approach for Optimizing Scheduling Crude Oil Operations", *AIChE Journal*, 50(6), 1177-1197, 2004.

[53] LI, J., LI, W., KARIMI, I.A., SRINIVASAN, R., "Improving the Robustness and Efficiency of Crude Scheduling Algorithms", *AIChE Journal*, 53 (10), pp. 2659-2680, 2007.

[54] PAN, M., LI, X., QIAN, Y., "New approach for scheduling crude oil operations", *Chemical Engineering Science*, 64, pp. 965-983, 2009.

[55] NEIRO, S.M.S., PINTO, J.M., "Langrangean decomposition applied to multi-period planning of petroleum refineries under uncertainty", *Latin American Applied Research*, 36, 4, pp. 213-220, 2006.

[56] LI, Z., IERAPETRITOU, M.G., "Robust Scheduling Optimization", *Proceedings Foundations of Computer-Aided Process Operations (FOCAPO 2008)*, pp. 425-428, 2008.

[57] PINTO, J.M., MORO, L.F.L., "A planning model for petroleum refineries", *Brazilian Journal of Chemical Engineering* 17, 4-7, pp. 575-586, 2000.

[58] MORO, L.F.L., PINTO, J.M., "Mixed-Integer Programming Approach for Short-Term Crude Oil Scheduling", *Industrial Engineering and Chemistry Research*, 43, pp. 85-94, 2004.

[59] LEE, T., RYU, J., LEE, H.K., LEE, I.B., "Development of an Integrated Naphta Feeding Framework Simultaneously Considering Vessel Scheduling and Storage Tank Management", *Proceedings Foundations of Computer-Aided Process Operations (FOCAPO 2008)*, pp. 429-432, 2008.

[60] BLANCO, A.M., DIAZ, A.B.M., ANGELES, A.R., SÁNCHEZ, A., "Operational Planning of Crude Oil Processing Terminals", *European Symposium on Computer Aided Process Engineering*, L. Puigjaner & A. Espuña (ed.), Elsevier, 2005.

[61] LIN, X., CHAJAKIS, E.D., FLOUDAS, C.A., "Scheduling of Tanker Lightering via a Novel Continuous-Time", *Industrial and Engineering Chemistry Research*, 42, pp. 4441-4451, 2003.

[62] VAN ASPEREN, E., DEKKER, R., POLMAN, M., ARONS, H.S., WALTMAN, L., "Arrival processes for vessels in a port simulation", *ERIM Report Series Research in Management ERS-2003-067-LIS*, Erasmus Research Institute of Management, 1-15, 2003.

[63] DAI, J., LIN, W., MOORTHY, R., TEO, C.P., "Berth Allocation Planning Optimization in Container Terminals", Relatório Técnico, Georgia Institute of Technology/University of Singapore, 2004.

[64] RONEN, D., "Cargo Ships Routing and Scheduling: A survey of models and problems", *European Journal of Operations Research*, 12, 119, 1983.

[65] RONEN, D., "Ship Scheduling: the last decade", *European Journal of Operations Research*, 71, 325, 1993.

[66] FLOOD, M.F., "Application of Transportation Theory to Scheduling a Military Tanker Fleet", *Operations Research*, 1, pp. 150-162, 1954.

[67] DANTZIG, G.B., FULKERSON, D.R., "Minimizing the number of tankers to meet a fixed schedule", *Naval Research Logistics Quarterly*, 1, pp. 217-222, 1954.

[68] BROWN, G.G., GRAVES, G.W., RONEN, D., "Scheduling Ocean Transportation of Crude Oil", *Management Science*, 33, 3, pp. 335-346, 1987.

[69] SHERALI, H.D., AL-YAKKOB, S.M., HASSAN, M.M., "Fleet Management Model and Algorithms for an oil tanker routing and scheduling problem", *IEE Transactions*, 31, pp. 395-406, 1999.

[70] IAKOVOU, E., DOULIGERIS, C., "Strategic Transportation Model for Oil in US Waters", *Computers and Industrial Engineering*, 31, 1/2, pp. 59-62, 1996.

[71] HWANG, S.J., *Inventory Constrained Maritime Routing and Scheduling for Multi-Commodity Liquid Bulk*, Ph.D. Thesis, Georgia Institute of Technology, Atlanta, 2005.

[72] LI, J., KARIMI, I.A., SRINIVASAN, R., "Supply and Distribution of Multiple Products via Bulk Maritime Logistics", *Proceedings Foundations of Computer-Aided Process Operations (FOCAPO 2008)*, pp. 497-500, 2008.

[73] FAGUNDEZ, F.D., *Modelos de Controle Ótimo do Scheduling de Petróleo e Derivados em Portos*, Dissertação de M.Sc., Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2005.

[74] PARDALOS, P.M., PROKOPYEV, O.A., BUSYGIN, S. (2006). Continuous Approaches for Solving Discrete Optimization Problems, in *Handbook on Modeling for Discrete Optimization*, G. Appa, L. Pitsoulis, H. P. Williams (ed.). Springer-Verlag: Berlin Heidelberg.

[75] XAVIER, A.E., "Hyperbolic Penalty: a new method for nonlinear programming with inequalities", *International Transactions in Operational Research*, 8, pp. 659-671, 2001.

[76] RUI, S.P., XU, C.X., "A smoothing inexact Newton method for nonlinear complementarity problems", *Journal of Computational and Applied Mathematics*, 233, pp. 2332-2338, 2010.

[77] LEYFFER, S. (2006). Complementarity Constraints as Nonlinear Equations: Theory and Numerical Experience, in *Optimization with Multivalued Mappings*, S. Dempe, V. Kalashnikov (eds.), Springer, 169-208.

[78] FLETCHER, R., LEYFFER, S., RALPH, D., SCHOLTES, S. (2001), Local Convergence of SQP Methods for Mathematical Programs with Equilibrium Constraints, Numerical Analysis Report, Department of Mathematics, University of Dundee, Dundee.

[79] ANITESCU, M. (2000). On solving mathematical programs with complementarity constraints as nonlinear programs. Preprint ANL/MCS-P864-1200, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne.

[80] BADAKHSHAN, K.P., KAMYAD, A.V., "Numerical solution of nonlinear optimal control problems using nonlinear programming", *Applied Mathematics and Computation*, 187, pp. 1511-1519, 2007.

[81] TABAK, D., "Applications of Mathematical Programming Techniques in Optimal Control: A Survey", JACC, Atlanta, 1970.

[82] LIN, G.H., FUKUSHIMA, M., "New Relaxation Method for Mathematical Programs with Complementarity Constraints", *Journal of Optimization Theory and Applications*, 118, 1, 81-116, 2003.

[83] HUANG, C., WANG, S., "A power penalty approach to a Nonlinear Complementarity Problem", *Operations Research Letters*, 38, pp. 72-76, 2010.

[84] SMITH, S., LASDON, L.S., "Solve Large-Sparse Nonlinear Programs using GRG", *ORSA Journal of Computing*, 4, pp. 1-15, 1992.

[85] FRONTLINE SYSTEMS, *Premium Solver Platform Solver DLL Platform Field-Installable Solver Engines User Guide*, v. 6.0, 2006. Web: http://www.solver.com (Último acesso: Dezembro 2006).

[86] FOURER, R., GAY, D.M., KERNIGHAN, B.W., *AMPL: A Modeling Language for Mathematical Programming*, 3rd ed., Pacific Grove, Duxbury Press Brooks/Cole-Thomson, 2003.

[87] ILOG INC., *ILOG AMPL/CPLEX System Version 8.0 User's Guide*, New York, ILOG, 2002.

[88] GILL, P. E., MURRAY, W., SAUNDERS, M.A., "SNOPT: An SQP algorithm for large-scale constrained optimization", *SIAM Journal on Optimization*, 12, 4, pp. 979-1006, 2002.

[89] MURTAGH, B.A., SAUNDERS, M.A., "A projected Lagrangian algorithm and its implementation for sparse non-linear constraints", *Mathematical Programming Studies*, 16, pp. 84-117, 1982.

[90] DOLAN, E.D., FOURER, R., MORE, J.J., MUNSON, T.S., "The NEOS Server for Optimization: Version 4 and Beyond", Mathematics and

Computer Science Division, Argonne National Laboratory, 2002. Web: $ftp : //info.mcs.anl.gov/pub/tech_reports/reports/P947.pdf$ (Último acesso em Novembro de 2008)

[91] BROOKE, A., KENDRICK, D., MEERAUS, A., *GAMS: A User's Guide*, San Francisco, The Scientific Press, 1988.

[92] VISWANATHAN, J., GROSSMANN, I.E., "A Combined Penalty Function and Outer Approximation Method for MINLP Optimization", *Computers and Chemical Engineering*,14, pp. 762-775, 1990.

Apêndice A

Modelos AMPL

A.1 Modelo do Caso de Teste 5 (Moro e Pinto, 2004)

```
#####################################################
# REVAP Model
# Based on the problem described in
# Moro and Pinto
# Ind. Eng. Chem. Res., 2004, 43, pp.85-94
#####################################################
#####################################################
# Parameters
#####################################################
param dt;
let dt := 1.0;
param T = 112.0/dt;
# Max Tank volume (m3)
param max_vol = 80.000;
# Min Tank volume (m3)
param min_vol = 13.000;
# Initial volumes
param ini_vol1 = 40.000; # ini vol of tank1
param ini_vol2 = 50.000; # ini vol of tank2
param ini_vol3 = 15.000; # ini vol of tank3
```

```
param ini_vol4 = 50.000; # ini vol of tank4
param ini_vol5 = 20.000; # ini vol of tank5
param ini_vol6 = 15.000; # ini vol of tank6
# Initial composition (3 crudes: Bonito (1), Marlin (2), RGN(3))
param ini_cru11 = 0.50; # crude 1 - tank 1
param ini_cru21 = 0.50; # crude 2 - tank 1
param ini_cru31 = 0; # crude 3 - tank 1
param ini_cru12 = 0; # crude 1 - tank 2
param ini_cru22 = 1.00; # crude 2 - tank 2
param ini_cru32 = 0; # crude 3 - tank 2
param ini_cru13 = 0.70; # crude 1 - tank 3
param ini_cru23 = 0; # crude 2 - tank 3
param ini_cru33 = 0.30; # crude 3 - tank 3
param ini_cru14 = 0; # crude 1 - tank 4
param ini_cru24 = 1.00; # crude 2 - tank 4
param ini_cru34 = 0; # crude 3 - tank 4
param ini_cru15 = 0.60; # crude 1 - tank 5
param ini_cru25 = 0; # crude 2 - tank 5
param ini_cru35 = 0.40; # crude 3 - tank 5
param ini_cru16 = 0.60; # crude 1 - tank 6
param ini_cru26 = 0.30; # crude 2 - tank 6
param ini_cru36 = 0.10; # crude 3 - tank 6
# Ship final volume
param final_parcelvol = 0.0;
#arrival crude parcels
param arrival1 = ceil(8/dt);     # t = 8h
param arrival2 = ceil(48/dt);  # t = 48h
param arrival3 = ceil(58/dt);  # t = 58h
param arrival4 = ceil(100/dt);  # t = 100h
#end crude parcels
param end1 = max(ceil(20/dt), arrival1 + 1);    # t = 20h
param end2 = max(ceil(58/dt), arrival2 + 1);  # t = 58h
param end3 = max(ceil(58.2/dt), arrival3 + 1);  # t = 58.2h -> 60
param end4 = max(ceil(112/dt), arrival4 + 1);  # t = 112h
```

```
#crude parcels
param crude1 = 0.00;    # 100% Bonito (1)
param crude2 = 1.00;  # 100% Marlin (2)
param crude3 = 1.00;  # 100% Marlin (2)
param crude4 = 0.00;  # 100% RGN (3)
# max pipeline flow 5000 m^3/h
param q_max_pipe = 5.000;
# parcel volumes
param ini_parcel1 = 60.000;     # 60000 m3
param ini_parcel2 = 50.000;  # 50000 m3
param ini_parcel3 = 1.000;  # 1000 m3
param ini_parcel4 = 60.000;  # 60000 m3
# resting/settling time
param settling_time = ceil(24/dt); # 24h to settle brine
#CDU feed flow 1500 m^3/h
param feedflow = 1.500;
param cdu_vol = feedflow*T*dt;
#min tank flow
param min_q = 0;
# penalty variables
var twocduflow_1 {0..T-1} ;
var twocduflow_2 {0..T-1} ;
var twocduflow_3 {0..T-1} ;
var twocduflow_4 {0..T-1} ;
var twocduflow_5 {0..T-1} ;
var settling_1 {1..T-1} ;
var settling_2 {1..T-1} ;
var settling_3 {1..T-1} ;
var settling_4 {1..T-1} ;
var settling_5 {1..T-1} ;
var settling_6 {1..T-1} ;
var onepipeflow_1 {0..T-1} ;
var onepipeflow_2 {0..T-1} ;
var onepipeflow_3 {0..T-1} ;
```

```
var onepipeflow_4 {0..T-1} ;
var onepipeflow_5 {0..T-1} ;

########################################
# Control variables
########################################
# transferred volume from pipeline to TX
var q11 {0..T-1} <= q_max_pipe, := 0;
var q12 {0..T-1} <= q_max_pipe, := 0;
var q13 {0..T-1} <= q_max_pipe, := 0;
var q14 {0..T-1} <= q_max_pipe, := 0;
var q15 {0..T-1} <= q_max_pipe, := 0;
var q16 {0..T-1} <= q_max_pipe, := 0;
subject to MIN_Q11 {t in 0..T-1}: q11[t]  >= min_q;
subject to MIN_Q12 {t in 0..T-1}: q12[t]  >= min_q;
subject to MIN_Q13 {t in 0..T-1}: q13[t]  >= min_q;
subject to MIN_Q14 {t in 0..T-1}: q14[t]  >= min_q;
subject to MIN_Q15 {t in 0..T-1}: q15[t]  >= min_q;
subject to MIN_Q16 {t in 0..T-1}: q16[t]  >= min_q;
# transferred volume from TX to CDU
var q1 {0..T-1} <= feedflow, := 0;
var q2 {0..T-1} <= feedflow, := 0;
var q3 {0..T-1} <= feedflow, := 0;
var q4 {0..T-1} <= feedflow, := 0;
var q5 {0..T-1} <= feedflow, := 0;
var q6 {0..T-1} <= feedflow, := 0;
subject to MIN_Q1 {t in 0..T-1}: q1[t]  >= min_q;
subject to MIN_Q2 {t in 0..T-1}: q2[t]  >= min_q;
subject to MIN_Q3 {t in 0..T-1}: q3[t]  >= min_q;
subject to MIN_Q4 {t in 0..T-1}: q4[t]  >= min_q;
subject to MIN_Q5 {t in 0..T-1}: q5[t]  >= min_q;
subject to MIN_Q6 {t in 0..T-1}: q6[t]  >= min_q;

################################################
```

```
# State Variables
#####################################################
var vol1 {1..T} <= max_vol, := ini_vol1;
var vol2 {1..T} <= max_vol, := ini_vol2;
var vol3 {1..T} <= max_vol, := ini_vol3;
var vol4 {1..T} <= max_vol, := ini_vol4;
var vol5 {1..T} <= max_vol, := ini_vol5;
var vol6 {1..T} <= max_vol, := ini_vol6;
subject to MIN_VOL1 {t in 1..T}: vol1[t]  >= min_vol;
subject to MIN_VOL2 {t in 1..T}: vol2[t]  >= min_vol;
subject to MIN_VOL3 {t in 1..T}: vol3[t]  >= min_vol;
subject to MIN_VOL4 {t in 1..T}: vol4[t]  >= min_vol;
subject to MIN_VOL5 {t in 1..T}: vol5[t]  >= min_vol;
subject to MIN_VOL6 {t in 1..T}: vol6[t]  >= min_vol;
var crude21 {1..T} := ini_cru21;
var crude22 {1..T} := ini_cru22;
var crude23 {1..T} := ini_cru23;
var crude24 {1..T} := ini_cru24;
var crude25 {1..T} := ini_cru25;
var crude26 {1..T} := ini_cru26;
var parcel1 {arrival1..end1} <= ini_parcel1, := ini_parcel1;
var parcel2 {arrival2..end2} <= ini_parcel2, := ini_parcel2;
var parcel3 {arrival3..end3} <= ini_parcel3, := ini_parcel3;
var parcel4 {arrival4..end4} <= ini_parcel4, := ini_parcel4;
subject to MIN_PARCEL21 {t in arrival1..end1}: parcel1[t]  >= final_parcelvol;
subject to MIN_PARCEL22 {t in arrival2..end2}: parcel2[t]  >= final_parcelvol;
subject to MIN_PARCEL23 {t in arrival3..end3}: parcel3[t]  >= final_parcelvol;
subject to MIN_PARCEL24 {t in arrival4..end4}: parcel4[t]  >= final_parcelvol;

#####################################################
# State Equations - volumes
#####################################################
subject to INITIAL_STATE1: vol1[1] = ini_vol1 + dt*(q11[0] - q1[0]);
subject to STATE_EQ1 {t in 2..T}: vol1[t] = vol1[t-1] + dt*(q11[t-1] - q1[t-1]);
```

```
subject to INITIAL_STATE2: vol2[1] = ini_vol2 + dt*(q12[0] - q2[0]);
subject to STATE_EQ2 {t in 2..T}: vol2[t] = vol2[t-1] + dt*(q12[t-1] - q2[t-1]);
subject to INITIAL_STATE3: vol3[1] = ini_vol3 + dt*(q13[0] - q3[0]);
subject to STATE_EQ3 {t in 2..T}: vol3[t] = vol3[t-1] + dt*(q13[t-1] - q3[t-1]);
subject to INITIAL_STATE4: vol4[1] = ini_vol4 + dt*(q14[0] - q4[0]);
subject to STATE_EQ4 {t in 2..T}: vol4[t] = vol4[t-1] + dt*(q14[t-1] - q4[t-1]);
subject to INITIAL_STATE5: vol5[1] = ini_vol5 + dt*(q15[0] - q5[0]);
subject to STATE_EQ5 {t in 2..T}: vol5[t] = vol5[t-1] + dt*(q15[t-1] - q5[t-1]);
subject to INITIAL_STATE6: vol6[1] = ini_vol6 + dt*(q16[0] - q6[0]);
subject to STATE_EQ6 {t in 2..T}: vol6[t] = vol6[t-1] + dt*(q16[t-1] - q6[t-1]);

##############################################
# State Equations - Crude parcel volumes
##############################################
subject to INITIAL_PARCEL1: parcel1[arrival1] = ini_parcel1;
subject to STATE_PARCEL1 {t in (arrival1 + 1)..(end1)}:
parcel1[t]=parcel1[t-1] -dt*(q11[t-1]+q12[t-1]+q13[t-1]+q14[t-1]+q15[t-1]+q16[t-1]);
subject to FINAL_PARCEL_1: parcel1[end1] = final_parcelvol;
subject to INITIAL_PARCEL2: parcel2[arrival2] = ini_parcel2;
subject to STATE_PARCEL2 {t in (arrival2 + 1)..(end2)}:
parcel2[t]=parcel2[t-1] -dt*(q11[t-1]+q12[t-1]+q13[t-1]+q14[t-1]+q15[t-1]+q16[t-1]);
subject to FINAL_PARCEL_2: parcel2[end2] = final_parcelvol;
subject to INITIAL_PARCEL3: parcel3[arrival3] = ini_parcel3;
subject to STATE_PARCEL3 {t in (arrival3 + 1)..(end3)}:
parcel3[t]=parcel3[t-1] -dt*(q11[t-1]+q12[t-1]+q13[t-1]+q14[t-1]+q15[t-1]+q16[t-1]);
subject to FINAL_PARCEL_3: parcel3[end3] = final_parcelvol;
subject to INITIAL_PARCEL4: parcel4[arrival4] = ini_parcel4;
subject to STATE_PARCEL4 {t in (arrival4 + 1)..(end4)}:
parcel4[t]=parcel4[t-1] -dt*(q11[t-1]+q12[t-1]+q13[t-1]+q14[t-1]+q15[t-1]+q16[t-1]);
subject to FINAL_PARCEL_4: parcel4[end4] = final_parcelvol;

#############################################################################
# State Equations - composition
# Crude X at Tank Y = initial crude x + crude x from pipeline - crude x to CDU
```

```
#############################################################################
# parcel 1: Crude 1 (Bonito) 100%
subject to INITIAL_COMP2_T1 {t in 1..arrival1}: crude21[t] = ini_cru21;
subject to INITIAL_COMP2_T2 {t in 1..arrival1}: crude22[t] = ini_cru22;
subject to INITIAL_COMP2_T3 {t in 1..arrival1}: crude23[t] = ini_cru23;
subject to INITIAL_COMP2_T4 {t in 1..arrival1}: crude24[t] = ini_cru24;
subject to INITIAL_COMP2_T5 {t in 1..arrival1}: crude25[t] = ini_cru25;
subject to INITIAL_COMP2_T6 {t in 1..arrival1}: crude26[t] = ini_cru26;
subject to COMP2_T1_1 {t in (arrival1 + 1)..end1}:
crude21[t] = crude21[t-1] + (crude1 - crude21[t-1])*q11[t-1]/vol1[t-1];
subject to COMP2_T2_1 {t in (arrival1 + 1)..end1}:
crude22[t] = crude22[t-1] + (crude1 - crude22[t-1])*q12[t-1]/vol2[t-1];
subject to COMP2_T3_1 {t in (arrival1 + 1)..end1}:
crude23[t] = crude23[t-1] + (crude1 - crude23[t-1])*q13[t-1]/vol3[t-1];
subject to COMP2_T4_1 {t in (arrival1 + 1)..end1}:
crude24[t] = crude24[t-1] + (crude1 - crude24[t-1])*q14[t-1]/vol4[t-1];
subject to COMP2_T5_1 {t in (arrival1 + 1)..end1}:
crude25[t] = crude25[t-1] + (crude1 - crude25[t-1])*q15[t-1]/vol5[t-1];
subject to COMP2_T6_1 {t in (arrival1 + 1)..end1}:
crude26[t] = crude26[t-1] + (crude1 - crude26[t-1])*q16[t-1]/vol6[t-1];
subject to _COMP2_T1 {t in (end1 + 1)..arrival2}: crude21[t] = crude21[end1];
subject to _COMP2_T2 {t in (end1 + 1)..arrival2}: crude22[t] = crude22[end1];
subject to _COMP2_T3 {t in (end1 + 1)..arrival2}: crude23[t] = crude23[end1];
subject to _COMP2_T4 {t in (end1 + 1)..arrival2}: crude24[t] = crude24[end1];
subject to _COMP2_T5 {t in (end1 + 1)..arrival2}: crude25[t] = crude25[end1];
subject to _COMP2_T6 {t in (end1 + 1)..arrival2}: crude26[t] = crude26[end1];

# parcel 2: Crude 2 (Marlin) 100%
subject to COMP2_T1_2 {t in (arrival2 + 1)..end2}:
crude21[t] = crude21[t-1] + (crude2 - crude21[t-1])*q11[t-1]/vol1[t-1];
subject to COMP2_T2_2 {t in (arrival2 + 1)..end2}:
crude22[t] = crude22[t-1] + (crude2 - crude22[t-1])*q12[t-1]/vol2[t-1];
subject to COMP2_T3_2 {t in (arrival2 + 1)..end2}:
crude23[t] = crude23[t-1] + (crude2 - crude23[t-1])*q13[t-1]/vol3[t-1];
```

```
subject to COMP2_T4_2 {t in (arrival2 + 1)..end2}:
crude24[t] = crude24[t-1] + (crude2 - crude24[t-1])*q14[t-1]/vol4[t-1];
subject to COMP2_T5_2 {t in (arrival2 + 1)..end2}:
crude25[t] = crude25[t-1] + (crude2 - crude25[t-1])*q15[t-1]/vol5[t-1];
subject to COMP2_T6_2 {t in (arrival2 + 1)..end2}:
crude26[t] = crude26[t-1] + (crude2 - crude26[t-1])*q16[t-1]/vol6[t-1];
subject to 2_COMP2_T1 {t in (end2 + 1)..arrival3}: crude21[t] = crude21[end2];
subject to 2_COMP2_T2 {t in (end2 + 1)..arrival3}: crude22[t] = crude22[end2];
subject to 2_COMP2_T3 {t in (end2 + 1)..arrival3}: crude23[t] = crude23[end2];
subject to 2_COMP2_T4 {t in (end2 + 1)..arrival3}: crude24[t] = crude24[end2];
subject to 2_COMP2_T5 {t in (end2 + 1)..arrival3}: crude25[t] = crude25[end2];
subject to 2_COMP2_T6 {t in (end2 + 1)..arrival3}: crude26[t] = crude26[end2];

# parcel 3: Crude 2 (Marlin) 100%
subject to COMP2_T1_3 {t in (arrival3 + 1)..end3}:
crude21[t] = crude21[t-1] + (crude3 - crude21[t-1])*q11[t-1]/vol1[t-1];
subject to COMP2_T2_3 {t in (arrival3 + 1)..end3}:
crude22[t] = crude22[t-1] + (crude3 - crude22[t-1])*q12[t-1]/vol2[t-1];
subject to COMP2_T3_3 {t in (arrival3 + 1)..end3}:
crude23[t] = crude23[t-1] + (crude3 - crude23[t-1])*q13[t-1]/vol3[t-1];
subject to COMP2_T4_3 {t in (arrival3 + 1)..end3}:
crude24[t] = crude24[t-1] + (crude3 - crude24[t-1])*q14[t-1]/vol4[t-1];
subject to COMP2_T5_3 {t in (arrival3 + 1)..end3}:
crude25[t] = crude25[t-1] + (crude3 - crude25[t-1])*q15[t-1]/vol5[t-1];
subject to COMP2_T6_3 {t in (arrival3 + 1)..end3}:
crude26[t] = crude26[t-1] + (crude3 - crude26[t-1])*q16[t-1]/vol6[t-1];
subject to 3_COMP2_T1 {t in (end3 + 1)..arrival4}: crude21[t] = crude21[end3];
subject to 3_COMP2_T2 {t in (end3 + 1)..arrival4}: crude22[t] = crude22[end3];
subject to 3_COMP2_T3 {t in (end3 + 1)..arrival4}: crude23[t] = crude23[end3];
subject to 3_COMP2_T4 {t in (end3 + 1)..arrival4}: crude24[t] = crude24[end3];
subject to 3_COMP2_T5 {t in (end3 + 1)..arrival4}: crude25[t] = crude25[end3];
subject to 3_COMP2_T6 {t in (end3 + 1)..arrival4}: crude26[t] = crude26[end3];

# parcel 4: Crude 3 (RGN) 100%
```

```
subject to COMP2_T1_4 {t in (arrival4 + 1)..end4}:
crude21[t] = crude21[t-1] + (crude4 - crude21[t-1])*q11[t-1]/vol1[t-1];
subject to COMP2_T2_4 {t in (arrival4 + 1)..end4}:
crude22[t] = crude22[t-1] + (crude4 - crude22[t-1])*q12[t-1]/vol2[t-1];
subject to COMP2_T3_4 {t in (arrival4 + 1)..end4}:
crude23[t] = crude23[t-1] + (crude4 - crude23[t-1])*q13[t-1]/vol3[t-1];
subject to COMP2_T4_4 {t in (arrival4 + 1)..end4}:
crude24[t] = crude24[t-1] + (crude4 - crude24[t-1])*q14[t-1]/vol4[t-1];
subject to COMP2_T5_4 {t in (arrival4 + 1)..end4}:
crude25[t] = crude25[t-1] + (crude4 - crude25[t-1])*q15[t-1]/vol5[t-1];
subject to COMP2_T6_4 {t in (arrival4 + 1)..end4}:
crude26[t] = crude26[t-1] + (crude4 - crude26[t-1])*q16[t-1]/vol6[t-1];

#####################################################
# Flow Constraints
#####################################################
#####################################################
# Pipeline flow
#####################################################
subject to PIPEFLOW {t in 0 .. T-1}:
(q11[t] + q12[t] + q13[t] + q14[t] + q15[t] + q16[t]) <= q_max_pipe;
subject to ARRIVAL_11 {t in 0..(arrival1 - 1)}:
q11[t] + q12[t] + q13[t] + q14[t] + q15[t] + q16[t] = 0;
subject to ARRIVAL_21 {t in (end1)..(arrival2 - 1)}:
q11[t] + q12[t] + q13[t] + q14[t] + q15[t] + q16[t]= 0;
subject to ARRIVAL_31 {t in (end2)..(arrival3 - 1)}:
q11[t] + q12[t] + q13[t] + q14[t] + q15[t] + q16[t]= 0;
subject to ARRIVAL_41 {t in (end3)..(arrival4 - 1)}:
q11[t] + q12[t] + q13[t] + q14[t] + q15[t] + q16[t]= 0;

#############################################################################
# Settling time implies in no running tank Constant Settling time = 24h
#############################################################################
subject to SETTLING_TIME_1 {t in 1..T-1}:
```

```
settling_1[t] = q1[t] * (sum{t1 in max(0, t - settling_time)..t} q11[t1]);
subject to SETTLING_TIME_2 {t in 1..T-1}:
settling_2[t] = q2[t] * (sum{t1 in max(0, t - settling_time)..t} q12[t1]);
subject to SETTLING_TIME_3 {t in 1..T-1}:
settling_3[t] = q3[t] * (sum{t1 in max(0, t - settling_time)..t} q13[t1]);
subject to SETTLING_TIME_4 {t in 1..T-1}:
settling_4[t] = q4[t] * (sum{t1 in max(0, t - settling_time)..t} q14[t1]);
subject to SETTLING_TIME_5 {t in 1..T-1}:
settling_5[t] = q5[t] * (sum{t1 in max(0, t - settling_time)..t} q15[t1]);
subject to SETTLING_TIME_6 {t in 1..T-1}:
settling_6[t] = q6[t] * (sum{t1 in max(0, t - settling_time)..t} q16[t1]);

#########################################################################
# At most 2 tanks can feed CDU
#########################################################################
subject to AT_MOST_TWO_TANKS_1 {t in 0..T-1}:
twocduflow_1[t] = q1[t]* (q2[t] + q3[t] + q4[t] + q5[t] + q6[t]);
subject to AT_MOST_TWO_TANKS_2 {t in 0..T-1}:
twocduflow_2[t] = q2[t]* (q3[t] + q4[t] + q5[t] + q6[t]);
subject to AT_MOST_TWO_TANKS_3 {t in 0..T-1}:
twocduflow_3[t] = q3[t]* (q4[t] + q5[t] + q6[t]);
subject to AT_MOST_TWO_TANKS_4 {t in 0..T-1}:
twocduflow_4[t] = q4[t]* (q5[t] + q6[t]);
subject to AT_MOST_TWO_TANKS_5 {t in 0..T-1}:
twocduflow_5[t] = q5[t]* (q6[t]);
var feedcrude2 {0..T-1};
# CDU Composition
subject to CDU_CRUDE_1: feedcrude2[0] = q1[0]*ini_cru21 + q2[0]*ini_cru22 +
q3[0]*ini_cru23 + q4[0]*ini_cru24 + q5[0]*ini_cru25 + q6[0]*ini_cru26;
subject to CDU_CRUDE_3 {t in 1..T-1}: feedcrude2[t] = q1[t]*crude21[t] +
q2[t]*crude22[t] +q3[t]*crude23[t] +q4[t]*crude24[t] +q5[t]*crude25[t]
+q6[t]*crude26[t];
subject to MIN_FEED_CRUDE2 {t in 0..T-1}:
feedcrude2[t]  >= 0;
```

```
subject to MAX_FEED_CRUDE2 {t in 0..T-1}:
feedcrude2[t]  <= (q1[t] + q2[t]+ q3[t]+ q4[t]+ q5[t]+ q6[t])/2;

#######################################################
# CDU Flow
#######################################################
subject to CDU_FLOW1 {t in 0..T-1}:
(q1[t] + q2[t]+ q3[t]+ q4[t]+ q5[t]+ q6[t]) <= feedflow;
#########################################################################
# Pipeline -> one Tank
# by definition, when one parcel is unloaded, the others have zero flow
#########################################################################
subject to ONE_PIPE_FLOW_1 {t in 0..T-1}:
onepipeflow_1[t] = q11[t]* (q12[t] + q13[t] + q14[t] +
q15[t] + q16[t]);
subject to ONE_PIPE_FLOW_2 {t in 0..T-1}:
onepipeflow_2[t] = q12[t]* (q13[t] + q14[t] + q15[t] + q16[t]);
subject to ONE_PIPE_FLOW_3 {t in 0..T-1}:
onepipeflow_3[t] = q13[t]* (q14[t] + q15[t] + q16[t]);
subject to ONE_PIPE_FLOW_4 {t in 0..T-1}:
onepipeflow_4[t] = q14[t]* (q15[t] + q16[t]);
subject to ONE_PIPE_FLOW_5 {t in 0..T-1}:
onepipeflow_5[t] = q15[t]* (q16[t]);

#########################################################################
# Objective Function - components
# Maximize Profit = (CDU Production) - (Cost of Tank Operation)
# In this formulation CDU feed is fixed, and tank operation as well
# Therefore, the objective function will be to minimize the
# tank switchover
#########################################################################
var switchover >= 0;
var twocdu >= 0;
var feed_diff >= 0;
```

```
var onepipe >= 0;
var settling >= 0;
param max_feed_diff;
let max_feed_diff := cdu_vol;
subject to MAX_FEED_DIFF: feed_diff <= max_feed_diff;
param max_onepipe;
let max_onepipe := 1;
subject to MAX_ONEPIPE: onepipe <= max_onepipe;
param max_settling;
let max_settling := 1;
subject to MAX_SETTLING: settling <= max_settling;
param max_switchover;
let max_switchover := 1000;
subject to MAX_SWITCHOVER: switchover <= max_switchover;
param max_twocdu;
let max_twocdu := 100;
subject to MAX_TWOCDU: twocdu <= max_twocdu;
subject to SWITCHOVER_COST: switchover = sum {t in 1..T-1} (
(q11[t-1] - q11[t])^2 + (q12[t-1] - q12[t])^2 + (q13[t-1] - q13[t])^2
+ (q14[t-1] - q14[t])^2 + (q15[t-1] - q15[t])^2 + (q16[t-1] - q16[t])^2
+ (q1[t-1] - q1[t])^2 + (q2[t-1] - q2[t])^2 + (q3[t-1] - q3[t])^2
+ (q4[t-1] - q4[t])^2
+ (q5[t-1] - q5[t])^2 + (q6[t-1] - q6[t])^2);

subject to FEED_COST:
feed_diff = cdu_vol - dt* (sum{t in 0..T-1}
(q1[t] + q2[t]+ q3[t]+ q4[t]+ q5[t]+ q6[t]));
subject to ONEPIPE_COST:
onepipe = sum{t in 0..T-1}(onepipeflow_1[t]+onepipeflow_2[t]+
onepipeflow_3[t]+onepipeflow_4[t]+onepipeflow_5[t]);
subject to SETTLING_COST:
settling = sum {t in 1..T-1}
(settling_1[t] + settling_2[t] + settling_3[t] +
settling_4[t] + settling_5[t] + settling_6[t]);
```

```
subject to TWOCDU_COST: twocdu = sum {t in 0..T-1}
(twocduflow_1[t] + twocduflow_2[t] + twocduflow_3[t] +
twocduflow_4[t] + twocduflow_5[t])/feedflow^2;
param W_SW;
let W_SW := 1;
param W_FEED;
let W_FEED := 1;
param W_PIPE;
let W_PIPE := 1;
param W_CDU;
let W_CDU := 1;
param W_SET;
let W_SET := 1;
param W_PENALTY;
let W_PENALTY := 0;
param W_COST;
let W_COST := 0;
var cost;
subject to COST: cost =  W_SW * switchover +  W_FEED * feed_diff;
var penalty;
subject to PENALTY:
penalty =  W_PIPE * onepipe + W_CDU * twocdu + W_SET * settling;
minimize obj: W_COST * cost + W_PENALTY * penalty;
```

A.2 Modelos dos Casos de Teste 6,7 e 8 (Pan et al., 2009)

A.2.1 Caso de Teste 6

```
#####################################################
# Example 1 - Typical refinery
# Based on the example #1 from
```

```
# Pan et al. (2009)
# Chemical Engineering Science 64, 965-983
#
# Originally from Reddy et al. (2004)
# Chemical Engineering Science 59, 1325-1341
#
#######################################################
#######################################################
# Parameters
#######################################################
param dt;
let dt := 1.0;
# No of periods 72h
param T = 72.0/dt;
#######################################################################
# 8 tanks (T1-18), 3 CDU (CDU1-CDU3), 2 classes of Oil (cl1, cl2)
# Tanks T1, T6-T8 and CDU3 handle class1 crude oil (C1-C4),
# while the rest handle class2 crude oil (C5-C8)
# ONLY ONE KEY COMPONENT IS CONSIDERED TO
# CARACTERIZE THE CDU FEED: SULFUR OR METAL
#######################################################################
# Max Tank volume (m3)
param max_vol1 = 570.000;
param max_vol2 = 980.000;
# Min Tank volume (m3)
param min_vol1 = 60.000;
param min_vol2 = 110.000;
# Initial volumes
param ini_vol1 = 350.000; # ini vol of tank1
param ini_vol2 = 400.000; # ini vol of tank2
param ini_vol3 = 350.000; # ini vol of tank3
param ini_vol4 = 950.000; # ini vol of tank4
param ini_vol5 = 300.000; # ini vol of tank5
param ini_vol6 = 80.000; # ini vol of tank6
```

```
param ini_vol7 = 80.000; # ini vol of tank7
param ini_vol8 = 450.000; # ini vol of tank8

##############################################
# KEY COMPONENT (SULFUR OR METAL)
##############################################
# Initial composition (8 crudes: C1, C2, C3, C4, C5, C6, C7, C8,
# each with a different key component concentration and net back (profit))
param key_c1 = 0.20;
param key_c2 = 0.25;
param key_c3 = 0.15;
param key_c4 = 0.60;
param key_c5 = 1.20;
param key_c6 = 1.30;
param key_c7 = 0.90;
param key_c8 = 1.50;
param profit_c1 = 1.50;
param profit_c2 = 1.70;
param profit_c3 = 1.50;
param profit_c4 = 1.60;
param profit_c5 = 1.45;
param profit_c6 = 1.60;
param profit_c7 = 1.55;
param profit_c8 = 1.60;
param ini_cru11 = 50/350; # crude 1 - tank 1
param ini_cru21 = 100/350; # crude 2 - tank 1
param ini_cru31 = 100/350; # crude 3 - tank 1
param ini_cru41 = 100/350; # crude 4 - tank 1
#param ini_cru51 = 0; # crude 5 - tank 1
#param ini_cru61 = 0; # crude 6 - tank 1
#param ini_cru71 = 0; # crude 7 - tank 1
#param ini_cru81 = 0; # crude 8 - tank 1
param ini_key_T1 = ini_cru11 * key_c1 +
ini_cru21 * key_c2 + ini_cru31 * key_c3 + ini_cru41 * key_c4;
```

```
#param ini_cru12 = 0; # crude 1 - tank 2
#param ini_cru22 = 0; # crude 2 - tank 2
#param ini_cru32 = 0; # crude 3 - tank 2
#param ini_cru42 = 0; # crude 4 - tank 2
param ini_cru52 = 0.25; # crude 5 - tank 2
param ini_cru62 = 0.25; # crude 6 - tank 2
param ini_cru72 = 0.25; # crude 7 - tank 2
param ini_cru82 = 0.25; # crude 8 - tank 2
param ini_key_T2 = ini_cru52 * key_c5 + ini_cru62 *
key_c6 + ini_cru72 * key_c7 + ini_cru82 * key_c8;

#param ini_cru13 = 0; # crude 1 - tank 3
#param ini_cru23 = 0; # crude 2 - tank 3
#param ini_cru33 = 0; # crude 3 - tank 3
#param ini_cru43 = 0; # crude 4 - tank 3
param ini_cru53 = 100/350; # crude 5 - tank 3
param ini_cru63 = 100/350; # crude 6 - tank 3
param ini_cru73 = 50/350; # crude 7 - tank 3
param ini_cru83 = 100/350; # crude 8 - tank 3
param ini_key_T3 = ini_cru53 * key_c5 + ini_cru63 * key_c6 +
ini_cru73 * key_c7 + ini_cru83 * key_c8;
#param ini_cru14 = 0; # crude 1 - tank 4
#param ini_cru24 = 0; # crude 2 - tank 4
#param ini_cru34 = 0; # crude 3 - tank 4
#param ini_cru44 = 0; # crude 4 - tank 4
param ini_cru54 = 200/950; # crude 5 - tank 4
param ini_cru64 = 250/950; # crude 6 - tank 4
param ini_cru74 = 200/950; # crude 7 - tank 4
param ini_cru84 = 300/950; # crude 8 - tank 4
param ini_key_T4 = ini_cru54 * key_c5 + ini_cru64 * key_c6 +
ini_cru74 * key_c7 + ini_cru84 * key_c8;
#param ini_cru15 = 0; # crude 1 - tank 5
#param ini_cru25 = 0; # crude 2 - tank 5
#param ini_cru35 = 0; # crude 3 - tank 5
```

```
#param ini_cru45 = 0; # crude 4 - tank 5
param ini_cru55 = 100/300; # crude 5 - tank 5
param ini_cru65 = 100/300; # crude 6 - tank 5
param ini_cru75 = 50/300; # crude 7 - tank 5
param ini_cru85 = 50/300; # crude 8 - tank 5
param ini_key_T5 = ini_cru55 * key_c5 + ini_cru65 * key_c6 +
ini_cru75 * key_c7 + ini_cru85 * key_c8;
param ini_cru16 = 0.25; # crude 1 - tank 6
param ini_cru26 = 0.25; # crude 2 - tank 6
param ini_cru36 = 0.25; # crude 3 - tank 6
param ini_cru46 = 0.25; # crude 4 - tank 6
#param ini_cru56 = 0; # crude 5 - tank 6
#param ini_cru66 = 0; # crude 6 - tank 6
#param ini_cru76 = 0; # crude 7 - tank 6
#param ini_cru86 = 0; # crude 8 - tank 6
param ini_key_T6 = ini_cru16 * key_c1 + ini_cru26 * key_c2 +
ini_cru36 * key_c3 + ini_cru46 * key_c4;
param ini_cru17 = 0.25; # crude 1 - tank 7
param ini_cru27 = 0.25; # crude 2 - tank 7
param ini_cru37 = 0.25; # crude 3 - tank 7
param ini_cru47 = 0.25; # crude 4 - tank 7
#param ini_cru57 = 0; # crude 5 - tank 7
#param ini_cru67 = 0; # crude 6 - tank 7
#param ini_cru77 = 0; # crude 7 - tank 7
#param ini_cru87 = 0; # crude 8 - tank 7
param ini_key_T7 = ini_cru17 * key_c1 + ini_cru27 * key_c2 +
ini_cru37 * key_c3 + ini_cru47 * key_c4;
param ini_cru18 = 100/450; # crude 1 - tank 8
param ini_cru28 = 100/450; # crude 2 - tank 8
param ini_cru38 = 100/450; # crude 3 - tank 8
param ini_cru48 = 150/450; # crude 4 - tank 8
#param ini_cru58 = 0; # crude 5 - tank 8
#param ini_cru68 = 0; # crude 6 - tank 8
#param ini_cru78 = 0; # crude 7 - tank 8
```

```
#param ini_cru88 = 0; # crude 8 - tank 8
param ini_key_T8 = ini_cru18 * key_c1 + ini_cru28 * key_c2 +
ini_cru38 * key_c3 + ini_cru48 * key_c4;

# max pipeline flow 400 kbbl/8h = 50 kbbl/h
param q_max_pipe = 50.0;
# Ship final volume
param final_parcelvol = 0.0;
# parcel volumes
param ini_parcel1 = 10.000;     # 10 kbbl C2
param ini_parcel2 = 250.000;  # 250 kbbl C3
param ini_parcel3 = 300.000;  # 300 kbbl C4
param ini_parcel4 = 190.000;  # 190 kbbl C5
#arrival crude parcels
param arrival1 = ceil(15/dt);   # t = 15h
param arrival2 = ceil(16/dt);  # t = 16h
param arrival3 = ceil(21/dt);  # t = 21h
param arrival4 = ceil(27/dt);  # t = 27h
#end crude parcels
# t = 15.2h -> 16h
param end1 = max(ceil(15.2/dt), arrival1 + 1);
# t = 16h + 5h = 21h
param end2 = max(ceil(21/dt), arrival2 + 1);
# t = 21h + 6h = 27h
param end3 = max(ceil(27/dt), arrival3 + 1);
# t = 27h + 4h = 31h
param end4 = max(ceil(31/dt), arrival4 + 1);
#crude parcels
param key_parcel1 = key_c2;   # 100% C2
param key_parcel2 = key_c3;  # 100% C3
param key_parcel3 = key_c4;  # 100% C4
param key_parcel4 = key_c5;  # 100% C5
#param profit_parcel1 = profit_c2;  # 100% C2
#param profit_parcel2 = profit_c3;  # 100% C3
```

```
#param profit_parcel3 = profit_c4;  # 100% C4
#param profit_parcel4 = profit_c5;  # 100% C5
# resting/settling time
# 8h to settle brine
param settling_time = ceil(8/dt);
#CDU feed flow 48 kbbl/8h = 6 kbbl/h
param feedflow = 6.0;
param cdu_vol = 300;
param min_feedflow = 2.0;
# CDU feed key component limits
param feed_key_max_A = 1.40;
param feed_key_max_B = 1.30;
param feed_key_max_C = 0.40;
param feed_key_min_A = 0.10;
param feed_key_min_B = 0.10;
param feed_key_min_C = 0.10;
#min tank flow
param min_q = 0;
# penalty variables
var twocduflow_1C {0..T-1} ;
var twocduflow_2A {0..T-1} ;
var twocduflow_3A {0..T-1} ;
var twocduflow_4A {0..T-1} ;
var twocduflow_2B {0..T-1} ;
var twocduflow_3B {0..T-1} ;
var twocduflow_4B {0..T-1} ;
var twocduflow_6C {0..T-1} ;
var twocduflow_7C {0..T-1} ;
var settling_1 {1..T-1} ;
var settling_2 {1..T-1} ;
var settling_3 {1..T-1} ;
var settling_4 {1..T-1} ;
var settling_5 {1..T-1} ;
var settling_6 {1..T-1} ;
```

```
var settling_7 {1..T-1} ;
var settling_8 {1..T-1} ;
var onepipeflow_1 {0..T-1} ;
var onepipeflow_2 {0..T-1} ;
var onepipeflow_3 {0..T-1} ;
var onepipeflow_4 {0..T-1} ;
var onepipeflow_5 {0..T-1} ;
var onepipeflow_6 {0..T-1} ;
var onepipeflow_7 {0..T-1} ;

#####################################################
# Control variables
#####################################################
# transferred volume from pipeline VLCC to T1
var q11 {0..T-1} <= q_max_pipe, := 0;
# transferred volume from pipeline VLCC to T2
var q12 {0..T-1} <= q_max_pipe, := 0;
# transferred volume from pipeline VLCC to T3
var q13 {0..T-1} <= q_max_pipe, := 0;
# transferred volume from pipeline VLCC to T4
var q14 {0..T-1} <= q_max_pipe, := 0;
# transferred volume from pipeline VLCC to T5
var q15 {0..T-1} <= q_max_pipe, := 0;
# transferred volume from pipeline VLCC to T6
var q16 {0..T-1} <= q_max_pipe, := 0;
# transferred volume from pipeline VLCC to T7
var q17 {0..T-1} <= q_max_pipe, := 0;
# transferred volume from pipeline VLCC to T8
var q18 {0..T-1} <= q_max_pipe, := 0;
subject to MIN_Q11 {t in 0..T-1}: q11[t]  >= min_q;
subject to MIN_Q12 {t in 0..T-1}: q12[t]  >= min_q;
subject to MIN_Q13 {t in 0..T-1}: q13[t]  >= min_q;
subject to MIN_Q14 {t in 0..T-1}: q14[t]  >= min_q;
subject to MIN_Q15 {t in 0..T-1}: q15[t]  >= min_q;
```

```
subject to MIN_Q16 {t in 0..T-1}: q16[t]  >= min_q;
subject to MIN_Q17 {t in 0..T-1}: q17[t]  >= min_q;
subject to MIN_Q18 {t in 0..T-1}: q18[t]  >= min_q;
# transferred volume from T1 to CDU1
var q1A {0..T-1} <= feedflow, := 0;
# transferred volume from T2 to CDU1
var q2A {0..T-1} <= feedflow, := 0;
# transferred volume from T3 to CDU1
var q3A {0..T-1} <= feedflow, := 0;
# transferred volume from T4 to CDU1
var q4A {0..T-1} <= feedflow, := 0;
# transferred volume from T5 to CDU1
var q5A {0..T-1} <= feedflow, := 0;
# transferred volume from T6 to CDU1
var q6A {0..T-1} <= feedflow, := 0;
# transferred volume from T7 to CDU1
var q7A {0..T-1} <= feedflow, := 0;
# transferred volume from T8 to CDU1
var q8A {0..T-1} <= feedflow, := 0;
subject to MIN_Q1A {t in 0..T-1}: q1A[t]  = min_q;
subject to MIN_Q2A {t in 0..T-1}: q2A[t]  >= min_q;
subject to MIN_Q3A {t in 0..T-1}: q3A[t]  >= min_q;
subject to MIN_Q4A {t in 0..T-1}: q4A[t]  >= min_q;
subject to MIN_Q5A {t in 0..T-1}: q5A[t]  >= min_q;
subject to MIN_Q6A {t in 0..T-1}: q6A[t]  = min_q;
subject to MIN_Q7A {t in 0..T-1}: q7A[t]  = min_q;
subject to MIN_Q8A {t in 0..T-1}: q8A[t]  = min_q;
# transferred volume from T1 to CDU2
var q1B {0..T-1} <= feedflow, := 0;
# transferred volume from T2 to CDU2
var q2B {0..T-1} <= feedflow, := 0;
# transferred volume from T3 to CDU2
var q3B {0..T-1} <= feedflow, := 0;
# transferred volume from T4 to CDU2
```

```
var q4B {0..T-1} <= feedflow, := 0;
# transferred volume from T5 to CDU2
var q5B {0..T-1} <= feedflow, := 0;
# transferred volume from T6 to CDU2
var q6B {0..T-1} <= feedflow, := 0;
# transferred volume from T7 to CDU2
var q7B {0..T-1} <= feedflow, := 0;
# transferred volume from T8 to CDU2
var q8B {0..T-1} <= feedflow, := 0;
subject to MIN_Q1B {t in 0..T-1}: q1B[t]  = min_q;
subject to MIN_Q2B {t in 0..T-1}: q2B[t]  >= min_q;
subject to MIN_Q3B {t in 0..T-1}: q3B[t]  >= min_q;
subject to MIN_Q4B {t in 0..T-1}: q4B[t]  >= min_q;
subject to MIN_Q5B {t in 0..T-1}: q5B[t]  >= min_q;
subject to MIN_Q6B {t in 0..T-1}: q6B[t]  = min_q;
subject to MIN_Q7B {t in 0..T-1}: q7B[t]  = min_q;
subject to MIN_Q8B {t in 0..T-1}: q8B[t]  = min_q;
# transferred volume from T1 to CDU3
var q1C {0..T-1} <= feedflow, := 0;
# transferred volume from T2 to CDU3
var q2C {0..T-1} <= feedflow, := 0;
# transferred volume from T3 to CDU3
var q3C {0..T-1} <= feedflow, := 0;
# transferred volume from T4 to CDU3
var q4C {0..T-1} <= feedflow, := 0;
# transferred volume from T5 to CDU3
var q5C {0..T-1} <= feedflow, := 0;
# transferred volume from T6 to CDU3
var q6C {0..T-1} <= feedflow, := 0;
# transferred volume from T7 to CDU3
var q7C {0..T-1} <= feedflow, := 0;
# transferred volume from T8 to CDU3
var q8C {0..T-1} <= feedflow, := 0;
subject to MIN_Q1C {t in 0..T-1}: q1C[t]  >= min_q;
```

```
subject to MIN_Q2C {t in 0..T-1}: q2C[t]  = min_q;
subject to MIN_Q3C {t in 0..T-1}: q3C[t]  = min_q;
subject to MIN_Q4C {t in 0..T-1}: q4C[t]  = min_q;
subject to MIN_Q5C {t in 0..T-1}: q5C[t]  = min_q;
subject to MIN_Q6C {t in 0..T-1}: q6C[t]  >= min_q;
subject to MIN_Q7C {t in 0..T-1}: q7C[t]  >= min_q;
subject to MIN_Q8C {t in 0..T-1}: q8C[t]  >= min_q;
####################################
# State Variables
####################################
var vol1 {1..T} <= max_vol1, := ini_vol1;
var vol2 {1..T} <= max_vol1, := ini_vol2;
var vol3 {1..T} <= max_vol1, := ini_vol3;
var vol4 {1..T} <= max_vol2, := ini_vol4;
var vol5 {1..T} <= max_vol2, := ini_vol5;
var vol6 {1..T} <= max_vol1, := ini_vol6;
var vol7 {1..T} <= max_vol1, := ini_vol7;
var vol8 {1..T} <= max_vol1, := ini_vol8;

subject to MIN_VOL1 {t in 1..T}: vol1[t]  >= min_vol1;
subject to MIN_VOL2 {t in 1..T}: vol2[t]  >= min_vol1;
subject to MIN_VOL3 {t in 1..T}: vol3[t]  >= min_vol1;
subject to MIN_VOL4 {t in 1..T}: vol4[t]  >= min_vol2;
subject to MIN_VOL5 {t in 1..T}: vol5[t]  >= min_vol2;
subject to MIN_VOL6 {t in 1..T}: vol6[t]  >= min_vol1;
subject to MIN_VOL7 {t in 1..T}: vol7[t]  >= min_vol1;
subject to MIN_VOL8 {t in 1..T}: vol8[t]  >= min_vol1;

var parcel1 {arrival1..end1} <= ini_parcel1, := ini_parcel1;
var parcel2 {arrival2..end2} <= ini_parcel2, := ini_parcel2;
var parcel3 {arrival3..end3} <= ini_parcel3, := ini_parcel3;
var parcel4 {arrival4..end4} <= ini_parcel4, := ini_parcel4;
var parcel5;
var parcel6;
```

```
var parcel7;
var parcel8;
subject to MIN_PARCEL21 {t in arrival1..end1}:
parcel1[t]  >= final_parcelvol;
subject to MIN_PARCEL22 {t in arrival2..end2}:
parcel2[t]  >= final_parcelvol;
subject to MIN_PARCEL23 {t in arrival3..end3}:
parcel3[t]  >= final_parcelvol;
subject to MIN_PARCEL24 {t in arrival4..end4}:
parcel4[t]  >= final_parcelvol;

#####################################################
# Tank Composition
#####################################################
var key_T1 {1..T} := ini_key_T1;        # key in T1
var key_T2 {1..T} := ini_key_T2;        # key in T2
var key_T3 {1..T} := ini_key_T3;        # key in T3
var key_T4 {1..T} := ini_key_T4;        # key in T4
var key_T5 {1..T} := ini_key_T5;        # key in T5
var key_T6 {1..T} := ini_key_T6;        # key in T6
var key_T7 {1..T} := ini_key_T7;        # key in T7
var key_T8 {1..T} := ini_key_T8;        # key in T8

#####################################################
# State Equations - volumes
#####################################################
subject to INITIAL_STATE1: vol1[1] =
ini_vol1 + dt*(q11[0] - q1A[0] - q1B[0] - q1C[0]);
subject to STATE_EQ1 {t in 2..T}: vol1[t] =
vol1[t-1] + dt*(q11[t-1] - q1A[t-1] - q1B[t-1] - q1C[t-1]);
subject to INITIAL_STATE2: vol2[1] =
ini_vol2 + dt*(q12[0] - q2A[0] - q2B[0] - q2C[0]);
subject to STATE_EQ2 {t in 2..T}: vol2[t] =
vol2[t-1] + dt*(q12[t-1] - q2A[t-1] - q2B[t-1] - q2C[t-1]);
```

```
subject to INITIAL_STATE3: vol3[1] =
ini_vol3 + dt*(q13[0] - q3A[0] - q3B[0] - q3C[0]);
subject to STATE_EQ3 {t in 2..T}: vol3[t] =
vol3[t-1] + dt*(q13[t-1] - q3A[t-1] - q3B[t-1] - q3C[t-1]);
subject to INITIAL_STATE4: vol4[1] =
ini_vol4 + dt*(q14[0] - q4A[0] - q4B[0] - q4C[0]);
subject to STATE_EQ4 {t in 2..T}: vol4[t] =
vol4[t-1] + dt*(q14[t-1] - q4A[t-1] - q4B[t-1] - q4C[t-1]);
subject to INITIAL_STATE5: vol5[1] =
ini_vol5 + dt*(q15[0] - q5A[0] - q5B[0] - q5C[0]);
subject to STATE_EQ5 {t in 2..T}: vol5[t] =
vol5[t-1] + dt*(q15[t-1] - q5A[t-1] - q5B[t-1] - q5C[t-1]);
subject to INITIAL_STATE6: vol6[1] =
ini_vol6 + dt*(q16[0] - q6A[0] - q6B[0] - q6C[0]);
subject to STATE_EQ6 {t in 2..T}: vol6[t] =
vol6[t-1] + dt*(q16[t-1] - q6A[t-1] - q6B[t-1] - q6C[t-1]);
subject to INITIAL_STATE7: vol7[1] =
ini_vol7 + dt*(q17[0] - q7A[0] - q7B[0] - q7C[0]);
subject to STATE_EQ7 {t in 2..T}: vol7[t] =
vol7[t-1] + dt*(q17[t-1] - q7A[t-1] - q7B[t-1] - q7C[t-1]);
subject to INITIAL_STATE8: vol8[1] =
ini_vol8 + dt*(q18[0] - q8A[0] - q8B[0] - q8C[0]);
subject to STATE_EQ8 {t in 2..T}: vol8[t] =
vol8[t-1] + dt*(q18[t-1] - q8A[t-1] - q8B[t-1] - q8C[t-1]);
##########################################################
# State Equations - Crude parcel volumes
##########################################################
#C2 only in T1, T6-T8
subject to INITIAL_PARCEL1: parcel1[arrival1] = ini_parcel1;
subject to STATE_PARCEL1 {t in (arrival1 + 1)..(end1)}:
parcel1[t] = parcel1[t-1] - dt*(q11[t-1] + q16[t-1] + q17[t-1] + q18[t-1]);
subject to STATE_PARCEL1_C2 {t in (arrival1 + 1)..(end1)}:
q12[t-1] + q13[t-1] + q14[t-1] + q15[t-1] = 0;
subject to FINAL_PARCEL_1: parcel1[end1] = final_parcelvol;
```

```
#C3 only in T1, T6-T8
subject to INITIAL_PARCEL2: parcel2[arrival2] = ini_parcel2;
subject to STATE_PARCEL2 {t in (arrival2 + 1)..(end2)}:
parcel2[t] = parcel2[t-1] - dt*(q11[t-1] + q16[t-1] + q17[t-1] + q18[t-1]);
subject to STATE_PARCEL2_C3 {t in (arrival2 + 1)..(end2)}:
q12[t-1] + q13[t-1] + q14[t-1] + q15[t-1] = 0;
subject to FINAL_PARCEL_2: parcel2[end2] = final_parcelvol;
#C4 only in T1, T6-T8
subject to INITIAL_PARCEL3: parcel3[arrival3] = ini_parcel3;
subject to STATE_PARCEL3 {t in (arrival3 + 1)..(end3)}:
parcel3[t] = parcel3[t-1] - dt*(q11[t-1] + q16[t-1] + q17[t-1] + q18[t-1]);
subject to STATE_PARCEL3_C4 {t in (arrival3 + 1)..(end3)}:
q12[t-1] + q13[t-1] + q14[t-1] + q15[t-1] = 0;
subject to FINAL_PARCEL_3: parcel3[end3] = final_parcelvol;
#C5 only in T2-T5
subject to INITIAL_PARCEL4: parcel4[arrival4] = ini_parcel4;
subject to STATE_PARCEL4 {t in (arrival4 + 1)..(end4)}:
parcel4[t] = parcel4[t-1] - dt*(q12[t-1] + q13[t-1] + q14[t-1] + q15[t-1]);
subject to STATE_PARCEL4_C5 {t in (arrival4 + 1)..(end4)}:
q11[t-1] + q16[t-1] + q17[t-1] + q18[t-1] = 0;
subject to FINAL_PARCEL_4: parcel4[end4] = final_parcelvol;

################################################################
# State Equations - composition
# Crude X at Tank Y = initial crude x + crude x from
# pipeline - crude x to CDU
################################################################
# parcel 1: Crude 2 100% => T1, T6-T8
subject to INITIAL_COMP_T1 {t in 1..arrival1}: key_T1[t] = ini_key_T1;
subject to INITIAL_COMP_T2 {t in 1..arrival1}: key_T2[t] = ini_key_T2;
subject to INITIAL_COMP_T3 {t in 1..arrival1}: key_T3[t] = ini_key_T3;
subject to INITIAL_COMP_T4 {t in 1..arrival1}: key_T4[t] = ini_key_T4;
subject to INITIAL_COMP_T5 {t in 1..arrival1}: key_T5[t] = ini_key_T5;
```

```
subject to INITIAL_COMP_T6 {t in 1..arrival1}: key_T6[t] = ini_key_T6;
subject to INITIAL_COMP_T7 {t in 1..arrival1}: key_T7[t] = ini_key_T7;
subject to INITIAL_COMP_T8 {t in 1..arrival1}: key_T8[t] = ini_key_T8;
subject to COMP2_T1_1 {t in (arrival1 + 1)..end1}:
key_T1[t] = key_T1[t-1] + (key_parcel1 - key_T1[t-1])*q11[t-1]/vol1[t-1];
subject to COMP2_T2_1 {t in (arrival1 + 1)..end1}: key_T2[t] = ini_key_T2;
subject to COMP2_T3_1 {t in (arrival1 + 1)..end1}: key_T3[t] = ini_key_T3;
subject to COMP2_T4_1 {t in (arrival1 + 1)..end1}: key_T4[t] = ini_key_T4;
subject to COMP2_T5_1 {t in (arrival1 + 1)..end1}: key_T5[t] = ini_key_T5;
subject to COMP2_T6_1 {t in (arrival1 + 1)..end1}:
key_T6[t] = key_T6[t-1] + (key_parcel1 - key_T6[t-1])*q16[t-1]/vol6[t-1];
subject to COMP2_T7_1 {t in (arrival1 + 1)..end1}:
key_T7[t] = key_T7[t-1] + (key_parcel1 - key_T7[t-1])*q17[t-1]/vol7[t-1];
subject to COMP2_T8_1 {t in (arrival1 + 1)..end1}:
key_T8[t] = key_T8[t-1] + (key_parcel1 - key_T8[t-1])*q18[t-1]/vol8[t-1];

# parcel 2: Crude 3 100% => T1, T6-T8
subject to COMP2_T1_2 {t in (arrival2 + 1)..end2}:
key_T1[t] = key_T1[t-1] + (key_parcel2 - key_T1[t-1])*q11[t-1]/vol1[t-1];
subject to COMP2_T2_2 {t in (arrival2 + 1)..end2}: key_T2[t] = ini_key_T2;
subject to COMP2_T3_2 {t in (arrival2 + 1)..end2}: key_T3[t] = ini_key_T3;
subject to COMP2_T4_2 {t in (arrival2 + 1)..end2}: key_T4[t] = ini_key_T4;
subject to COMP2_T5_2 {t in (arrival2 + 1)..end2}: key_T5[t] = ini_key_T5;
subject to COMP2_T6_2 {t in (arrival2 + 1)..end2}:
key_T6[t] = key_T6[t-1] + (key_parcel2 - key_T6[t-1])*q16[t-1]/vol6[t-1];
subject to COMP2_T7_2 {t in (arrival2 + 1)..end2}:
key_T7[t] = key_T7[t-1] + (key_parcel2 - key_T7[t-1])*q17[t-1]/vol7[t-1];
subject to COMP2_T8_2 {t in (arrival2 + 1)..end2}:
key_T8[t] = key_T8[t-1] + (key_parcel2 - key_T8[t-1])*q18[t-1]/vol8[t-1];

# parcel 3: Crude 4 100% => T1, T6-T8
subject to COMP2_T1_3 {t in (arrival3 + 1)..end3}:
key_T1[t] = key_T1[t-1] + (key_parcel3 - key_T1[t-1])*q11[t-1]/vol1[t-1];
subject to COMP2_T2_3 {t in (arrival3 + 1)..end3}: key_T2[t] = ini_key_T2;
```

```
subject to COMP2_T3_3 {t in (arrival3 + 1)..end3}: key_T3[t] = ini_key_T3;
subject to COMP2_T4_3 {t in (arrival3 + 1)..end3}: key_T4[t] = ini_key_T4;
subject to COMP2_T5_3 {t in (arrival3 + 1)..end3}: key_T5[t] = ini_key_T5;
subject to COMP2_T6_3 {t in (arrival3 + 1)..end3}:
key_T6[t] = key_T6[t-1] + (key_parcel3 - key_T6[t-1])*q16[t-1]/vol6[t-1];
subject to COMP2_T7_3 {t in (arrival3 + 1)..end3}:
key_T7[t] = key_T7[t-1] + (key_parcel3 - key_T7[t-1])*q17[t-1]/vol7[t-1];
subject to COMP2_T8_3 {t in (arrival3 + 1)..end3}:
key_T8[t] = key_T8[t-1] + (key_parcel3 - key_T8[t-1])*q18[t-1]/vol8[t-1];

# parcel 4: Crude 5 100% => T2-T5
subject to COMP2_T1_4 {t in (arrival4 + 1)..end4}: key_T1[t] = key_T1[end3];
subject to COMP2_T2_4 {t in (arrival4 + 1)..end4}:
key_T2[t] = key_T2[t-1] + (key_parcel4 - key_T2[t-1])*q12[t-1]/vol2[t-1];
subject to COMP2_T3_4 {t in (arrival4 + 1)..end4}:
key_T3[t] = key_T3[t-1] + (key_parcel4 - key_T3[t-1])*q13[t-1]/vol3[t-1];
subject to COMP2_T4_4 {t in (arrival4 + 1)..end4}:
key_T4[t] = key_T4[t-1] + (key_parcel4 - key_T4[t-1])*q14[t-1]/vol4[t-1];
subject to COMP2_T5_4 {t in (arrival4 + 1)..end4}:
key_T2[t] = key_T5[t-1] + (key_parcel4 - key_T5[t-1])*q15[t-1]/vol5[t-1];
subject to COMP2_T6_4 {t in (arrival4 + 1)..end4}: key_T6[t] = key_T6[end3];
subject to COMP2_T7_4 {t in (arrival4 + 1)..end4}: key_T7[t] = key_T7[end3];
subject to COMP2_T8_4 {t in (arrival4 + 1)..end4}: key_T8[t] = key_T8[end3];

subject to 4_COMP2_T1 {t in (end4 + 1)..T}: key_T1[t] = key_T1[end4];
subject to 4_COMP2_T2 {t in (end4 + 1)..T}: key_T2[t] = key_T2[end4];
subject to 4_COMP2_T3 {t in (end4 + 1)..T}: key_T3[t] = key_T3[end4];
subject to 4_COMP2_T4 {t in (end4 + 1)..T}: key_T4[t] = key_T4[end4];
subject to 4_COMP2_T5 {t in (end4 + 1)..T}: key_T5[t] = key_T5[end4];
subject to 4_COMP2_T6 {t in (end4 + 1)..T}: key_T6[t] = key_T6[end4];
subject to 4_COMP2_T7 {t in (end4 + 1)..T}: key_T7[t] = key_T7[end4];
subject to 4_COMP2_T8 {t in (end4 + 1)..T}: key_T8[t] = key_T8[end4];

#####################################################################
```

```
# Flow Constraints
#########################################################################
#####################################################
# Ship flow
#####################################################
#####################################################################
# Settling time implies in no running tank
# Constant Settling time = 8h
#####################################################################
subject to PIPEFLOW {t in 0 .. T-1}:
(q11[t] + q12[t] + q13[t] + q14[t] + q15[t] + q16[t] + q17[t] + q18[t]) <= q_max_pipe;
subject to ARRIVAL_PREVIOUS {t in 0..(arrival1 - 1)}:
q11[t] + q12[t] + q13[t] + q14[t] + q15[t] + q16[t] + q17[t] + q18[t] = 0;
# Only allow flow on tanks that can receive the crude parcel
subject to ARRIVAL_Q11_1 {t in (arrival1)..(end1 - 1)}: q11[t] >= q11[t-1];
subject to ARRIVAL_Q12_1 {t in (arrival1)..(end1 - 1)}: q12[t] = 0;
subject to ARRIVAL_Q13_1 {t in (arrival1)..(end1 - 1)}: q13[t] = 0;
subject to ARRIVAL_Q14_1 {t in (arrival1)..(end1 - 1)}: q14[t] = 0;
subject to ARRIVAL_Q15_1 {t in (arrival1)..(end1 - 1)}: q15[t] = 0;
subject to ARRIVAL_Q16_1 {t in (arrival1)..(end1 - 1)}: q16[t] >= q16[t-1];
subject to ARRIVAL_Q17_1 {t in (arrival1)..(end1 - 1)}: q17[t] >= q17[t-1];
subject to ARRIVAL_Q18_1 {t in (arrival1)..(end1 - 1)}: q18[t] >= q18[t-1];

# the first tank choice is free
subject to ARRIVAL_Q11_2 {t in (arrival2 + 1)..(end2 - 1)}: q11[t] >= q11[t-1];
subject to ARRIVAL_Q12_2 {t in (arrival2 + 1)..(end2 - 1)}: q12[t] = 0;
subject to ARRIVAL_Q13_2 {t in (arrival2 + 1)..(end2 - 1)}: q13[t] = 0;
subject to ARRIVAL_Q14_2 {t in (arrival2 + 1)..(end2 - 1)}: q14[t] = 0;
subject to ARRIVAL_Q15_2 {t in (arrival2 + 1)..(end2 - 1)}: q15[t] = 0;
subject to ARRIVAL_Q16_2 {t in (arrival2 + 1)..(end2 - 1)}: q16[t] >= q16[t-1];
subject to ARRIVAL_Q17_2 {t in (arrival2 + 1)..(end2 - 1)}: q17[t] >= q17[t-1];
subject to ARRIVAL_Q18_2 {t in (arrival2 + 1)..(end2 - 1)}: q18[t] >= q18[t-1];

# the first tank choice is free
```

```
subject to ARRIVAL_Q11_3 {t in (arrival3 + 1)..(end3 - 1)}: q11[t] >= q11[t-1];
subject to ARRIVAL_Q12_3 {t in (arrival3 + 1)..(end3 - 1)}: q12[t] = 0;
subject to ARRIVAL_Q13_3 {t in (arrival3 + 1)..(end3 - 1)}: q13[t] = 0;
subject to ARRIVAL_Q14_3 {t in (arrival3 + 1)..(end3 - 1)}: q14[t] = 0;
subject to ARRIVAL_Q15_3 {t in (arrival3 + 1)..(end3 - 1)}: q15[t] = 0;
subject to ARRIVAL_Q16_3 {t in (arrival3 + 1)..(end3 - 1)}: q16[t] >= q16[t-1];
subject to ARRIVAL_Q17_3 {t in (arrival3 + 1)..(end3 - 1)}: q17[t] >= q17[t-1];
subject to ARRIVAL_Q18_3 {t in (arrival3 + 1)..(end3 - 1)}: q18[t] >= q18[t-1];

subject to ARRIVAL_CDU_1 {t in (arrival1)..(end3 - 1)}: q11[t] * q1C[t] = 0;
subject to ARRIVAL_CDU_6 {t in (arrival1)..(end3 - 1)}: q16[t] * q6C[t] = 0;
subject to ARRIVAL_CDU_7 {t in (arrival1)..(end3 - 1)}: q17[t] * q7C[t] = 0;
subject to ARRIVAL_CDU_8 {t in (arrival1)..(end3 - 1)}: q18[t] * q8C[t] = 0;

subject to ARRIVAL_CDU_1_SETTLING_1
{t in (end1)..(end1 + settling_time - 1)}: q11[end1 - 1] * q1C[t] = 0;
subject to ARRIVAL_CDU_6_SETTLING_1
{t in (end1)..(end1 + settling_time - 1)}: q16[end1 - 1] * q6C[t] = 0;
subject to ARRIVAL_CDU_7_SETTLING_1
{t in (end1)..(end1 + settling_time - 1)}: q17[end1 - 1] * q7C[t] = 0;
subject to ARRIVAL_CDU_8_SETTLING_1
{t in (end1)..(end1 + settling_time - 1)}: q18[end1 - 1] * q8C[t] = 0;

subject to ARRIVAL_CDU_1_SETTLING_2
{t in (end2)..(end2 + settling_time - 1)}: q11[end2 - 1] * q1C[t] = 0;
subject to ARRIVAL_CDU_6_SETTLING_2
{t in (end2)..(end2 + settling_time - 1)}: q16[end2 - 1] * q6C[t] = 0;
subject to ARRIVAL_CDU_7_SETTLING_2
{t in (end2)..(end2 + settling_time - 1)}: q17[end2 - 1] * q7C[t] = 0;
subject to ARRIVAL_CDU_8_SETTLING_2
{t in (end2)..(end2 + settling_time - 1)}: q18[end2 - 1] * q8C[t] = 0;

subject to ARRIVAL_CDU_1_SETTLING_3
{t in (end3)..(end3 + settling_time - 1)}: q11[end3 - 1] * q1C[t] = 0;
```

```
subject to ARRIVAL_CDU_6_SETTLING_3
{t in (end3)..(end3 + settling_time - 1)}: q16[end3 - 1] * q6C[t] = 0;
subject to ARRIVAL_CDU_7_SETTLING_3
{t in (end3)..(end3 + settling_time - 1)}: q17[end3 - 1] * q7C[t] = 0;
subject to ARRIVAL_CDU_8_SETTLING_3
{t in (end3)..(end3 + settling_time - 1)}: q18[end3 - 1] * q8C[t] = 0;

subject to ARRIVAL_Q11_4 {t in (arrival4)..(end4 - 1)}: q11[t] = 0;
subject to ARRIVAL_Q12_4 {t in (arrival4)..(end4 - 1)}: q12[t] >= q12[t-1];
subject to ARRIVAL_Q13_4 {t in (arrival4)..(end4 - 1)}: q13[t] >= q13[t-1];
subject to ARRIVAL_Q14_4 {t in (arrival4)..(end4 - 1)}: q14[t] >= q14[t-1];
subject to ARRIVAL_Q15_4 {t in (arrival4)..(end4 - 1)}: q15[t] >= q15[t-1];
subject to ARRIVAL_Q16_4 {t in (arrival4)..(end4 - 1)}: q16[t] = 0;
subject to ARRIVAL_Q17_4 {t in (arrival4)..(end4 - 1)}: q17[t] = 0;
subject to ARRIVAL_Q18_4 {t in (arrival4)..(end4 - 1)}: q18[t] = 0;

subject to ARRIVAL_CDU_2 {t in (arrival4)..(end4 - 1)}:
q12[t] * (q2A[t] + q2B[t]) = 0;
subject to ARRIVAL_CDU_3 {t in (arrival4)..(end4 - 1)}:
q13[t] * (q3A[t] + q3B[t]) = 0;
subject to ARRIVAL_CDU_4 {t in (arrival4)..(end4 - 1)}:
q14[t] * (q4A[t] + q4B[t]) = 0;
subject to ARRIVAL_CDU_5 {t in (arrival4)..(end4 - 1)}:
q15[t] * (q5A[t] + q5B[t]) = 0;

subject to ARRIVAL_CDU_2_SETTLING_4
{t in (end4)..(end4 + settling_time - 1)}: q12[end4 - 1] * (q2A[t] + q2B[t]) = 0;
subject to ARRIVAL_CDU_3_SETTLING_4
{t in (end4)..(end4 + settling_time - 1)}: q13[end4 - 1] * (q3A[t] + q3B[t]) = 0;
subject to ARRIVAL_CDU_4_SETTLING_4
{t in (end4)..(end4 + settling_time - 1)}: q14[end4 - 1] * (q4A[t] + q4B[t]) = 0;
subject to ARRIVAL_CDU_5_SETTLING_4
{t in (end4)..(end4 + settling_time - 1)}: q15[end4 - 1] * (q5A[t] + q5B[t]) = 0;
```

```
subject to ARRIVAL_END {t in (end4)..(T-1)}:
  q11[t] + q12[t] + q13[t] + q14[t] + q15[t] + q16[t] + q17[t] + q18[t] = 0;

#############################################################################
# CDU flow
#############################################################################
subject to CDU_FLOW_A {t in 0..T-1}:
(q1A[t] + q2A[t]+ q3A[t]+ q4A[t]+ q5A[t]+ q6A[t] + q7A[t] + q8A[t]) <= feedflow;
subject to CDU_FLOW_B {t in 0..T-1}:
(q1B[t] + q2B[t]+ q3B[t]+ q4B[t]+ q5B[t]+ q6B[t] + q7B[t] + q8B[t]) <= feedflow;
subject to CDU_FLOW_C {t in 0..T-1}:
(q1C[t] + q2C[t]+ q3C[t]+ q4C[t]+ q5C[t]+ q6C[t] + q7C[t] + q8C[t]) <= feedflow;
subject to CDU_FLOW_MIN_A {t in 0..T-1}:
(q1A[t] + q2A[t]+ q3A[t]+ q4A[t]+ q5A[t]+ q6A[t] + q7A[t] + q8A[t]) >= min_feedflow;
subject to CDU_FLOW_MIN_B {t in 0..T-1}:
(q1B[t] + q2B[t]+ q3B[t]+ q4B[t]+ q5B[t]+ q6B[t] + q7B[t] + q8B[t]) >= min_feedflow;
subject to CDU_FLOW_MIN_C {t in 0..T-1}:
(q1C[t] + q2C[t]+ q3C[t]+ q4C[t]+ q5C[t]+ q6C[t] + q7C[t] + q8C[t]) >= min_feedflow;
subject to CDU_VOL_A: dt*(sum {t in 0..T-1}
(q1A[t] + q2A[t]+ q3A[t]+ q4A[t]+ q5A[t]+ q6A[t] + q7A[t] + q8A[t])) >= cdu_vol;
subject to CDU_VOL_B: dt*(sum {t in 0..T-1}
(q1B[t] + q2B[t]+ q3B[t]+ q4B[t]+ q5B[t]+ q6B[t] + q7B[t] + q8B[t])) >= cdu_vol;
subject to CDU_VOL_C: dt*(sum {t in 0..T-1}
(q1C[t] + q2C[t]+ q3C[t]+ q4C[t]+ q5C[t]+ q6C[t] + q7C[t] + q8C[t])) >= cdu_vol;

#############################################################################################
# At most 2 tanks can feed CDU
#############################################################################################
subject to AT_MOST_TWO_TANKS_1C {t in 0..T-1}:
twocduflow_1C[t] = q1C[t]* (q6C[t] + q7C[t] + q8C[t]);
subject to AT_MOST_TWO_TANKS_6C {t in 0..T-1}:
twocduflow_6C[t] = q6C[t]* (q7C[t] + q8C[t]);
subject to AT_MOST_TWO_TANKS_7C {t in 0..T-1}:
twocduflow_7C[t] = q7C[t]* q8C[t];
```

```
subject to AT_MOST_TWO_TANKS_2A {t in 0..T-1}:
twocduflow_2A[t] = q2A[t]* (q3A[t] + q4A[t] + q5A[t]);
subject to AT_MOST_TWO_TANKS_3A {t in 0..T-1}:
twocduflow_3A[t] = q3A[t]* (q4A[t] + q5A[t]);
subject to AT_MOST_TWO_TANKS_4A {t in 0..T-1}:
twocduflow_4A[t] = q4A[t]* q5A[t];
subject to AT_MOST_TWO_TANKS_2B {t in 0..T-1}:
twocduflow_2B[t] = q2B[t]* (q3B[t] + q4B[t] + q5B[t]);
subject to AT_MOST_TWO_TANKS_3B {t in 0..T-1}:
twocduflow_3B[t] = q3B[t]* (q4B[t] + q5B[t]);
subject to AT_MOST_TWO_TANKS_4B {t in 0..T-1}:
twocduflow_4B[t] = q4B[t]* q5B[t];

subject to CDU_TANK_FLOW_DAY1_T1_C {t in 1..24}:
q1C[t] <= q1C[t-1];
subject to CDU_TANK_FLOW_DAY1_T6_C {t in 1..24}:
q6C[t] <= q6C[t-1];
subject to CDU_TANK_FLOW_DAY1_T7_C {t in 1..24}:
q7C[t] <= q7C[t-1];
subject to CDU_TANK_FLOW_DAY1_T8_C {t in 1..24}:
q8C[t] <= q8C[t-1];
subject to CDU_TANK_FLOW_DAY2_T1_C {t in 26..48}:
q1C[t] <= q1C[t-1];
subject to CDU_TANK_FLOW_DAY2_T6_C {t in 26..48}:
q6C[t] <= q6C[t-1];
subject to CDU_TANK_FLOW_DAY2_T7_C {t in 26..48}:
q7C[t] <= q7C[t-1];
subject to CDU_TANK_FLOW_DAY2_T8_C {t in 26..48}:
q8C[t] <= q8C[t-1];
subject to CDU_TANK_FLOW_DAY3_T1_C {t in 50..T-1}:
q1C[t] <= q1C[t-1];
subject to CDU_TANK_FLOW_DAY3_T6_C {t in 50..T-1}:
q6C[t] <= q6C[t-1];
subject to CDU_TANK_FLOW_DAY3_T7_C {t in 50..T-1}:
```

```
q7C[t] <= q7C[t-1];
subject to CDU_TANK_FLOW_DAY3_T8_C {t in 50..T-1}:
q8C[t] <= q8C[t-1];

subject to CDU_TANK_FLOW_DAY1_T2_A {t in 1..24}:
q2A[t] <= q2A[t-1];
subject to CDU_TANK_FLOW_DAY1_T3_A {t in 1..24}:
q3A[t] <= q3A[t-1];
subject to CDU_TANK_FLOW_DAY1_T4_A {t in 1..24}:
q4A[t] <= q4A[t-1];
subject to CDU_TANK_FLOW_DAY1_T5_A {t in 1..24}:
q5A[t] <= q5A[t-1];
subject to CDU_TANK_FLOW_DAY2_T2_A {t in 26..48}:
q2A[t] <= q2A[t-1];
subject to CDU_TANK_FLOW_DAY2_T3_A {t in 26..48}:
q3A[t] <= q3A[t-1];
subject to CDU_TANK_FLOW_DAY2_T4_A {t in 26..48}:
q4A[t] <= q4A[t-1];
subject to CDU_TANK_FLOW_DAY2_T5_A {t in 26..48}:
q5A[t] <= q5A[t-1];
subject to CDU_TANK_FLOW_DAY3_T2_A {t in 50..T-1}:
q2A[t] <= q2A[t-1];
subject to CDU_TANK_FLOW_DAY3_T3_A {t in 50..T-1}:
q3A[t] <= q3A[t-1];
subject to CDU_TANK_FLOW_DAY3_T4_A {t in 50..T-1}:
q4A[t] <= q4A[t-1];
subject to CDU_TANK_FLOW_DAY3_T5_A {t in 50..T-1}:
q5A[t] <= q5A[t-1];

subject to CDU_TANK_FLOW_DAY1_T2_B {t in 1..24}:
q2B[t] <= q2B[t-1];
subject to CDU_TANK_FLOW_DAY1_T3_B {t in 1..24}:
q3B[t] <= q3B[t-1];
subject to CDU_TANK_FLOW_DAY1_T4_B {t in 1..24}:
```

```
q4B[t] <= q4B[t-1];
subject to CDU_TANK_FLOW_DAY1_T5_B {t in 1..24}:
q5B[t] <= q5B[t-1];
subject to CDU_TANK_FLOW_DAY2_T2_B {t in 26..48}:
q2B[t] <= q2B[t-1];
subject to CDU_TANK_FLOW_DAY2_T3_B {t in 26..48}:
q3B[t] <= q3B[t-1];
subject to CDU_TANK_FLOW_DAY2_T4_B {t in 26..48}:
q4B[t] <= q4B[t-1];
subject to CDU_TANK_FLOW_DAY2_T5_B {t in 26..48}:
q5B[t] <= q5B[t-1];
subject to CDU_TANK_FLOW_DAY3_T2_B {t in 50..T-1}:
q2B[t] <= q2B[t-1];
subject to CDU_TANK_FLOW_DAY3_T3_B {t in 50..T-1}:
q3B[t] <= q3B[t-1];
subject to CDU_TANK_FLOW_DAY3_T4_B {t in 50..T-1}:
q4B[t] <= q4B[t-1];
subject to CDU_TANK_FLOW_DAY3_T5_B {t in 50..T-1}:
q5B[t] <= q5B[t-1];

#####################################################
# FEED KEY COMPONENT LIMITS
#####################################################
var feed_key_A {0..T-1} <= feed_key_max_A * feedflow;
var feed_key_B {0..T-1} <= feed_key_max_B * feedflow;
var feed_key_C {0..T-1} <= feed_key_max_C * feedflow;

subject to CDU_KEY_A_T_MAX {t in 0..T-1}: feed_key_A[t] <=
(q1A[t]+q2A[t]+q3A[t]+q4A[t]+q5A[t]+q6A[t]+q7A[t]+q8A[t]) * feed_key_max_A;
subject to CDU_KEY_B_T_MAX {t in 0..T-1}: feed_key_B[t] <=
(q1B[t]+q2B[t]+q3B[t]+q4B[t]+q5B[t]+q6B[t]+q7B[t]+q8B[t]) * feed_key_max_B;
subject to CDU_KEY_C_T_MAX {t in 0..T-1}: feed_key_C[t] <=
(q1C[t]+q2C[t]+q3C[t]+q4C[t]+q5C[t]+q6C[t]+q7C[t]+q8C[t]) * feed_key_max_C;
```

```
subject to MIN_FEED_CRUDE_A {t in 0..T-1}: feed_key_A[t]  >=
feed_key_min_A * min_feedflow;
subject to MIN_FEED_CRUDE_B {t in 0..T-1}: feed_key_B[t]  >=
feed_key_min_B * min_feedflow;
subject to MIN_FEED_CRUDE_C {t in 0..T-1}: feed_key_C[t]  >=
feed_key_min_C * min_feedflow;

subject to CDU_KEY_A_T_MIN {t in 0..T-1}: feed_key_A[t] >=
(q1A[t]+q2A[t]+q3A[t]+q4A[t]+q5A[t]+q6A[t]+q7A[t]+q8A[t]) * feed_key_min_A;
subject to CDU_KEY_B_T_MIN {t in 0..T-1}: feed_key_B[t] >=
(q1B[t]+q2B[t]+q3B[t]+q4B[t]+q5B[t]+q6B[t]+q7B[t]+q8B[t]) * feed_key_min_B;
subject to CDU_KEY_C_T_MIN {t in 0..T-1}: feed_key_C[t] >=
(q1C[t]+q2C[t]+q3C[t]+q4C[t]+q5C[t]+q6C[t]+q7C[t]+q8C[t]) * feed_key_min_C;

# CDU A => T2-T5
subject to CDU_KEY_A_T0: feed_key_A[0] =
(q1A[0]*ini_key_T1 + q2A[0]*ini_key_T2 + q3A[0]*ini_key_T3 + q4A[0]*ini_key_T4 +
q5A[0]*ini_key_T5 + q6A[0]*ini_key_T6 + q7A[0]*ini_key_T7 + q8A[0]*ini_key_T8);
subject to CDU_KEY_A_T {t in 1..T-1}: feed_key_A[t] =
(q1A[t]*key_T1[t] + q2A[t]*key_T2[t] + q3A[t]*key_T3[t] + q4A[t]*key_T4[t] +
q5A[t]*key_T5[t] + q6A[t]*key_T6[t] + q7A[t]*key_T7[t] + q8A[t]*key_T8[t]);
# CDU B => T1, T6-T8
subject to CDU_KEY_B_T0: feed_key_B[0] =
(q1B[0]*ini_key_T1 + q2B[0]*ini_key_T2 + q3B[0]*ini_key_T3 + q4B[0]*ini_key_T4 +
q5B[0]*ini_key_T5 + q6B[0]*ini_key_T6 + q7B[0]*ini_key_T7 + q8B[0]*ini_key_T8);
subject to CDU_KEY_B_T {t in 1..T-1}: feed_key_B[t] =
(q1B[t]*key_T1[t] + q2B[t]*key_T2[t] + q3B[t]*key_T3[t] + q4B[t]*key_T4[t] +
q5B[t]*key_T5[t] + q6B[t]*key_T6[t] + q7B[t]*key_T7[t] + q8B[t]*key_T8[t]);

# CDU C => T1, T6-T8
subject to CDU_KEY_C_T0: feed_key_C[0] =
(q1C[0]*ini_key_T1 + q2C[0]*ini_key_T2 + q3C[0]*ini_key_T3 + q4C[0]*ini_key_T4 +
q5C[0]*ini_key_T5 + q6C[0]*ini_key_T6 + q7C[0]*ini_key_T7 + q8C[0]*ini_key_T8);
subject to CDU_KEY_C_T {t in 1..T-1}: feed_key_C[t] =
```

```
(q1C[t]*key_T1[t] + q2C[t]*key_T2[t] + q3C[t]*key_T3[t] + q4C[t]*key_T4[t] +
q5C[t]*key_T5[t] + q6C[t]*key_T6[t] + q7C[t]*key_T7[t] + q8C[t]*key_T8[t]);


#############################################################################
# Pipeline -> one Tank
# by definition, when one parcel is unloaded, the others have zero flow
#############################################################################
subject to ONE_PIPE_FLOW_1 {t in 0..T-1}:
onepipeflow_1[t] = q11[t]* (q12[t] + q13[t] + q14[t] +
q15[t] + q16[t] + q17[t] + q18[t]);
subject to ONE_PIPE_FLOW_2 {t in 0..T-1}:
onepipeflow_2[t] = q12[t]* (q13[t] + q14[t] + q15[t] +
q16[t] + q17[t] + q18[t]);
subject to ONE_PIPE_FLOW_3 {t in 0..T-1}:
onepipeflow_3[t] = q13[t]* (q14[t] + q15[t] + q16[t] + q17[t] + q18[t]);
subject to ONE_PIPE_FLOW_4 {t in 0..T-1}:
onepipeflow_4[t] = q14[t]* (q15[t] + q16[t] + q17[t] + q18[t]);
subject to ONE_PIPE_FLOW_5 {t in 0..T-1}:
onepipeflow_5[t] = q15[t]* (q16[t] + q17[t] + q18[t]);
subject to ONE_PIPE_FLOW_6 {t in 0..T-1}:
onepipeflow_6[t] = q16[t]* (q17[t] + q18[t]);
subject to ONE_PIPE_FLOW_7 {t in 0..T-1}:
onepipeflow_7[t] = q17[t]* (q18[t]);
subject to ONE_PIPE_FLOW_TOTAL1 {t in 0..T-1}:
onepipeflow_1[t] = 0; #
subject to ONE_PIPE_FLOW_TOTAL2 {t in 0..T-1}:
onepipeflow_2[t] = 0; #
subject to ONE_PIPE_FLOW_TOTAL3 {t in 0..T-1}:
onepipeflow_3[t] = 0; #
subject to ONE_PIPE_FLOW_TOTAL4 {t in 0..T-1}:
onepipeflow_4[t] = 0; #
subject to ONE_PIPE_FLOW_TOTAL5 {t in 0..T-1}:
onepipeflow_5[t] = 0; #
```

```
subject to ONE_PIPE_FLOW_TOTAL6 {t in 0..T-1}:
onepipeflow_6[t] = 0; #
subject to ONE_PIPE_FLOW_TOTAL7 {t in 0..T-1}:
onepipeflow_7[t] = 0; #

#####################################################
# Objective Function - components
#####################################################
var switchover >= 0;
var twocdu >= 0;
var feed_diff >= 0;
var onepipe >= 0;
var settling >= 0;
param netback = 0;

param max_feed_diff;
let max_feed_diff := cdu_vol;
subject to MAX_FEED_DIFF: feed_diff <= max_feed_diff;

param max_onepipe;
let max_onepipe := 1;
subject to MAX_ONEPIPE: onepipe <= max_onepipe;

param max_settling;
let max_settling := 1;
subject to MAX_SETTLING: settling <= max_settling;

param max_switchover;
let max_switchover := 1000;
subject to MAX_SWITCHOVER: switchover <= max_switchover;

param max_twocdu;
let max_twocdu := 100;
subject to MAX_TWOCDU: twocdu <= max_twocdu;
```

```
subject to SWITCHOVER_COST:
switchover = sum {t in 1..T-1}
(
(q11[t-1] - q11[t])^2
+ (q12[t-1] - q12[t])^2
+ (q13[t-1] - q13[t])^2
+ (q14[t-1] - q14[t])^2
+ (q15[t-1] - q15[t])^2
+ (q16[t-1] - q16[t])^2
+ (q1C[t-1] - q1C[t])^2
+ (q2A[t-1] - q2A[t])^2
+ (q3A[t-1] - q3A[t])^2
+ (q4A[t-1] - q4A[t])^2
+ (q5A[t-1] - q5A[t])^2
+ (q2B[t-1] - q2B[t])^2
+ (q3B[t-1] - q3B[t])^2
+ (q4B[t-1] - q4B[t])^2
+ (q5B[t-1] - q5B[t])^2
+ (q6C[t-1] - q6C[t])^2
+ (q7C[t-1] - q7C[t])^2
+ (q8C[t-1] - q8C[t])^2
)/q_max_pipe^2;

subject to FEED_COST: feed_diff = 0;
subject to ONEPIPE_COST: onepipe = 0;
subject to SETTLING_COST: settling = 0;
subject to TWOCDU_COST:
twocdu = sum {t in 0..T-1}
(twocduflow_1C[t] + twocduflow_2A[t] + twocduflow_3A[t] +
twocduflow_4A[t] + twocduflow_2B[t] + twocduflow_3B[t] +
twocduflow_4B[t] + twocduflow_6C[t] + twocduflow_7C[t])/feedflow^2;

param W_SW;
```

```
let W_SW := 1;
param W_FEED;
let W_FEED := 1;
param W_PIPE;
let W_PIPE := 1;
param W_CDU;
let W_CDU := 1;
param W_SET;
let W_SET := 1;
param W_PENALTY;
let W_PENALTY := 0;
param W_COST;
let W_COST := 0;
var cost;
subject to COST: cost =  W_SW * switchover +  W_FEED * feed_diff;
var penalty;
subject to PENALTY:
penalty =  W_PIPE * onepipe + W_CDU * twocdu + W_SET * settling;
minimize obj: W_COST * cost + W_PENALTY * penalty;
```

A.2.2 Caso de Teste 7

```
#####################################################
# Example 2 - Typical refinery
# Based on the example #2 from
# Pan et al. (2009)
# Chemical Engineering Science 64, 965-983
#
# Originally from Reddy et al. (2004)
# Chemical Engineering Science 59, 1325-1341
#
#####################################################
#####################################################
```

```
# Parameters
#####################################################
param dt;
let dt := 1.0;
# No of periods 80h
param T = 80.0/dt;
#####################################################################
# 8 tanks (T1-18), 3 CDU (CDU1-CDU3), 2 classes of Oil (cl1, cl2)
# Tanks T1, T6-T8 and CDU3 handle class1 crude oil (C1-C4),
# while the rest handle class2 crude oil (C5-C8)
# ONLY ONE KEY COMPONENT IS CONSIDERED TO CARACTERIZE THE CDU FEED:
# SULFUR OR METAL
#####################################################################
# Max Tank volume (m3)
param max_vol1 = 570.000;
param max_vol2 = 980.000;
# Min Tank volume (m3)
param min_vol1 = 60.000;
param min_vol2 = 110.000;
# Initial volumes
param ini_vol1 = 350.000; # ini vol of tank1
param ini_vol2 = 400.000; # ini vol of tank2
param ini_vol3 = 350.000; # ini vol of tank3
param ini_vol4 = 950.000; # ini vol of tank4
param ini_vol5 = 300.000; # ini vol of tank5
param ini_vol6 = 240.000; # ini vol of tank6
param ini_vol7 = 120.000; # ini vol of tank7
param ini_vol8 = 550.000; # ini vol of tank8
#############################################
# KEY COMPONENT (SULFUR OR METAL)
#############################################
# Initial composition (8 crudes: C1, C2, C3, C4,
# C5, C6, C7, C8, each with a different key component
# concentration and net back (profit))
```

```
param key_c1 = 0.20;
param key_c2 = 0.25;
param key_c3 = 0.15;
param key_c4 = 0.60;
param key_c5 = 1.20;
param key_c6 = 1.30;
param key_c7 = 0.90;
param key_c8 = 1.50;
param profit_c1 = 1.50;
param profit_c2 = 1.70;
param profit_c3 = 1.50;
param profit_c4 = 1.60;
param profit_c5 = 1.45;
param profit_c6 = 1.60;
param profit_c7 = 1.55;
param profit_c8 = 1.60;
param ini_cru11 = 50/350; # crude 1 - tank 1
param ini_cru21 = 100/350; # crude 2 - tank 1
param ini_cru31 = 150/350; # crude 3 - tank 1
param ini_cru41 = 50/350; # crude 4 - tank 1
param ini_key_T1 = ini_cru11 * key_c1 + ini_cru21 * key_c2 +
ini_cru31 * key_c3 + ini_cru41 * key_c4;
param ini_profit_T1 = ini_cru11 * profit_c1 + ini_cru21 * profit_c2 +
ini_cru31 * profit_c3 + ini_cru41 * profit_c4;
param ini_cru52 = 200/400; # crude 5 - tank 2
param ini_cru62 = 0/400; # crude 6 - tank 2
param ini_cru72 = 50/400; # crude 7 - tank 2
param ini_cru82 = 150/400; # crude 8 - tank 2
param ini_key_T2 = ini_cru52 * key_c5 + ini_cru62 * key_c6 +
ini_cru72 * key_c7 + ini_cru82 * key_c8;
param ini_profit_T2 = ini_cru52 * profit_c5 + ini_cru62 * profit_c6 +
ini_cru72 * profit_c7 + ini_cru82 * profit_c8;
param ini_cru53 = 100/350; # crude 5 - tank 3
param ini_cru63 = 100/350; # crude 6 - tank 3
```

```
param ini_cru73 = 50/350; # crude 7 - tank 3
param ini_cru83 = 100/350; # crude 8 - tank 3
param ini_key_T3 = ini_cru53 * key_c5 + ini_cru63 * key_c6 +
ini_cru73 * key_c7 + ini_cru83 * key_c8;
param ini_profit_T3 = ini_cru53 * profit_c5 + ini_cru63 * profit_c6 +
ini_cru73 * profit_c7 + ini_cru83 * profit_c8;
param ini_cru54 = 200/950; # crude 5 - tank 4
param ini_cru64 = 250/950; # crude 6 - tank 4
param ini_cru74 = 200/950; # crude 7 - tank 4
param ini_cru84 = 300/950; # crude 8 - tank 4
param ini_key_T4 = ini_cru54 * key_c5 + ini_cru64 * key_c6 +
ini_cru74 * key_c7 + ini_cru84 * key_c8;
param ini_profit_T4 = ini_cru54 * profit_c5 + ini_cru64 * profit_c6 +
ini_cru74 * profit_c7 + ini_cru84 * profit_c8;
param ini_cru55 = 100/300; # crude 5 - tank 5
param ini_cru65 = 100/300; # crude 6 - tank 5
param ini_cru75 = 50/300; # crude 7 - tank 5
param ini_cru85 = 50/300; # crude 8 - tank 5
param ini_key_T5 = ini_cru55 * key_c5 + ini_cru65 * key_c6 +
ini_cru75 * key_c7 + ini_cru85 * key_c8;
param ini_profit_T5 = ini_cru55 * profit_c5 + ini_cru65 * profit_c6 +
ini_cru75 * profit_c7 + ini_cru85 * profit_c8;
param ini_cru16 = 30/240; # crude 1 - tank 6
param ini_cru26 = 30/240; # crude 2 - tank 6
param ini_cru36 = 150/240; # crude 3 - tank 6
param ini_cru46 = 30/240; # crude 4 - tank 6
param ini_key_T6 = ini_cru16 * key_c1 + ini_cru26 * key_c2 +
ini_cru36 * key_c3 + ini_cru46 * key_c4;
param ini_profit_T6 = ini_cru16 * profit_c1 + ini_cru26 * profit_c2 +
ini_cru36 * profit_c3 + ini_cru46 * profit_c4;
param ini_cru17 = 30/120; # crude 1 - tank 7
param ini_cru27 = 30/120; # crude 2 - tank 7
param ini_cru37 = 50/120; # crude 3 - tank 7
param ini_cru47 = 10/120; # crude 4 - tank 7
```

```
param ini_key_T7 = ini_cru17 * key_c1 + ini_cru27 * key_c2 +
ini_cru37 * key_c3 + ini_cru47 * key_c4;
param ini_profit_T7 = ini_cru17 * profit_c1 + ini_cru27 * profit_c2 +
ini_cru37 * profit_c3 + ini_cru47 * profit_c4;
param ini_cru18 = 150/550; # crude 1 - tank 8
param ini_cru28 = 100/550; # crude 2 - tank 8
param ini_cru38 = 210/550; # crude 3 - tank 8
param ini_cru48 = 90/550; # crude 4 - tank 8
param ini_key_T8 = ini_cru18 * key_c1 + ini_cru28 * key_c2 +
ini_cru38 * key_c3 + ini_cru48 * key_c4;
param ini_profit_T8 = ini_cru18 * profit_c1 + ini_cru28 * profit_c2 +
ini_cru38 * profit_c3 + ini_cru48 * profit_c4;
# max pipeline flow 400 kbbl/8h = 50 kbbl/h
param q_max_pipe = 50.0;
# Ship final volume
param final_parcelvol = 0.0;
# parcel volumes
param ini_parcel1 = 10.000;     # 10 kbbl C2
param ini_parcel2 = 500.000;  # 500 kbbl C4
param ini_parcel3 = 500.000;  # 500 kbbl C3
param ini_parcel4 = 440.000;  # 440 kbbl C5
#arrival crude parcels
param arrival1 = ceil(8/dt);   # t = 8h
param arrival2 = ceil(9/dt);  # t = 9h
param arrival3 = ceil(19/dt);  # t = 19h
param arrival4 = ceil(29/dt);  # t = 29h
#end crude parcels
# t = 8.2h -> 9h
param end1 = max(ceil(8.2/dt), arrival1 + 1);
# t = 9h + 10h = 19h
param end2 = max(ceil(19/dt), arrival2 + 1);
# t = 19h + 10h = 29h
param end3 = max(ceil(29/dt), arrival3 + 1);
# t = 29h + 8.8h = 37.8h -> 38
```

```
param end4 = max(ceil(37.8/dt), arrival4 + 1);
#crude parcels
param key_parcel1 = key_c2;   # 100% C2
param key_parcel2 = key_c4;  # 100% C4
param key_parcel3 = key_c3;  # 100% C3
param key_parcel4 = key_c5;  # 100% C5
param profit_parcel1 = profit_c2;   # 100% C2
param profit_parcel2 = profit_c4;  # 100% C4
param profit_parcel3 = profit_c3;  # 100% C3
param profit_parcel4 = profit_c5;  # 100% C5
# resting/settling time
param settling_time = ceil(8/dt); # 8h to settle brine
#CDU feed flow 48 kbbl/8h = 6 kbbl/h
param feedflow = 6.0;
param cdu_vol1 = 375;
param cdu_vol = 400;
param min_feedflow = 2.0;
# CDU feed key component limits
param feed_key_max_A = 1.35;
param feed_key_max_B = 1.30;
param feed_key_max_C = 0.35;
param feed_key_min_A = 0.10;
param feed_key_min_B = 0.10;
param feed_key_min_C = 0.10;
#min tank flow
param min_q = 0;
# penalty variables
var twocduflow_1C {0..T-1} ;
var twocduflow_2A {0..T-1} ;
var twocduflow_3A {0..T-1} ;
var twocduflow_4A {0..T-1} ;
var twocduflow_2B {0..T-1} ;
var twocduflow_3B {0..T-1} ;
var twocduflow_4B {0..T-1} ;
```

```
var twocduflow_6C {0..T-1} ;
var twocduflow_7C {0..T-1} ;
var settling_1 {1..T-1} ;
var settling_2 {1..T-1} ;
var settling_3 {1..T-1} ;
var settling_4 {1..T-1} ;
var settling_5 {1..T-1} ;
var settling_6 {1..T-1} ;
var settling_7 {1..T-1} ;
var settling_8 {1..T-1} ;
var onepipeflow_1 {0..T-1} ;
var onepipeflow_2 {0..T-1} ;
var onepipeflow_3 {0..T-1} ;
var onepipeflow_4 {0..T-1} ;
var onepipeflow_5 {0..T-1} ;
var onepipeflow_6 {0..T-1} ;
var onepipeflow_7 {0..T-1} ;
##################################################
# Control variables
##################################################
# transferred volume from pipeline VLCC to T1
var q11 {0..T-1} <= q_max_pipe, := 0;
# transferred volume from pipeline VLCC to T2
var q12 {0..T-1} <= q_max_pipe, := 0;
# transferred volume from pipeline VLCC to T3
var q13 {0..T-1} <= q_max_pipe, := 0;
# transferred volume from pipeline VLCC to T4
var q14 {0..T-1} <= q_max_pipe, := 0;
# transferred volume from pipeline VLCC to T5
var q15 {0..T-1} <= q_max_pipe, := 0;
# transferred volume from pipeline VLCC to T6
var q16 {0..T-1} <= q_max_pipe, := 0;
# transferred volume from pipeline VLCC to T7
var q17 {0..T-1} <= q_max_pipe, := 0;
```

```
# transferred volume from pipeline VLCC to T8
var q18 {0..T-1} <= q_max_pipe, := 0;
subject to MIN_Q11 {t in 0..T-1}: q11[t]  >= min_q;
subject to MIN_Q12 {t in 0..T-1}: q12[t]  >= min_q;
subject to MIN_Q13 {t in 0..T-1}: q13[t]  >= min_q;
subject to MIN_Q14 {t in 0..T-1}: q14[t]  >= min_q;
subject to MIN_Q15 {t in 0..T-1}: q15[t]  >= min_q;
subject to MIN_Q16 {t in 0..T-1}: q16[t]  >= min_q;
subject to MIN_Q17 {t in 0..T-1}: q17[t]  >= min_q;
subject to MIN_Q18 {t in 0..T-1}: q18[t]  >= min_q;
# transferred volume from T1 to CDU1
var q1A {0..T-1} <= feedflow, := 0;
# transferred volume from T2 to CDU1
var q2A {0..T-1} <= feedflow, := 0;
# transferred volume from T3 to CDU1
var q3A {0..T-1} <= feedflow, := 0;
# transferred volume from T4 to CDU1
var q4A {0..T-1} <= feedflow, := 0;
# transferred volume from T5 to CDU1
var q5A {0..T-1} <= feedflow, := 0;
# transferred volume from T6 to CDU1
var q6A {0..T-1} <= feedflow, := 0;
# transferred volume from T7 to CDU1
var q7A {0..T-1} <= feedflow, := 0;
# transferred volume from T8 to CDU1
var q8A {0..T-1} <= feedflow, := 0;
subject to MIN_Q1A {t in 0..T-1}: q1A[t]  = min_q;
subject to MIN_Q2A {t in 0..T-1}: q2A[t]  >= min_q;
subject to MIN_Q3A {t in 0..T-1}: q3A[t]  >= min_q;
subject to MIN_Q4A {t in 0..T-1}: q4A[t]  >= min_q;
subject to MIN_Q5A {t in 0..T-1}: q5A[t]  >= min_q;
subject to MIN_Q6A {t in 0..T-1}: q6A[t]  = min_q;
subject to MIN_Q7A {t in 0..T-1}: q7A[t]  = min_q;
subject to MIN_Q8A {t in 0..T-1}: q8A[t]  = min_q;
```

```
# transferred volume from T1 to CDU2
var q1B {0..T-1} <= feedflow, := 0;
# transferred volume from T2 to CDU2
var q2B {0..T-1} <= feedflow, := 0;
# transferred volume from T3 to CDU2
var q3B {0..T-1} <= feedflow, := 0;
# transferred volume from T4 to CDU2
var q4B {0..T-1} <= feedflow, := 0;
# transferred volume from T5 to CDU2
var q5B {0..T-1} <= feedflow, := 0;
# transferred volume from T6 to CDU2
var q6B {0..T-1} <= feedflow, := 0;
# transferred volume from T7 to CDU2
var q7B {0..T-1} <= feedflow, := 0;
# transferred volume from T8 to CDU2
var q8B {0..T-1} <= feedflow, := 0;
subject to MIN_Q1B {t in 0..T-1}: q1B[t]  = min_q;
subject to MIN_Q2B {t in 0..T-1}: q2B[t]  >= min_q;
subject to MIN_Q3B {t in 0..T-1}: q3B[t]  >= min_q;
subject to MIN_Q4B {t in 0..T-1}: q4B[t]  >= min_q;
subject to MIN_Q5B {t in 0..T-1}: q5B[t]  >= min_q;
subject to MIN_Q6B {t in 0..T-1}: q6B[t]  = min_q;
subject to MIN_Q7B {t in 0..T-1}: q7B[t]  = min_q;
subject to MIN_Q8B {t in 0..T-1}: q8B[t]  = min_q;
# transferred volume from T1 to CDU3
var q1C {0..T-1} <= feedflow, := 0;
# transferred volume from T2 to CDU3
var q2C {0..T-1} <= feedflow, := 0;
# transferred volume from T3 to CDU3
var q3C {0..T-1} <= feedflow, := 0;
# transferred volume from T4 to CDU3
var q4C {0..T-1} <= feedflow, := 0;
# transferred volume from T5 to CDU3
var q5C {0..T-1} <= feedflow, := 0;
```

```
# transferred volume from T6 to CDU3
var q6C {0..T-1} <= feedflow, := 0;
# transferred volume from T7 to CDU3
var q7C {0..T-1} <= feedflow, := 0;
# transferred volume from T8 to CDU3
var q8C {0..T-1} <= feedflow, := 0;
subject to MIN_Q1C {t in 0..T-1}: q1C[t]  >= min_q;
subject to MIN_Q2C {t in 0..T-1}: q2C[t]  = min_q;
subject to MIN_Q3C {t in 0..T-1}: q3C[t]  = min_q;
subject to MIN_Q4C {t in 0..T-1}: q4C[t]  = min_q;
subject to MIN_Q5C {t in 0..T-1}: q5C[t]  = min_q;
subject to MIN_Q6C {t in 0..T-1}: q6C[t]  >= min_q;
subject to MIN_Q7C {t in 0..T-1}: q7C[t]  >= min_q;
subject to MIN_Q8C {t in 0..T-1}: q8C[t]  >= min_q;
#####################################################
# State Variables
#####################################################
var vol1 {1..T} <= max_vol1, := ini_vol1;
var vol2 {1..T} <= max_vol1, := ini_vol2;
var vol3 {1..T} <= max_vol1, := ini_vol3;
var vol4 {1..T} <= max_vol2, := ini_vol4;
var vol5 {1..T} <= max_vol2, := ini_vol5;
var vol6 {1..T} <= max_vol1, := ini_vol6;
var vol7 {1..T} <= max_vol1, := ini_vol7;
var vol8 {1..T} <= max_vol1, := ini_vol8;
subject to MIN_VOL1 {t in 1..T}: vol1[t]  >= min_vol1;
subject to MIN_VOL2 {t in 1..T}: vol2[t]  >= min_vol1;
subject to MIN_VOL3 {t in 1..T}: vol3[t]  >= min_vol1;
subject to MIN_VOL4 {t in 1..T}: vol4[t]  >= min_vol2;
subject to MIN_VOL5 {t in 1..T}: vol5[t]  >= min_vol2;
subject to MIN_VOL6 {t in 1..T}: vol6[t]  >= min_vol1;
subject to MIN_VOL7 {t in 1..T}: vol7[t]  >= min_vol1;
subject to MIN_VOL8 {t in 1..T}: vol8[t]  >= min_vol1;
var parcel1 {arrival1..end1} <= ini_parcel1, := ini_parcel1;
```

```
var parcel2 {arrival2..end2} <= ini_parcel2, := ini_parcel2;
var parcel3 {arrival3..end3} <= ini_parcel3, := ini_parcel3;
var parcel4 {arrival4..end4} <= ini_parcel4, := ini_parcel4;
var parcel5;
var parcel6;
var parcel7;
var parcel8;
subject to MIN_PARCEL21 {t in arrival1..end1}:
parcel1[t]  >= final_parcelvol;
subject to MIN_PARCEL22 {t in arrival2..end2}:
parcel2[t]  >= final_parcelvol;
subject to MIN_PARCEL23 {t in arrival3..end3}:
parcel3[t]  >= final_parcelvol;
subject to MIN_PARCEL24 {t in arrival4..end4}:
parcel4[t]  >= final_parcelvol;
#########################################################
# Tank Composition
#########################################################
var key_T1 {1..T} := ini_key_T1;        # key in T1
var key_T2 {1..T} := ini_key_T2;        # key in T2
var key_T3 {1..T} := ini_key_T3;        # key in T3
var key_T4 {1..T} := ini_key_T4;        # key in T4
var key_T5 {1..T} := ini_key_T5;        # key in T5
var key_T6 {1..T} := ini_key_T6;        # key in T6
var key_T7 {1..T} := ini_key_T7;        # key in T7
var key_T8 {1..T} := ini_key_T8;        # key in T8
#####################################################
# State Equations - volumes
#####################################################
subject to INITIAL_STATE1: vol1[1] = ini_vol1 +
dt*(q11[0] - q1A[0] - q1B[0] - q1C[0]);
subject to STATE_EQ1 {t in 2..T}: vol1[t] = vol1[t-1] +
dt*(q11[t-1] - q1A[t-1] - q1B[t-1] - q1C[t-1]);
subject to INITIAL_STATE2: vol2[1] = ini_vol2 +
```

```
dt*(q12[0] - q2A[0] - q2B[0] - q2C[0]);
subject to STATE_EQ2 {t in 2..T}: vol2[t] = vol2[t-1] +
dt*(q12[t-1] - q2A[t-1] - q2B[t-1] - q2C[t-1]);
subject to INITIAL_STATE3: vol3[1] = ini_vol3 +
dt*(q13[0] - q3A[0] - q3B[0] - q3C[0]);
subject to STATE_EQ3 {t in 2..T}: vol3[t] = vol3[t-1] +
dt*(q13[t-1] - q3A[t-1] - q3B[t-1] - q3C[t-1]);
subject to INITIAL_STATE4: vol4[1] = ini_vol4 +
dt*(q14[0] - q4A[0] - q4B[0] - q4C[0]);
subject to STATE_EQ4 {t in 2..T}: vol4[t] = vol4[t-1] +
dt*(q14[t-1] - q4A[t-1] - q4B[t-1] - q4C[t-1]);
subject to INITIAL_STATE5: vol5[1] = ini_vol5 +
dt*(q15[0] - q5A[0] - q5B[0] - q5C[0]);
subject to STATE_EQ5 {t in 2..T}: vol5[t] = vol5[t-1] +
dt*(q15[t-1] - q5A[t-1] - q5B[t-1] - q5C[t-1]);
subject to INITIAL_STATE6: vol6[1] = ini_vol6 +
dt*(q16[0] - q6A[0] - q6B[0] - q6C[0]);
subject to STATE_EQ6 {t in 2..T}: vol6[t] = vol6[t-1] +
dt*(q16[t-1] - q6A[t-1] - q6B[t-1] - q6C[t-1]);
subject to INITIAL_STATE7: vol7[1] = ini_vol7 +
dt*(q17[0] - q7A[0] - q7B[0] - q7C[0]);
subject to STATE_EQ7 {t in 2..T}: vol7[t] = vol7[t-1] +
dt*(q17[t-1] - q7A[t-1] - q7B[t-1] - q7C[t-1]);
subject to INITIAL_STATE8: vol8[1] = ini_vol8 +
dt*(q18[0] - q8A[0] - q8B[0] - q8C[0]);
subject to STATE_EQ8 {t in 2..T}: vol8[t] = vol8[t-1] +
dt*(q18[t-1] - q8A[t-1] - q8B[t-1] - q8C[t-1]);
#########################################################
# State Equations - Crude parcel volumes
#########################################################
#C2 only in T1, T6-T8
subject to INITIAL_PARCEL1: parcel1[arrival1] = ini_parcel1;
subject to STATE_PARCEL1 {t in (arrival1 + 1)..(end1)}:
parcel1[t] = parcel1[t-1] - dt*(q11[t-1] + q16[t-1] + q17[t-1] + q18[t-1]);
```

```
subject to STATE_PARCEL1_C2 {t in (arrival1 + 1)..(end1)}:
q12[t-1] + q13[t-1] + q14[t-1] + q15[t-1] = 0;
subject to FINAL_PARCEL_1: parcel1[end1] = final_parcelvol;
#C3 only in T1, T6-T8
subject to INITIAL_PARCEL2: parcel2[arrival2] = ini_parcel2;
subject to STATE_PARCEL2 {t in (arrival2 + 1)..(end2)}:
parcel2[t] = parcel2[t-1] - dt*(q11[t-1] + q16[t-1] + q17[t-1] + q18[t-1]);
subject to STATE_PARCEL2_C3 {t in (arrival2 + 1)..(end2)}:
q12[t-1] + q13[t-1] + q14[t-1] + q15[t-1] = 0;
subject to FINAL_PARCEL_2: parcel2[end2] = final_parcelvol;
#C4 only in T1, T6-T8
subject to INITIAL_PARCEL3: parcel3[arrival3] = ini_parcel3;
subject to STATE_PARCEL3 {t in (arrival3 + 1)..(end3)}:
parcel3[t] = parcel3[t-1] - dt*(q11[t-1] + q16[t-1] + q17[t-1] + q18[t-1]);
subject to STATE_PARCEL3_C4 {t in (arrival3 + 1)..(end3)}:
q12[t-1] + q13[t-1] + q14[t-1] + q15[t-1] = 0;
subject to FINAL_PARCEL_3: parcel3[end3] = final_parcelvol;
#C5 only in T2-T5
subject to INITIAL_PARCEL4: parcel4[arrival4] = ini_parcel4;
subject to STATE_PARCEL4 {t in (arrival4 + 1)..(end4)}:
parcel4[t] = parcel4[t-1] - dt*(q12[t-1] + q13[t-1] + q14[t-1] + q15[t-1]);
subject to STATE_PARCEL4_C5 {t in (arrival4 + 1)..(end4)}:
q11[t-1] + q16[t-1] + q17[t-1] + q18[t-1] = 0;
subject to FINAL_PARCEL_4: parcel4[end4] = final_parcelvol;
#####################################################################
# State Equations - composition
#####################################################################
# parcel 1: Crude 2 100% => T1, T6-T8
subject to INITIAL_COMP_T1 {t in 1..arrival1}:
key_T1[t] = ini_key_T1;
subject to INITIAL_COMP_T2 {t in 1..arrival1}:
key_T2[t] = ini_key_T2;
subject to INITIAL_COMP_T3 {t in 1..arrival1}:
key_T3[t] = ini_key_T3;
```

```
subject to INITIAL_COMP_T4 {t in 1..arrival1}:
key_T4[t] = ini_key_T4;
subject to INITIAL_COMP_T5 {t in 1..arrival1}:
key_T5[t] = ini_key_T5;
subject to INITIAL_COMP_T6 {t in 1..arrival1}:
key_T6[t] = ini_key_T6;
subject to INITIAL_COMP_T7 {t in 1..arrival1}:
key_T7[t] = ini_key_T7;
subject to INITIAL_COMP_T8 {t in 1..arrival1}:
key_T8[t] = ini_key_T8;
subject to COMP2_T1_1 {t in (arrival1 + 1)..end1}:
key_T1[t] = key_T1[t-1] + (key_parcel1 - key_T1[t-1])*q11[t-1]/vol1[t-1];
subject to COMP2_T2_1 {t in (arrival1 + 1)..end1}:
key_T2[t] = ini_key_T2;
subject to COMP2_T3_1 {t in (arrival1 + 1)..end1}:
key_T3[t] = ini_key_T3;
subject to COMP2_T4_1 {t in (arrival1 + 1)..end1}:
key_T4[t] = ini_key_T4;
subject to COMP2_T5_1 {t in (arrival1 + 1)..end1}:
key_T5[t] = ini_key_T5;
subject to COMP2_T6_1 {t in (arrival1 + 1)..end1}:
key_T6[t] = key_T6[t-1] + (key_parcel1 - key_T6[t-1])*q16[t-1]/vol6[t-1];
subject to COMP2_T7_1 {t in (arrival1 + 1)..end1}:
key_T7[t] = key_T7[t-1] + (key_parcel1 - key_T7[t-1])*q17[t-1]/vol7[t-1];
subject to COMP2_T8_1 {t in (arrival1 + 1)..end1}:
key_T8[t] = key_T8[t-1] + (key_parcel1 - key_T8[t-1])*q18[t-1]/vol8[t-1];
# parcel 2: Crude 3 100% => T1, T6-T8
subject to COMP2_T1_2 {t in (arrival2 + 1)..end2}:
key_T1[t] = key_T1[t-1] + (key_parcel2 - key_T1[t-1])*q11[t-1]/vol1[t-1];
subject to COMP2_T2_2 {t in (arrival2 + 1)..end2}:
key_T2[t] = ini_key_T2;
subject to COMP2_T3_2 {t in (arrival2 + 1)..end2}:
key_T3[t] = ini_key_T3;
subject to COMP2_T4_2 {t in (arrival2 + 1)..end2}:
```

```
key_T4[t] = ini_key_T4;
subject to COMP2_T5_2 {t in (arrival2 + 1)..end2}:
key_T5[t] = ini_key_T5;
subject to COMP2_T6_2 {t in (arrival2 + 1)..end2}:
key_T6[t] = key_T6[t-1] + (key_parcel2 - key_T6[t-1])*q16[t-1]/vol6[t-1];
subject to COMP2_T7_2 {t in (arrival2 + 1)..end2}:
key_T7[t] = key_T7[t-1] + (key_parcel2 - key_T7[t-1])*q17[t-1]/vol7[t-1];
subject to COMP2_T8_2 {t in (arrival2 + 1)..end2}:
key_T8[t] = key_T8[t-1] + (key_parcel2 - key_T8[t-1])*q18[t-1]/vol8[t-1];
# parcel 3: Crude 4 100% => T1, T6-T8
subject to COMP2_T1_3 {t in (arrival3 + 1)..end3}:
key_T1[t] = key_T1[t-1] + (key_parcel3 - key_T1[t-1])*q11[t-1]/vol1[t-1];
subject to COMP2_T2_3 {t in (arrival3 + 1)..end3}:
key_T2[t] = ini_key_T2;
subject to COMP2_T3_3 {t in (arrival3 + 1)..end3}:
key_T3[t] = ini_key_T3;
subject to COMP2_T4_3 {t in (arrival3 + 1)..end3}:
key_T4[t] = ini_key_T4;
subject to COMP2_T5_3 {t in (arrival3 + 1)..end3}:
key_T5[t] = ini_key_T5;
subject to COMP2_T6_3 {t in (arrival3 + 1)..end3}:
key_T6[t] = key_T6[t-1] + (key_parcel3 - key_T6[t-1])*q16[t-1]/vol6[t-1];
subject to COMP2_T7_3 {t in (arrival3 + 1)..end3}:
key_T7[t] = key_T7[t-1] + (key_parcel3 - key_T7[t-1])*q17[t-1]/vol7[t-1];
subject to COMP2_T8_3 {t in (arrival3 + 1)..end3}:
key_T8[t] = key_T8[t-1] + (key_parcel3 - key_T8[t-1])*q18[t-1]/vol8[t-1];
# parcel 4: Crude 5 100% => T2-T5
subject to COMP2_T1_4 {t in (arrival4 + 1)..end4}:
key_T1[t] = key_T1[end3];
subject to COMP2_T2_4 {t in (arrival4 + 1)..end4}:
key_T2[t] = key_T2[t-1] + (key_parcel4 - key_T2[t-1])*q12[t-1]/vol2[t-1];
subject to COMP2_T3_4 {t in (arrival4 + 1)..end4}:
key_T3[t] = key_T3[t-1] + (key_parcel4 - key_T3[t-1])*q13[t-1]/vol3[t-1];
subject to COMP2_T4_4 {t in (arrival4 + 1)..end4}:
```

```
key_T4[t] = key_T4[t-1] + (key_parcel4 - key_T4[t-1])*q14[t-1]/vol4[t-1];
subject to COMP2_T5_4 {t in (arrival4 + 1)..end4}:
key_T2[t] = key_T5[t-1] + (key_parcel4 - key_T5[t-1])*q15[t-1]/vol5[t-1];
subject to COMP2_T6_4 {t in (arrival4 + 1)..end4}:
key_T6[t] = key_T6[end3];
subject to COMP2_T7_4 {t in (arrival4 + 1)..end4}:
key_T7[t] = key_T7[end3];
subject to COMP2_T8_4 {t in (arrival4 + 1)..end4}:
key_T8[t] = key_T8[end3];
subject to 4_COMP2_T1 {t in (end4 + 1)..T}:
key_T1[t] = key_T1[end4];
subject to 4_COMP2_T2 {t in (end4 + 1)..T}:
key_T2[t] = key_T2[end4];
subject to 4_COMP2_T3 {t in (end4 + 1)..T}:
key_T3[t] = key_T3[end4];
subject to 4_COMP2_T4 {t in (end4 + 1)..T}:
key_T4[t] = key_T4[end4];
subject to 4_COMP2_T5 {t in (end4 + 1)..T}:
key_T5[t] = key_T5[end4];
subject to 4_COMP2_T6 {t in (end4 + 1)..T}:
key_T6[t] = key_T6[end4];
subject to 4_COMP2_T7 {t in (end4 + 1)..T}:
key_T7[t] = key_T7[end4];
subject to 4_COMP2_T8 {t in (end4 + 1)..T}:
key_T8[t] = key_T8[end4];
#########################################################
# Flow Constraints
#########################################################
#########################################################
# Pipeline flow
#########################################################
subject to PIPEFLOW {t in 0 .. T-1}: (q11[t] + q12[t] +
q13[t] + q14[t] + q15[t] + q16[t] + q17[t] +
q18[t]) <= q_max_pipe;
```

```
subject to ARRIVAL_PREVIOUS {t in 0..(arrival1 - 1)}:
q11[t] + q12[t] + q13[t] + q14[t] + q15[t] + q16[t] +
q17[t] + q18[t] = 0;
subject to ARRIVAL_END {t in (end4)..(T-1)}:
q11[t] + q12[t] + q13[t] + q14[t] + q15[t] + q16[t] +
q17[t] + q18[t] = 0;
#####################################################################
# Settling time implies in no running tank
# Constant Settling time = 8h
#####################################################################
subject to SETTLING_TIME_1 {t in 1..T-1}:
settling_1[t] = (q1A[t] + q1B[t] + q1C[t]) *
(sum{t1 in max(0, t - settling_time)..t} q11[t1]);
subject to SETTLING_TIME_2 {t in 1..T-1}:
settling_2[t] = (q2A[t] + q2B[t] + q2C[t]) *
(sum{t1 in max(0, t - settling_time)..t} q12[t1]);
subject to SETTLING_TIME_3 {t in 1..T-1}:
settling_3[t] = (q3A[t] + q3B[t] + q3C[t]) *
(sum{t1 in max(0, t - settling_time)..t} q13[t1]);
subject to SETTLING_TIME_4 {t in 1..T-1}:
settling_4[t] = (q4A[t] + q4B[t] + q4C[t]) *
(sum{t1 in max(0, t - settling_time)..t} q14[t1]);
subject to SETTLING_TIME_5 {t in 1..T-1}:
settling_5[t] = (q5A[t] + q5B[t] + q5C[t]) *
(sum{t1 in max(0, t - settling_time)..t} q15[t1]);
subject to SETTLING_TIME_6 {t in 1..T-1}:
settling_6[t] = (q6A[t] + q6B[t] + q6C[t]) *
(sum{t1 in max(0, t - settling_time)..t} q16[t1]);
subject to SETTLING_TIME_7 {t in 1..T-1}:
settling_7[t] = (q7A[t] + q7B[t] + q7C[t]) *
(sum{t1 in max(0, t - settling_time)..t} q17[t1]);
subject to SETTLING_TIME_8 {t in 1..T-1}:
settling_8[t] = (q8A[t] + q8B[t] + q8C[t]) *
(sum{t1 in max(0, t - settling_time)..t} q18[t1]);
```

```
subject to SETTLING_TIME_TOTAL {t in 1..T-1}:
settling_1[t] + settling_2[t] + settling_3[t] + settling_4[t] +
settling_5[t] + settling_6[t] + settling_7[t] + settling_8[t] = 0;
#########################################################################
# CDU flow
#########################################################################
subject to CDU_FLOW_A {t in 0..T-1}:
(q1A[t] + q2A[t]+ q3A[t]+ q4A[t]+ q5A[t]+
q6A[t] + q7A[t] + q8A[t]) <= feedflow;
subject to CDU_FLOW_B {t in 0..T-1}: (q1B[t] + q2B[t]+
q3B[t]+ q4B[t]+ q5B[t]+ q6B[t] + q7B[t] + q8B[t]) <= feedflow;
subject to CDU_FLOW_C {t in 0..T-1}: (q1C[t] + q2C[t]+
q3C[t]+ q4C[t]+ q5C[t]+ q6C[t] + q7C[t] + q8C[t]) <= feedflow;
subject to CDU_FLOW_MIN_A {t in 0..T-1}: (q1A[t] + q2A[t]+
q3A[t]+ q4A[t]+ q5A[t]+ q6A[t] + q7A[t] + q8A[t]) >= min_feedflow;
subject to CDU_FLOW_MIN_B {t in 0..T-1}: (q1B[t] + q2B[t]+
q3B[t]+ q4B[t]+ q5B[t]+ q6B[t] + q7B[t] + q8B[t]) >= min_feedflow;
subject to CDU_FLOW_MIN_C {t in 0..T-1}: (q1C[t] + q2C[t]+
q3C[t]+ q4C[t]+ q5C[t]+ q6C[t] + q7C[t] + q8C[t]) >= min_feedflow;
subject to CDU_VOL_A: dt*(sum {t in 0..T-1} (q1A[t] + q2A[t]+
q3A[t]+ q4A[t]+ q5A[t]+ q6A[t] + q7A[t] + q8A[t])) >= cdu_vol1;
subject to CDU_VOL_B: dt*(sum {t in 0..T-1} (q1B[t] + q2B[t]+
q3B[t]+ q4B[t]+ q5B[t]+ q6B[t] + q7B[t] + q8B[t])) >= cdu_vol1;
subject to CDU_VOL_C: dt*(sum {t in 0..T-1} (q1C[t] + q2C[t]+
q3C[t]+ q4C[t]+ q5C[t]+ q6C[t] + q7C[t] + q8C[t])) >= cdu_vol;
#########################################################################
# At most 2 tanks can feed CDU
#########################################################################
subject to AT_MOST_TWO_TANKS_1C {t in 0..T-1}:
twocduflow_1C[t] = q1C[t]* (q6C[t] + q7C[t] + q8C[t]);
subject to AT_MOST_TWO_TANKS_6C {t in 0..T-1}:
twocduflow_6C[t] = q6C[t]* (q7C[t] + q8C[t]);
subject to AT_MOST_TWO_TANKS_7C {t in 0..T-1}:
twocduflow_7C[t] = q7C[t]* q8C[t];
```

```
subject to AT_MOST_TWO_TANKS_TOTAL_C {t in 0..T-1}:
twocduflow_1C[t] = 0;
subject to AT_MOST_TWO_TANKS_2A {t in 0..T-1}:
twocduflow_2A[t] = q2A[t]* (q3A[t] + q4A[t] + q5A[t]);
subject to AT_MOST_TWO_TANKS_3A {t in 0..T-1}:
twocduflow_3A[t] = q3A[t]* (q4A[t] + q5A[t]);
subject to AT_MOST_TWO_TANKS_4A {t in 0..T-1}:
twocduflow_4A[t] = q4A[t]* q5A[t];
subject to AT_MOST_TWO_TANKS_TOTAL_A {t in 0..T-1}:
twocduflow_2A[t] = 0;
subject to AT_MOST_TWO_TANKS_2B {t in 0..T-1}:
twocduflow_2B[t] = q2B[t]* (q3B[t] + q4B[t] + q5B[t]);
subject to AT_MOST_TWO_TANKS_3B {t in 0..T-1}:
twocduflow_3B[t] = q3B[t]* (q4B[t] + q5B[t]);
subject to AT_MOST_TWO_TANKS_4B {t in 0..T-1}:
twocduflow_4B[t] = q4B[t]* q5B[t];
subject to AT_MOST_TWO_TANKS_TOTAL_B {t in 0..T-1}:
twocduflow_2B[t] = 0;
#######################################################
# FEED KEY COMPONENT LIMITS
#######################################################
var feed_key_A {0..T-1} <= feed_key_max_A * feedflow;
var feed_key_B {0..T-1} <= feed_key_max_B * feedflow;
var feed_key_C {0..T-1} <= feed_key_max_C * feedflow;
subject to CDU_KEY_A_T_MAX {t in 0..T-1}:
feed_key_A[t] <= (q1A[t]+q2A[t]+q3A[t]+q4A[t]+
q5A[t]+q6A[t]+q7A[t]+q8A[t]) * feed_key_max_A;
subject to CDU_KEY_B_T_MAX {t in 0..T-1}:
feed_key_B[t] <= (q1B[t]+q2B[t]+q3B[t]+q4B[t]+
q5B[t]+q6B[t]+q7B[t]+q8B[t]) * feed_key_max_B;
subject to CDU_KEY_C_T_MAX {t in 0..T-1}:
feed_key_C[t] <= (q1C[t]+q2C[t]+q3C[t]+q4C[t]+
q5C[t]+q6C[t]+q7C[t]+q8C[t]) * feed_key_max_C;
subject to MIN_FEED_CRUDE_A {t in 0..T-1}:
```

```
feed_key_A[t]  >= feed_key_min_A * min_feedflow;
subject to MIN_FEED_CRUDE_B {t in 0..T-1}:
feed_key_B[t]  >= feed_key_min_B * min_feedflow;
subject to MIN_FEED_CRUDE_C {t in 0..T-1}:
feed_key_C[t]  >= feed_key_min_C * min_feedflow;
subject to CDU_KEY_A_T_MIN {t in 0..T-1}:
feed_key_A[t] >= (q1A[t]+q2A[t]+q3A[t]+
q4A[t]+q5A[t]+q6A[t]+q7A[t]+q8A[t]) * feed_key_min_A;
subject to CDU_KEY_B_T_MIN {t in 0..T-1}:
feed_key_B[t] >= (q1B[t]+q2B[t]+q3B[t]+q4B[t]+
q5B[t]+q6B[t]+q7B[t]+q8B[t]) * feed_key_min_B;
subject to CDU_KEY_C_T_MIN {t in 0..T-1}:
feed_key_C[t] >= (q1C[t]+q2C[t]+q3C[t]+q4C[t]+
q5C[t]+q6C[t]+q7C[t]+q8C[t]) * feed_key_min_C;
# CDU A => T2-T5
subject to CDU_KEY_A_T0: feed_key_A[0] =
(q1A[0]*ini_key_T1 + q2A[0]*ini_key_T2 +
q3A[0]*ini_key_T3 + q4A[0]*ini_key_T4 +
q5A[0]*ini_key_T5 + q6A[0]*ini_key_T6 +
q7A[0]*ini_key_T7 + q8A[0]*ini_key_T8);
subject to CDU_KEY_A_T {t in 1..T-1}: feed_key_A[t] =
(q1A[t]*key_T1[t] + q2A[t]*key_T2[t] +
q3A[t]*key_T3[t] + q4A[t]*key_T4[t] +
q5A[t]*key_T5[t] + q6A[t]*key_T6[t] +
q7A[t]*key_T7[t] + q8A[t]*key_T8[t]);
# CDU B => T1, T6-T8
subject to CDU_KEY_B_T0: feed_key_B[0] =
(q1B[0]*ini_key_T1 + q2B[0]*ini_key_T2 +
q3B[0]*ini_key_T3 + q4B[0]*ini_key_T4 +
q5B[0]*ini_key_T5 + q6B[0]*ini_key_T6 +
q7B[0]*ini_key_T7 + q8B[0]*ini_key_T8);
subject to CDU_KEY_B_T {t in 1..T-1}: feed_key_B[t] =
(q1B[t]*key_T1[t] + q2B[t]*key_T2[t] +
q3B[t]*key_T3[t] + q4B[t]*key_T4[t] +
```

```
q5B[t]*key_T5[t] + q6B[t]*key_T6[t] +
q7B[t]*key_T7[t] + q8B[t]*key_T8[t]);
# CDU C => T1, T6-T8
subject to CDU_KEY_C_T0: feed_key_C[0] =
(q1C[0]*ini_key_T1 + q2C[0]*ini_key_T2 +
q3C[0]*ini_key_T3 + q4C[0]*ini_key_T4 +
q5C[0]*ini_key_T5 + q6C[0]*ini_key_T6 +
q7C[0]*ini_key_T7 + q8C[0]*ini_key_T8);
subject to CDU_KEY_C_T {t in 1..T-1}: feed_key_C[t] =
(q1C[t]*key_T1[t] + q2C[t]*key_T2[t] +
q3C[t]*key_T3[t] + q4C[t]*key_T4[t] +
q5C[t]*key_T5[t] + q6C[t]*key_T6[t] +
q7C[t]*key_T7[t] + q8C[t]*key_T8[t]);

#########################################################
# Pipeline -> one Tank
#########################################################
subject to ONE_PIPE_FLOW_1 {t in 0..T-1}:
onepipeflow_1[t] = q11[t]* (q12[t] + q13[t] + q14[t] +
q15[t] + q16[t] + q17[t] + q18[t]);
subject to ONE_PIPE_FLOW_2 {t in 0..T-1}:
onepipeflow_2[t] = q12[t]* (q13[t] + q14[t] + q15[t] +
q16[t] + q17[t] + q18[t]);
subject to ONE_PIPE_FLOW_3 {t in 0..T-1}:
onepipeflow_3[t] = q13[t]* (q14[t] + q15[t] + q16[t] +
q17[t] + q18[t]);
subject to ONE_PIPE_FLOW_4 {t in 0..T-1}:
onepipeflow_4[t] = q14[t]* (q15[t] + q16[t] +
q17[t] + q18[t]);
subject to ONE_PIPE_FLOW_5 {t in 0..T-1}:
onepipeflow_5[t] = q15[t]* (q16[t] + q17[t] + q18[t]);
subject to ONE_PIPE_FLOW_6 {t in 0..T-1}:
onepipeflow_6[t] = q16[t]* (q17[t] + q18[t]);
subject to ONE_PIPE_FLOW_7 {t in 0..T-1}:
```

```
onepipeflow_7[t] = q17[t]* (q18[t]);
subject to ONE_PIPE_FLOW_TOTAL {t in 0..T-1}:
onepipeflow_1[t] = 0;
#########################################################
# Objective Function - components
#########################################################
var switchover >= 0;
var twocdu >= 0;
var feed_diff >= 0;
var onepipe >= 0;
var settling >= 0;
var netback >= 0;
param max_feed_diff;
let max_feed_diff := cdu_vol;
subject to MAX_FEED_DIFF: feed_diff <= max_feed_diff;
param max_onepipe;
let max_onepipe := 1;
subject to MAX_ONEPIPE: onepipe <= max_onepipe;
param max_settling;
let max_settling := 1;
subject to MAX_SETTLING: settling <= max_settling;
param max_switchover;
let max_switchover := 1000;
subject to MAX_SWITCHOVER: switchover <= max_switchover;
param max_twocdu;
let max_twocdu := 100;
subject to MAX_TWOCDU: twocdu <= max_twocdu;
subject to SWITCHOVER_COST:
switchover = sum {t in 1..T-1}
(
(q11[t-1] - q11[t])^2
+ (q12[t-1] - q12[t])^2
+ (q13[t-1] - q13[t])^2
+ (q14[t-1] - q14[t])^2
```

```
+ (q15[t-1] - q15[t])^2
+ (q16[t-1] - q16[t])^2
+ (q1C[t-1] - q1C[t])^2
+ (q2A[t-1] - q2A[t])^2
+ (q3A[t-1] - q3A[t])^2
+ (q4A[t-1] - q4A[t])^2
+ (q5A[t-1] - q5A[t])^2
+ (q2B[t-1] - q2B[t])^2
+ (q3B[t-1] - q3B[t])^2
+ (q4B[t-1] - q4B[t])^2
+ (q5B[t-1] - q5B[t])^2
+ (q6C[t-1] - q6C[t])^2
+ (q7C[t-1] - q7C[t])^2
+ (q8C[t-1] - q8C[t])^2
)/feedflow^2;
subject to FEED_COST: feed_diff = 0;
subject to ONEPIPE_COST:
onepipe = sum{t in 0..T-1}(onepipeflow_1[t]+
onepipeflow_2[t]+onepipeflow_3[t]+onepipeflow_4[t]+
onepipeflow_5[t]+onepipeflow_6[t]+onepipeflow_7[t]);
subject to SETTLING_COST:
settling = sum {t in 1..T-1}(settling_1[t] +
settling_2[t] + settling_3[t] + settling_4[t] +
settling_5[t] + settling_6[t] + settling_7[t] +
settling_8[t]);
subject to TWOCDU_COST:
twocdu = sum {t in 0..T-1}
(twocduflow_1C[t] + twocduflow_2A[t] +
twocduflow_3A[t] + twocduflow_4A[t] +
twocduflow_2B[t] + twocduflow_3B[t] +
twocduflow_4B[t] + twocduflow_6C[t] +
twocduflow_7C[t])/feedflow^2;
param W_SW;
let W_SW := 1;
```

```
param W_FEED;
let W_FEED := 1;
param W_PIPE;
let W_PIPE := 1;
param W_CDU;
let W_CDU := 1;
param W_SET;
let W_SET := 1;
param W_PENALTY;
let W_PENALTY := 0;
param W_COST;
let W_COST := 0;
var cost;
subject to COST:
cost =  W_SW * switchover +  W_FEED * feed_diff;
var penalty;
subject to PENALTY: penalty =
W_PIPE * onepipe + W_CDU * twocdu + W_SET * settling;
minimize obj: W_COST * cost + W_PENALTY * penalty;
```

A.2.3 Caso de Teste 8

```
#####################################################
# Example 3 - Typical refinery
# Based on the example #3 from
# Pan et al. (2009)
# Chemical Engineering Science 64, 965-983
# Originally from Reddy et al. (2004)
# Chemical Engineering Science 59, 1325-1341
#####################################################
#####################################################
# Parameters
#####################################################
param dt;
let dt := 1.0;
```

```
# No of periods 160h
param T = 160.0/dt;
#####################################################################
# 8 tanks (T1-18), 3 CDU (CDU1-CDU3), 2 classes of Oil (cl1, cl2)
# Tanks T1, T6-T8 and CDU3 handle class1 crude oil (C1-C4),
# while the rest handle class2 crude oil (C5-C8)
# ONLY ONE KEY COMPONENT IS CONSIDERED TO CARACTERIZE THE CDU FEED:
# SULFUR OR METAL
#####################################################################
# Max Tank volume (m3)
param max_vol1 = 570.000;
param max_vol2 = 980.000;
# Min Tank volume (m3)
param min_vol1 = 60.000;
param min_vol2 = 110.000;
# Initial volumes
param ini_vol1 = 350.000; # ini vol of tank1
param ini_vol2 = 400.000; # ini vol of tank2
param ini_vol3 = 350.000; # ini vol of tank3
param ini_vol4 = 950.000; # ini vol of tank4
param ini_vol5 = 300.000; # ini vol of tank5
param ini_vol6 = 80.000; # ini vol of tank6
param ini_vol7 = 80.000; # ini vol of tank7
param ini_vol8 = 450.000; # ini vol of tank8
#################################################
# KEY COMPONENT (SULFUR OR METAL)
#################################################
# Initial composition
# (8 crudes: C1, C2, C3, C4, C5, C6, C7, C8,
# each with a different key component concentration
# and net back (profit))
param key_c1 = 0.20;
param key_c2 = 0.25;
param key_c3 = 0.15;
```

```
param key_c4 = 0.60;
param key_c5 = 1.20;
param key_c6 = 1.30;
param key_c7 = 0.90;
param key_c8 = 1.50;
param profit_c1 = 1.50;
param profit_c2 = 1.70;
param profit_c3 = 1.50;
param profit_c4 = 1.60;
param profit_c5 = 1.45;
param profit_c6 = 1.60;
param profit_c7 = 1.55;
param profit_c8 = 1.60;
param ini_cru11 = 50/350; # crude 1 - tank 1
param ini_cru21 = 100/350; # crude 2 - tank 1
param ini_cru31 = 100/350; # crude 3 - tank 1
param ini_cru41 = 100/350; # crude 4 - tank 1
param ini_key_T1 = ini_cru11 * key_c1 +
ini_cru21 * key_c2 + ini_cru31 * key_c3 +
ini_cru41 * key_c4;
param ini_profit_T1 = ini_cru11 * profit_c1 +
ini_cru21 * profit_c2 + ini_cru31 * profit_c3 +
ini_cru41 * profit_c4;
param ini_cru52 = 0.25; # crude 5 - tank 2
param ini_cru62 = 0.25; # crude 6 - tank 2
param ini_cru72 = 0.25; # crude 7 - tank 2
param ini_cru82 = 0.25; # crude 8 - tank 2
param ini_key_T2 = ini_cru52 * key_c5 +
ini_cru62 * key_c6 + ini_cru72 * key_c7 +
ini_cru82 * key_c8;
param ini_profit_T2 = ini_cru52 * profit_c5 +
ini_cru62 * profit_c6 + ini_cru72 * profit_c7 +
ini_cru82 * profit_c8;
param ini_cru53 = 100/350; # crude 5 - tank 3
```

```
param ini_cru63 = 100/350; # crude 6 - tank 3
param ini_cru73 = 50/350; # crude 7 - tank 3
param ini_cru83 = 100/350; # crude 8 - tank 3
param ini_key_T3 = ini_cru53 * key_c5 +
ini_cru63 * key_c6 + ini_cru73 * key_c7 +
ini_cru83 * key_c8;
param ini_profit_T3 = ini_cru53 * profit_c5 +
ini_cru63 * profit_c6 + ini_cru73 * profit_c7 +
ini_cru83 * profit_c8;
param ini_cru54 = 200/950; # crude 5 - tank 4
param ini_cru64 = 250/950; # crude 6 - tank 4
param ini_cru74 = 200/950; # crude 7 - tank 4
param ini_cru84 = 300/950; # crude 8 - tank 4
param ini_key_T4 = ini_cru54 * key_c5 +
ini_cru64 * key_c6 + ini_cru74 * key_c7 +
ini_cru84 * key_c8;
param ini_profit_T4 = ini_cru54 * profit_c5 +
ini_cru64 * profit_c6 + ini_cru74 * profit_c7 +
ini_cru84 * profit_c8;
param ini_cru55 = 100/300; # crude 5 - tank 5
param ini_cru65 = 100/300; # crude 6 - tank 5
param ini_cru75 = 50/300; # crude 7 - tank 5
param ini_cru85 = 50/300; # crude 8 - tank 5
param ini_key_T5 = ini_cru55 * key_c5 +
ini_cru65 * key_c6 + ini_cru75 * key_c7 +
ini_cru85 * key_c8;
param ini_profit_T5 = ini_cru55 * profit_c5 +
ini_cru65 * profit_c6 + ini_cru75 * profit_c7 +
ini_cru85 * profit_c8;
param ini_cru16 = 0.25; # crude 1 - tank 6
param ini_cru26 = 0.25; # crude 2 - tank 6
param ini_cru36 = 0.25; # crude 3 - tank 6
param ini_cru46 = 0.25; # crude 4 - tank 6
param ini_key_T6 = ini_cru16 * key_c1 +
```

```
ini_cru26 * key_c2 + ini_cru36 * key_c3 +
ini_cru46 * key_c4;
param ini_profit_T6 = ini_cru16 * profit_c1 +
ini_cru26 * profit_c2 + ini_cru36 * profit_c3 +
ini_cru46 * profit_c4;
param ini_cru17 = 0.25; # crude 1 - tank 7
param ini_cru27 = 0.25; # crude 2 - tank 7
param ini_cru37 = 0.25; # crude 3 - tank 7
param ini_cru47 = 0.25; # crude 4 - tank 7
param ini_key_T7 = ini_cru17 * key_c1 +
ini_cru27 * key_c2 + ini_cru37 * key_c3 +
ini_cru47 * key_c4;
param ini_profit_T7 = ini_cru17 * profit_c1 +
ini_cru27 * profit_c2 + ini_cru37 * profit_c3 +
ini_cru47 * profit_c4;
param ini_cru18 = 100/450; # crude 1 - tank 8
param ini_cru28 = 100/450; # crude 2 - tank 8
param ini_cru38 = 100/450; # crude 3 - tank 8
param ini_cru48 = 150/450; # crude 4 - tank 8
param ini_key_T8 = ini_cru18 * key_c1 +
ini_cru28 * key_c2 + ini_cru38 * key_c3 +
ini_cru48 * key_c4;
param ini_profit_T8 = ini_cru18 * profit_c1 +
ini_cru28 * profit_c2 + ini_cru38 * profit_c3 +
ini_cru48 * profit_c4;
# max pipeline flow 400 kbbl/8h = 50 kbbl/h
param q_max_pipe = 50.0;
# Ship final volume
param final_parcelvol = 0.0;
################################################
# VLCC1 + VLCC2
################################################
# parcel volumes
param ini_parcel1 = 10.000;   # 10 kbbl C2
```

```
param ini_parcel2 = 250.000;  # 250 kbbl C3
param ini_parcel3 = 300.000;  # 300 kbbl C4
param ini_parcel4 = 190.000;  # 190 kbbl C5
param ini_parcel5 = 10.000; # 10 kbbl C5
param ini_parcel6 = 250.000; # 250 kbbl C6
param ini_parcel7 = 250.000; # 250 kbbl C8
param ini_parcel8 = 240.000; # 240 kbbl C3
#arrival crude parcels
param arrival1 = ceil(15/dt);   # t = 15h
param arrival2 = ceil(16/dt);  # t = 16h
param arrival3 = ceil(21/dt);  # t = 21h
param arrival4 = ceil(27/dt);  # t = 27h
param arrival5 = ceil(100/dt);   # t = 100h
param arrival6 = ceil(101/dt);  # t = 101h
param arrival7 = ceil(106/dt);  # t = 106h
param arrival8 = ceil(111/dt);  # t = 111h
#end crude parcels
# t = 15.2h -> 16h
param end1 = max(ceil(15.2/dt), arrival1 + 1);
# t = 16h + 5h = 21h
param end2 = max(ceil(21/dt), arrival2 + 1);
# t = 21h + 6h = 27h
param end3 = max(ceil(27/dt), arrival3 + 1);
# t = 27h + 4h = 31h
param end4 = max(ceil(31/dt), arrival4 + 1);
# t = 100.2h -> 101h
param end5 = max(ceil(100.2/dt), arrival5 + 1);
# t = 101h + 5h = 106h
param end6 = max(ceil(106/dt), arrival6 + 1);
# t = 106h + 5h = 111h
param end7 = max(ceil(111/dt), arrival7 + 1);
# t = 111h + 4.8h = 115.8h -> 116
param end8 = max(ceil(115.8/dt), arrival8 + 1);
#crude parcels
```

```
param key_parcel1 = key_c2;   # 100% C2
param key_parcel2 = key_c3;  # 100% C3
param key_parcel3 = key_c4;  # 100% C4
param key_parcel4 = key_c5;  # 100% C5
param key_parcel5 = key_c5;   # 100% C5
param key_parcel6 = key_c6;  # 100% C6
param key_parcel7 = key_c8;  # 100% C8
param key_parcel8 = key_c3;  # 100% C3
param profit_parcel1 = profit_c2;   # 100% C2
param profit_parcel2 = profit_c3;  # 100% C3
param profit_parcel3 = profit_c4;  # 100% C4
param profit_parcel4 = profit_c5;  # 100% C5
param profit_parcel5 = profit_c5;   # 100% C5
param profit_parcel6 = profit_c6;  # 100% C6
param profit_parcel7 = profit_c8;  # 100% C8
param profit_parcel8 = profit_c3;  # 100% C3
# resting/settling time
 # 8h to settle brine
param settling_time = ceil(8/dt);
#CDU feed flow 48 kbbl/8h = 6 kbbl/h
param feedflow = 6.0;
param cdu_vol = 700; # feedflow*T*dt
param min_feedflow = 2.0;
# CDU feed key component limits
param feed_key_max_A = 1.30;
param feed_key_max_B = 1.25;
param feed_key_max_C = 0.35;
param feed_key_min_A = 0.10;
param feed_key_min_B = 0.10;
param feed_key_min_C = 0.10;
#min tank flow
param min_q = 0;
# penalty variables
var twocduflow_1C {0..T-1} ;
```

```
var twocduflow_2A {0..T-1} ;
var twocduflow_3A {0..T-1} ;
var twocduflow_4A {0..T-1} ;
var twocduflow_2B {0..T-1} ;
var twocduflow_3B {0..T-1} ;
var twocduflow_4B {0..T-1} ;
var twocduflow_6C {0..T-1} ;
var twocduflow_7C {0..T-1} ;
##################################################
# Control variables
##################################################
# transferred volume from pipeline VLCC to T1
var q11 {0..T-1} <= q_max_pipe, := 0;
var q12 {0..T-1} <= q_max_pipe, := 0;
var q13 {0..T-1} <= q_max_pipe, := 0;
var q14 {0..T-1} <= q_max_pipe, := 0;
var q15 {0..T-1} <= q_max_pipe, := 0;
var q16 {0..T-1} <= q_max_pipe, := 0;
var q17 {0..T-1} <= q_max_pipe, := 0;
var q18 {0..T-1} <= q_max_pipe, := 0;
subject to MIN_Q11 {t in 0..T-1}: q11[t]  >= min_q;
subject to MIN_Q12 {t in 0..T-1}: q12[t]  >= min_q;
subject to MIN_Q13 {t in 0..T-1}: q13[t]  >= min_q;
subject to MIN_Q14 {t in 0..T-1}: q14[t]  >= min_q;
subject to MIN_Q15 {t in 0..T-1}: q15[t]  >= min_q;
subject to MIN_Q16 {t in 0..T-1}: q16[t]  >= min_q;
subject to MIN_Q17 {t in 0..T-1}: q17[t]  >= min_q;
subject to MIN_Q18 {t in 0..T-1}: q18[t]  >= min_q;
# transferred volume from T2 to CDU1
var q2A {0..T-1} <= feedflow, := 0;
# transferred volume from T3 to CDU1
var q3A {0..T-1} <= feedflow, := 0;
# transferred volume from T4 to CDU1
var q4A {0..T-1} <= feedflow, := 0;
```

```
# transferred volume from T5 to CDU1
var q5A {0..T-1} <= feedflow, := 0;
#subject to MIN_Q1A {t in 0..T-1}: q1A[t]  = min_q;
subject to MIN_Q2A {t in 0..T-1}: q2A[t]  >= min_q;
subject to MIN_Q3A {t in 0..T-1}: q3A[t]  >= min_q;
subject to MIN_Q4A {t in 0..T-1}: q4A[t]  >= min_q;
subject to MIN_Q5A {t in 0..T-1}: q5A[t]  >= min_q;
# transferred volume from T1 to CDU2
#var q1B {0..T-1} <= feedflow, := 0;
 # transferred volume from T2 to CDU2
var q2B {0..T-1} <= feedflow, := 0;
# transferred volume from T3 to CDU2
var q3B {0..T-1} <= feedflow, := 0;
# transferred volume from T4 to CDU2
var q4B {0..T-1} <= feedflow, := 0;
# transferred volume from T5 to CDU2
var q5B {0..T-1} <= feedflow, := 0;
#subject to MIN_Q1B {t in 0..T-1}:q1B[t]  = min_q;
subject to MIN_Q2B {t in 0..T-1}:q2B[t]  >= min_q;
subject to MIN_Q3B {t in 0..T-1}:q3B[t]  >= min_q;
subject to MIN_Q4B {t in 0..T-1}:q4B[t]  >= min_q;
subject to MIN_Q5B {t in 0..T-1}:q5B[t]  >= min_q;
 # transferred volume from T1 to CDU3
var q1C {0..T-1} <= feedflow, := 0;
# transferred volume from T6 to CDU3
var q6C {0..T-1} <= feedflow, := 0;
# transferred volume from T7 to CDU3
var q7C {0..T-1} <= feedflow, := 0;
# transferred volume from T8 to CDU3
var q8C {0..T-1} <= feedflow, := 0;
subject to MIN_Q1C {t in 0..T-1}: q1C[t]  >= min_q;
subject to MIN_Q6C {t in 0..T-1}: q6C[t]  >= min_q;
subject to MIN_Q7C {t in 0..T-1}: q7C[t]  >= min_q;
subject to MIN_Q8C {t in 0..T-1}: q8C[t]  >= min_q;
```

```
#####################################################
# State Variables
#####################################################
var vol1 {1..T} <= max_vol1, := ini_vol1;
var vol2 {1..T} <= max_vol1, := ini_vol2;
var vol3 {1..T} <= max_vol1, := ini_vol3;
var vol4 {1..T} <= max_vol2, := ini_vol4;
var vol5 {1..T} <= max_vol2, := ini_vol5;
var vol6 {1..T} <= max_vol1, := ini_vol6;
var vol7 {1..T} <= max_vol1, := ini_vol7;
var vol8 {1..T} <= max_vol1, := ini_vol8;
subject to MIN_VOL1 {t in 1..T}: vol1[t] >= min_vol1;
subject to MIN_VOL2 {t in 1..T}: vol2[t] >= min_vol1;
subject to MIN_VOL3 {t in 1..T}: vol3[t] >= min_vol1;
subject to MIN_VOL4 {t in 1..T}: vol4[t] >= min_vol2;
subject to MIN_VOL5 {t in 1..T}: vol5[t] >= min_vol2;
subject to MIN_VOL6 {t in 1..T}: vol6[t] >= min_vol1;
subject to MIN_VOL7 {t in 1..T}: vol7[t] >= min_vol1;
subject to MIN_VOL8 {t in 1..T}: vol8[t] >= min_vol1;
# volumes P1 in period t
var parcel1 {arrival1..end1} <= ini_parcel1, := ini_parcel1;
var parcel2 {arrival2..end2} <= ini_parcel2, := ini_parcel2;
var parcel3 {arrival3..end3} <= ini_parcel3, := ini_parcel3;
var parcel4 {arrival4..end4} <= ini_parcel4, := ini_parcel4;
var parcel5 {arrival5..end5} <= ini_parcel5, := ini_parcel5;
var parcel6 {arrival6..end6} <= ini_parcel6, := ini_parcel6;
var parcel7 {arrival7..end7} <= ini_parcel7, := ini_parcel7;
var parcel8 {arrival8..end8} <= ini_parcel8, := ini_parcel8;
subject to MIN_PARCEL21 {t in arrival1..end1}:
parcel1[t]  >= final_parcelvol;
subject to MIN_PARCEL22 {t in arrival2..end2}:
parcel2[t]  >= final_parcelvol;
subject to MIN_PARCEL23 {t in arrival3..end3}:
parcel3[t]  >= final_parcelvol;
```

```
subject to MIN_PARCEL24 {t in arrival4..end4}:
parcel4[t]  >= final_parcelvol;
subject to MIN_PARCEL25 {t in arrival5..end5}:
parcel5[t]  >= final_parcelvol;
subject to MIN_PARCEL26 {t in arrival6..end6}:
parcel6[t]  >= final_parcelvol;
subject to MIN_PARCEL27 {t in arrival7..end7}:
parcel7[t]  >= final_parcelvol;
subject to MIN_PARCEL28 {t in arrival8..end8}:
parcel8[t]  >= final_parcelvol;
#############################################################
# Tank Composition
#############################################################
var key_T1 {1..T} := ini_key_T1;        # key in T1
var key_T2 {1..T} := ini_key_T2;        # key in T2
var key_T3 {1..T} := ini_key_T3;        # key in T3
var key_T4 {1..T} := ini_key_T4;        # key in T4
var key_T5 {1..T} := ini_key_T5;        # key in T5
var key_T6 {1..T} := ini_key_T6;        # key in T6
var key_T7 {1..T} := ini_key_T7;        # key in T7
var key_T8 {1..T} := ini_key_T8;        # key in T8
#################################################
# State Equations - volumes
#################################################
subject to INITIAL_STATE1:
vol1[1] = ini_vol1 + dt*(q11[0] - q1C[0]);
subject to STATE_EQ1 {t in 2..T}:
vol1[t] = vol1[t-1] + dt*(q11[t-1] - q1C[t-1]);
subject to INITIAL_STATE2:
vol2[1] = ini_vol2 + dt*(q12[0] - q2A[0] - q2B[0]);
subject to STATE_EQ2 {t in 2..T}:
vol2[t] = vol2[t-1] + dt*(q12[t-1] - q2A[t-1] - q2B[t-1]);
subject to INITIAL_STATE3:
vol3[1] = ini_vol3 + dt*(q13[0] - q3A[0] - q3B[0]);
```

```
subject to STATE_EQ3 {t in 2..T}:
vol3[t] = vol3[t-1] + dt*(q13[t-1] - q3A[t-1] - q3B[t-1]);
subject to INITIAL_STATE4:
vol4[1] = ini_vol4 + dt*(q14[0] - q4A[0] - q4B[0]);
subject to STATE_EQ4 {t in 2..T}:
vol4[t] = vol4[t-1] + dt*(q14[t-1] - q4A[t-1] - q4B[t-1]);
subject to INITIAL_STATE5:
vol5[1] = ini_vol5 + dt*(q15[0] - q5A[0] - q5B[0]);
subject to STATE_EQ5 {t in 2..T}:
vol5[t] = vol5[t-1] + dt*(q15[t-1] - q5A[t-1] - q5B[t-1]);
subject to INITIAL_STATE6:
vol6[1] = ini_vol6 + dt*(q16[0] - q6C[0]);
subject to STATE_EQ6 {t in 2..T}:
vol6[t] = vol6[t-1] + dt*(q16[t-1] - q6C[t-1]);
subject to INITIAL_STATE7:
vol7[1] = ini_vol7 + dt*(q17[0] - q7C[0]);
subject to STATE_EQ7 {t in 2..T}:
vol7[t] = vol7[t-1] + dt*(q17[t-1] - q7C[t-1]);
subject to INITIAL_STATE8:
vol8[1] = ini_vol8 + dt*(q18[0] - q8C[0]);
subject to STATE_EQ8 {t in 2..T}:
vol8[t] = vol8[t-1] + dt*(q18[t-1] - q8C[t-1]);
############################################################
# State Equations - Crude parcel volumes
############################################################
#C2 only in T1, T6-T8
subject to INITIAL_PARCEL1: parcel1[arrival1] = ini_parcel1;
subject to STATE_PARCEL1 {t in (arrival1 + 1)..(end1)}:
parcel1[t] = parcel1[t-1] - dt*(q11[t-1] + q16[t-1] +
q17[t-1] + q18[t-1]);
subject to STATE_PARCEL1_C2 {t in (arrival1 + 1)..(end1)}:
q12[t-1] + q13[t-1] + q14[t-1] + q15[t-1] = 0;
subject to FINAL_PARCEL_1: parcel1[end1] = final_parcelvol;
#C3 only in T1, T6-T8
```

```
subject to INITIAL_PARCEL2: parcel2[arrival2] = ini_parcel2;
subject to STATE_PARCEL2 {t in (arrival2 + 1)..(end2)}:
parcel2[t] = parcel2[t-1] - dt*(q11[t-1] + q16[t-1] +
q17[t-1] + q18[t-1]);
subject to STATE_PARCEL2_C3 {t in (arrival2 + 1)..(end2)}:
q12[t-1] + q13[t-1] + q14[t-1] + q15[t-1] = 0;
subject to FINAL_PARCEL_2: parcel2[end2] = final_parcelvol;
#C4 only in T1, T6-T8
subject to INITIAL_PARCEL3: parcel3[arrival3] = ini_parcel3;
subject to STATE_PARCEL3 {t in (arrival3 + 1)..(end3)}:
parcel3[t] = parcel3[t-1] - dt*(q11[t-1] + q16[t-1] +
q17[t-1] + q18[t-1]);
subject to STATE_PARCEL3_C4 {t in (arrival3 + 1)..(end3)}:
q12[t-1] + q13[t-1] + q14[t-1] + q15[t-1] = 0;
subject to FINAL_PARCEL_3: parcel3[end3] = final_parcelvol;
#C5 only in T2-T5
subject to INITIAL_PARCEL4: parcel4[arrival4] = ini_parcel4;
subject to STATE_PARCEL4 {t in (arrival4 + 1)..(end4)}:
parcel4[t] = parcel4[t-1] - dt*(q12[t-1] + q13[t-1] +
q14[t-1] + q15[t-1]);
subject to STATE_PARCEL4_C5 {t in (arrival4 + 1)..(end4)}:
q11[t-1] + q16[t-1] + q17[t-1] + q18[t-1] = 0;
subject to FINAL_PARCEL_4: parcel4[end4] = final_parcelvol;
#C5 only in T2-T5
subject to INITIAL_PARCEL5: parcel5[arrival5] = ini_parcel5;
subject to STATE_PARCEL5 {t in (arrival5 + 1)..(end5)}:
parcel5[t] = parcel5[t-1] - dt*(q12[t-1] + q13[t-1] +
q14[t-1] + q15[t-1]);
subject to STATE_PARCEL5_C5 {t in (arrival5 + 1)..(end5)}:
q11[t-1] + q16[t-1] + q17[t-1] + q18[t-1] = 0;
subject to FINAL_PARCEL_5: parcel5[end5] = final_parcelvol;
#C6 only in T2-T5
subject to INITIAL_PARCEL6: parcel6[arrival6] = ini_parcel6;
subject to STATE_PARCEL6 {t in (arrival6 + 1)..(end6)}:
```

```
parcel6[t] = parcel6[t-1] - dt*(q12[t-1] + q13[t-1] +
q14[t-1] + q15[t-1]);
subject to STATE_PARCEL6_C6 {t in (arrival6 + 1)..(end6)}:
q11[t-1] + q16[t-1] + q17[t-1] + q18[t-1] = 0;
subject to FINAL_PARCEL_6: parcel6[end6] = final_parcelvol;

#C8 only in T2-T5
subject to INITIAL_PARCEL7: parcel7[arrival7] = ini_parcel7;
subject to STATE_PARCEL7 {t in (arrival7 + 1)..(end7)}:
parcel7[t] = parcel7[t-1] - dt*(q12[t-1] + q13[t-1] + q14[t-1] + q15[t-1]);
subject to STATE_PARCEL7_C8 {t in (arrival7 + 1)..(end7)}:
q11[t-1] + q16[t-1] + q17[t-1] + q18[t-1] = 0;
subject to FINAL_PARCEL_7: parcel7[end7] = final_parcelvol;
#C3 only in T1, T6-T8
subject to INITIAL_PARCEL8: parcel8[arrival8] = ini_parcel8;
subject to STATE_PARCEL8 {t in (arrival8 + 1)..(end8)}:
parcel8[t] = parcel8[t-1] - dt*(q11[t-1] + q16[t-1] + q17[t-1] + q18[t-1]);
subject to STATE_PARCEL8_C3 {t in (arrival8 + 1)..(end8)}:
q12[t-1] + q13[t-1] + q14[t-1] + q15[t-1] = 0;
subject to FINAL_PARCEL_8: parcel8[end8] = final_parcelvol;
#############################################################################
# State Equations - composition
# Crude X at Tank Y = initial crude x + crude x from pipeline - crude x to CDU
#############################################################################
# parcel 1: Crude 2 100% => T1, T6-T8
subject to INITIAL_COMP_T1 {t in 1..arrival1}:
key_T1[t] = ini_key_T1;
subject to INITIAL_COMP_T2 {t in 1..arrival1}:
key_T2[t] = ini_key_T2;
subject to INITIAL_COMP_T3 {t in 1..arrival1}:
key_T3[t] = ini_key_T3;
subject to INITIAL_COMP_T4 {t in 1..arrival1}:
key_T4[t] = ini_key_T4;
subject to INITIAL_COMP_T5 {t in 1..arrival1}:
```

```
key_T5[t] = ini_key_T5;
subject to INITIAL_COMP_T6 {t in 1..arrival1}:
key_T6[t] = ini_key_T6;
subject to INITIAL_COMP_T7 {t in 1..arrival1}:
key_T7[t] = ini_key_T7;
subject to INITIAL_COMP_T8 {t in 1..arrival1}:
key_T8[t] = ini_key_T8;
subject to COMP2_T1_1 {t in (arrival1 + 1)..end1}:
key_T1[t] = key_T1[t-1] + (key_parcel1 - key_T1[t-1])*q11[t-1]/vol1[t-1];
subject to COMP2_T2_1 {t in (arrival1 + 1)..end1}:
key_T2[t] = ini_key_T2;
subject to COMP2_T3_1 {t in (arrival1 + 1)..end1}:
key_T3[t] = ini_key_T3;
subject to COMP2_T4_1 {t in (arrival1 + 1)..end1}:
key_T4[t] = ini_key_T4;
subject to COMP2_T5_1 {t in (arrival1 + 1)..end1}:
key_T5[t] = ini_key_T5;
subject to COMP2_T6_1 {t in (arrival1 + 1)..end1}:
key_T6[t] = key_T6[t-1] + (key_parcel1 - key_T6[t-1])*q16[t-1]/vol6[t-1];
subject to COMP2_T7_1 {t in (arrival1 + 1)..end1}:
key_T7[t] = key_T7[t-1] + (key_parcel1 - key_T7[t-1])*q17[t-1]/vol7[t-1];
subject to COMP2_T8_1 {t in (arrival1 + 1)..end1}:
key_T8[t] = key_T8[t-1] + (key_parcel1 - key_T8[t-1])*q18[t-1]/vol8[t-1];
# parcel 2: Crude 3 100% => T1, T6-T8
subject to COMP2_T1_2 {t in (arrival2 + 1)..end2}:
key_T1[t] = key_T1[t-1] + (key_parcel2 - key_T1[t-1])*q11[t-1]/vol1[t-1];
subject to COMP2_T2_2 {t in (arrival2 + 1)..end2}:
key_T2[t] = ini_key_T2;
subject to COMP2_T3_2 {t in (arrival2 + 1)..end2}:
key_T3[t] = ini_key_T3;
subject to COMP2_T4_2 {t in (arrival2 + 1)..end2}:
key_T4[t] = ini_key_T4;
subject to COMP2_T5_2 {t in (arrival2 + 1)..end2}:
key_T5[t] = ini_key_T5;
```

```
subject to COMP2_T6_2 {t in (arrival2 + 1)..end2}:
key_T6[t] = key_T6[t-1] + (key_parcel2 - key_T6[t-1])*q16[t-1]/vol6[t-1];
subject to COMP2_T7_2 {t in (arrival2 + 1)..end2}:
key_T7[t] = key_T7[t-1] + (key_parcel2 - key_T7[t-1])*q17[t-1]/vol7[t-1];
subject to COMP2_T8_2 {t in (arrival2 + 1)..end2}:
key_T8[t] = key_T8[t-1] + (key_parcel2 - key_T8[t-1])*q18[t-1]/vol8[t-1];
# parcel 3: Crude 4 100% => T1, T6-T8
subject to COMP2_T1_3 {t in (arrival3 + 1)..end3}:
key_T1[t] = key_T1[t-1] + (key_parcel3 - key_T1[t-1])*q11[t-1]/vol1[t-1];
subject to COMP2_T2_3 {t in (arrival3 + 1)..end3}:
key_T2[t] = ini_key_T2;
subject to COMP2_T3_3 {t in (arrival3 + 1)..end3}:
key_T3[t] = ini_key_T3;
subject to COMP2_T4_3 {t in (arrival3 + 1)..end3}:
key_T4[t] = ini_key_T4;
subject to COMP2_T5_3 {t in (arrival3 + 1)..end3}:
key_T5[t] = ini_key_T5;
subject to COMP2_T6_3 {t in (arrival3 + 1)..end3}:
key_T6[t] = key_T6[t-1] + (key_parcel3 - key_T6[t-1])*q16[t-1]/vol6[t-1];
subject to COMP2_T7_3 {t in (arrival3 + 1)..end3}:
key_T7[t] = key_T7[t-1] + (key_parcel3 - key_T7[t-1])*q17[t-1]/vol7[t-1];
subject to COMP2_T8_3 {t in (arrival3 + 1)..end3}:
key_T8[t] = key_T8[t-1] + (key_parcel3 - key_T8[t-1])*q18[t-1]/vol8[t-1];
# parcel 4: Crude 5 100% => T2-T5
subject to COMP2_T1_4 {t in (arrival4 + 1)..end4}:
key_T1[t] = key_T1[end3];
subject to COMP2_T2_4 {t in (arrival4 + 1)..end4}:
key_T2[t] = key_T2[t-1] + (key_parcel4 - key_T2[t-1])*q12[t-1]/vol2[t-1];
subject to COMP2_T3_4 {t in (arrival4 + 1)..end4}:
key_T3[t] = key_T3[t-1] + (key_parcel4 - key_T3[t-1])*q13[t-1]/vol3[t-1];
subject to COMP2_T4_4 {t in (arrival4 + 1)..end4}:
key_T4[t] = key_T4[t-1] + (key_parcel4 - key_T4[t-1])*q14[t-1]/vol4[t-1];
subject to COMP2_T5_4 {t in (arrival4 + 1)..end4}:
key_T2[t] = key_T5[t-1] + (key_parcel4 - key_T5[t-1])*q15[t-1]/vol5[t-1];
```

```
subject to COMP2_T6_4 {t in (arrival4 + 1)..end4}:
key_T6[t] = key_T6[end3];
subject to COMP2_T7_4 {t in (arrival4 + 1)..end4}:
key_T7[t] = key_T7[end3];
subject to COMP2_T8_4 {t in (arrival4 + 1)..end4}:
key_T8[t] = key_T8[end3];
# Between parcels 4 and 5
subject to 4_COMP2_T1 {t in (end4 + 1)..arrival5}:
key_T1[t] = key_T1[end4];
subject to 4_COMP2_T2 {t in (end4 + 1)..arrival5}:
key_T2[t] = key_T2[end4];
subject to 4_COMP2_T3 {t in (end4 + 1)..arrival5}:
key_T3[t] = key_T3[end4];
subject to 4_COMP2_T4 {t in (end4 + 1)..arrival5}:
key_T4[t] = key_T4[end4];
subject to 4_COMP2_T5 {t in (end4 + 1)..arrival5}:
key_T5[t] = key_T5[end4];
subject to 4_COMP2_T6 {t in (end4 + 1)..arrival5}:
key_T6[t] = key_T6[end4];
subject to 4_COMP2_T7 {t in (end4 + 1)..arrival5}:
key_T7[t] = key_T7[end4];
subject to 4_COMP2_T8 {t in (end4 + 1)..arrival5}:
key_T8[t] = key_T8[end4];
# parcel 5: Crude 5 100% => T2-T5
subject to COMP2_T1_5 {t in (arrival5 + 1)..end5}:
key_T1[t] = key_T1[end4];
subject to COMP2_T2_5 {t in (arrival5 + 1)..end5}:
key_T2[t] = key_T2[t-1] + (key_parcel5 - key_T2[t-1])*q12[t-1]/vol2[t-1];
subject to COMP2_T3_5 {t in (arrival5 + 1)..end5}:
key_T3[t] = key_T3[t-1] + (key_parcel5 - key_T3[t-1])*q13[t-1]/vol3[t-1];
subject to COMP2_T4_5 {t in (arrival5 + 1)..end5}:
key_T4[t] = key_T4[t-1] + (key_parcel5 - key_T4[t-1])*q14[t-1]/vol4[t-1];
subject to COMP2_T5_5 {t in (arrival5 + 1)..end5}:
key_T2[t] = key_T5[t-1] + (key_parcel5 - key_T5[t-1])*q15[t-1]/vol5[t-1];
```

```
subject to COMP2_T6_5 {t in (arrival5 + 1)..end5}:
key_T6[t] = key_T6[end4];
subject to COMP2_T7_5 {t in (arrival5 + 1)..end5}:
key_T7[t] = key_T7[end4];
subject to COMP2_T8_5 {t in (arrival5 + 1)..end5}:
key_T8[t] = key_T8[end4];
# parcel 6: Crude 6 100% => T2-T5
subject to COMP2_T1_6 {t in (arrival6 + 1)..end6}:
key_T1[t] = key_T1[end5];
subject to COMP2_T2_6 {t in (arrival6 + 1)..end6}:
key_T2[t] = key_T2[t-1] + (key_parcel6 - key_T2[t-1])*q12[t-1]/vol2[t-1];
subject to COMP2_T3_6 {t in (arrival6 + 1)..end6}:
key_T3[t] = key_T3[t-1] + (key_parcel6 - key_T3[t-1])*q13[t-1]/vol3[t-1];
subject to COMP2_T4_6 {t in (arrival6 + 1)..end6}:
key_T4[t] = key_T4[t-1] + (key_parcel6 - key_T4[t-1])*q14[t-1]/vol4[t-1];
subject to COMP2_T5_6 {t in (arrival6 + 1)..end6}:
key_T2[t] = key_T5[t-1] + (key_parcel6 - key_T5[t-1])*q15[t-1]/vol5[t-1];
subject to COMP2_T6_6 {t in (arrival6 + 1)..end6}:
key_T6[t] = key_T6[end5];
subject to COMP2_T7_6 {t in (arrival6 + 1)..end6}:
key_T7[t] = key_T7[end5];
subject to COMP2_T8_6 {t in (arrival6 + 1)..end6}:
key_T8[t] = key_T8[end5];
# parcel 7: Crude 8 100% => T2-T5
subject to COMP2_T1_7 {t in (arrival7 + 1)..end7}:
key_T1[t] = key_T1[end6];
subject to COMP2_T2_7 {t in (arrival7 + 1)..end7}:
key_T2[t] = key_T2[t-1] + (key_parcel7 - key_T2[t-1])*q12[t-1]/vol2[t-1];
subject to COMP2_T3_7 {t in (arrival7 + 1)..end7}:
key_T3[t] = key_T3[t-1] + (key_parcel7 - key_T3[t-1])*q13[t-1]/vol3[t-1];
subject to COMP2_T4_7 {t in (arrival7 + 1)..end7}:
key_T4[t] = key_T4[t-1] + (key_parcel7 - key_T4[t-1])*q14[t-1]/vol4[t-1];
subject to COMP2_T5_7 {t in (arrival7 + 1)..end7}:
key_T2[t] = key_T5[t-1] + (key_parcel7 - key_T5[t-1])*q15[t-1]/vol5[t-1];
```

```
subject to COMP2_T6_7 {t in (arrival7 + 1)..end7}:
key_T6[t] = key_T6[end6];
subject to COMP2_T7_7 {t in (arrival7 + 1)..end7}:
key_T7[t] = key_T7[end6];
subject to COMP2_T8_7 {t in (arrival7 + 1)..end7}:
key_T8[t] = key_T8[end6];
# parcel 8: Crude 3 100% => T1, T6-T8
subject to COMP2_T1_8 {t in (arrival8 + 1)..end8}:
key_T1[t] = key_T1[t-1] + (key_parcel8 - key_T1[t-1])*q11[t-1]/vol1[t-1];
subject to COMP2_T2_8 {t in (arrival8 + 1)..end8}:
key_T2[t] = key_T2[end7];
subject to COMP2_T3_8 {t in (arrival8 + 1)..end8}:
key_T3[t] = key_T3[end7];
subject to COMP2_T4_8 {t in (arrival8 + 1)..end8}:
key_T4[t] = key_T4[end7];
subject to COMP2_T5_8 {t in (arrival8 + 1)..end8}:
key_T5[t] = key_T5[end7];
subject to COMP2_T6_8 {t in (arrival8 + 1)..end8}:
key_T6[t] = key_T6[t-1] + (key_parcel8 - key_T6[t-1])*q16[t-1]/vol6[t-1];
subject to COMP2_T7_8 {t in (arrival8 + 1)..end8}:
key_T7[t] = key_T7[t-1] + (key_parcel8 - key_T7[t-1])*q17[t-1]/vol7[t-1];
subject to COMP2_T8_8 {t in (arrival8 + 1)..end8}:
key_T8[t] = key_T8[t-1] + (key_parcel8 - key_T8[t-1])*q18[t-1]/vol8[t-1];
# After parcel 8
subject to COMP2_T1_A8 {t in (end8 + 1)..T}: key_T1[t] = key_T1[end8];
subject to COMP2_T2_A8 {t in (end8 + 1)..T}: key_T2[t] = key_T2[end8];
subject to COMP2_T3_A8 {t in (end8 + 1)..T}: key_T3[t] = key_T3[end8];
subject to COMP2_T4_A8 {t in (end8 + 1)..T}: key_T4[t] = key_T4[end8];
subject to COMP2_T5_A8 {t in (end8 + 1)..T}: key_T5[t] = key_T5[end8];
subject to COMP2_T6_A8 {t in (end8 + 1)..T}: key_T6[t] = key_T6[end8];
subject to COMP2_T7_A8 {t in (end8 + 1)..T}: key_T7[t] = key_T7[end8];
subject to COMP2_T8_A8 {t in (end8 + 1)..T}: key_T8[t] = key_T8[end8];
############################################################
# Flow Constraints
```

```
#############################################################
#####################################################
# Pipeline flow
#####################################################
subject to PIPEFLOW {t in 0 .. T-1}:
(q11[t] + q12[t] + q13[t] + q14[t] + q15[t] + q16[t] +
q17[t] + q18[t]) <= q_max_pipe;
subject to ARRIVAL_PREVIOUS {t in 0..(arrival1 - 1)}:
q11[t] + q12[t] + q13[t] + q14[t] + q15[t] + q16[t] +
q17[t] + q18[t] = 0;
# Only allow flow on tanks that can receive the crude parcel
subject to ARRIVAL_Q12_1 {t in (arrival1)..(end1 - 1)}: q12[t] = 0;
subject to ARRIVAL_Q13_1 {t in (arrival1)..(end1 - 1)}: q13[t] = 0;
subject to ARRIVAL_Q14_1 {t in (arrival1)..(end1 - 1)}: q14[t] = 0;
subject to ARRIVAL_Q15_1 {t in (arrival1)..(end1 - 1)}: q15[t] = 0;
# the first tank choice is free
subject to ARRIVAL_Q11_2 {t in (arrival2 + 1)..(end2 - 1)}: q11[t] = q11[t-1];
subject to ARRIVAL_Q12_2 {t in (arrival2 + 1)..(end2 - 1)}: q12[t] = 0;
subject to ARRIVAL_Q13_2 {t in (arrival2 + 1)..(end2 - 1)}: q13[t] = 0;
subject to ARRIVAL_Q14_2 {t in (arrival2 + 1)..(end2 - 1)}: q14[t] = 0;
subject to ARRIVAL_Q15_2 {t in (arrival2 + 1)..(end2 - 1)}: q15[t] = 0;
subject to ARRIVAL_Q16_2 {t in (arrival2 + 1)..(end2 - 1)}: q16[t] = q16[t-1];
subject to ARRIVAL_Q17_2 {t in (arrival2 + 1)..(end2 - 1)}: q17[t] = q17[t-1];
subject to ARRIVAL_Q18_2 {t in (arrival2 + 1)..(end2 - 1)}: q18[t] = q18[t-1];
# the first tank choice is free
subject to ARRIVAL_Q11_3 {t in (arrival3 + 1)..(end3 - 1)}: q11[t] = q11[t-1];
subject to ARRIVAL_Q12_3 {t in (arrival3 + 1)..(end3 - 1)}: q12[t] = 0;
subject to ARRIVAL_Q13_3 {t in (arrival3 + 1)..(end3 - 1)}: q13[t] = 0;
subject to ARRIVAL_Q14_3 {t in (arrival3 + 1)..(end3 - 1)}: q14[t] = 0;
subject to ARRIVAL_Q15_3 {t in (arrival3 + 1)..(end3 - 1)}: q15[t] = 0;
subject to ARRIVAL_Q16_3 {t in (arrival3 + 1)..(end3 - 1)}: q16[t] = q16[t-1];
subject to ARRIVAL_Q17_3 {t in (arrival3 + 1)..(end3 - 1)}: q17[t] = q17[t-1];
subject to ARRIVAL_Q18_3 {t in (arrival3 + 1)..(end3 - 1)}: q18[t] = q18[t-1];
subject to ARRIVAL_CDU_1 {t in (arrival1)..(end3 - 1)}: q11[t] * q1C[t] = 0;
```

```
subject to ARRIVAL_CDU_6 {t in (arrival1)..(end3 - 1)}: q16[t] * q6C[t] = 0;
subject to ARRIVAL_CDU_7 {t in (arrival1)..(end3 - 1)}: q17[t] * q7C[t] = 0;
subject to ARRIVAL_CDU_8 {t in (arrival1)..(end3 - 1)}: q18[t] * q8C[t] = 0;
subject to ARRIVAL_CDU_1_SETTLING_1 {t in (end1)..(end1 +
settling_time - 1)}: q11[end1 - 1] * q1C[t] = 0;
subject to ARRIVAL_CDU_6_SETTLING_1 {t in (end1)..(end1 +
settling_time - 1)}: q16[end1 - 1] * q6C[t] = 0;
subject to ARRIVAL_CDU_7_SETTLING_1 {t in (end1)..(end1 +
settling_time - 1)}: q17[end1 - 1] * q7C[t] = 0;
subject to ARRIVAL_CDU_8_SETTLING_1 {t in (end1)..(end1 +
settling_time - 1)}: q18[end1 - 1] * q8C[t] = 0;
subject to ARRIVAL_CDU_1_SETTLING_2 {t in (end2)..(end2 +
settling_time - 1)}: q11[end2 - 1] * q1C[t] = 0;
subject to ARRIVAL_CDU_6_SETTLING_2 {t in (end2)..(end2 +
settling_time - 1)}: q16[end2 - 1] * q6C[t] = 0;
subject to ARRIVAL_CDU_7_SETTLING_2 {t in (end2)..(end2 +
settling_time - 1)}: q17[end2 - 1] * q7C[t] = 0;
subject to ARRIVAL_CDU_8_SETTLING_2 {t in (end2)..(end2 +
settling_time - 1)}: q18[end2 - 1] * q8C[t] = 0;
subject to ARRIVAL_CDU_1_SETTLING_3 {t in (end3)..(end3 +
settling_time - 1)}: q11[end3 - 1] * q1C[t] = 0;
subject to ARRIVAL_CDU_6_SETTLING_3 {t in (end3)..(end3 +
settling_time - 1)}: q16[end3 - 1] * q6C[t] = 0;
subject to ARRIVAL_CDU_7_SETTLING_3 {t in (end3)..(end3 +
settling_time - 1)}: q17[end3 - 1] * q7C[t] = 0;
subject to ARRIVAL_CDU_8_SETTLING_3 {t in (end3)..(end3 +
settling_time - 1)}: q18[end3 - 1] * q8C[t] = 0;
subject to ARRIVAL_Q11_4 {t in (arrival4 + 1)..(end4 - 1)}:
q11[t] = 0;
subject to ARRIVAL_Q12_4 {t in (arrival4 + 1)..(end4 - 1)}:
q12[t] = q12[t-1];
subject to ARRIVAL_Q13_4 {t in (arrival4 + 1)..(end4 - 1)}:
q13[t] = q13[t-1];
subject to ARRIVAL_Q14_4 {t in (arrival4 + 1)..(end4 - 1)}:
```

```
q14[t] = q14[t-1];
subject to ARRIVAL_Q15_4 {t in (arrival4 + 1)..(end4 - 1)}:
q15[t] = q15[t-1];
subject to ARRIVAL_Q16_4 {t in (arrival4 + 1)..(end4 - 1)}:
q16[t] = 0;
subject to ARRIVAL_Q17_4 {t in (arrival4 + 1)..(end4 - 1)}:
q17[t] = 0;
subject to ARRIVAL_Q18_4 {t in (arrival4 + 1)..(end4 - 1)}:
q18[t] = 0;
subject to ARRIVAL_CDU_2 {t in (arrival4)..(end4 - 1)}:
q12[t] * (q2A[t] + q2B[t]) = 0;
subject to ARRIVAL_CDU_3 {t in (arrival4)..(end4 - 1)}:
q13[t] * (q3A[t] + q3B[t]) = 0;
subject to ARRIVAL_CDU_4 {t in (arrival4)..(end4 - 1)}:
q14[t] * (q4A[t] + q4B[t]) = 0;
subject to ARRIVAL_CDU_5 {t in (arrival4)..(end4 - 1)}:
q15[t] * (q5A[t] + q5B[t]) = 0;
subject to ARRIVAL_CDU_2_SETTLING_4 {t in (end4)..(end4 +
settling_time - 1)}: q12[end4 - 1] * (q2A[t] + q2B[t]) = 0;
subject to ARRIVAL_CDU_3_SETTLING_4 {t in (end4)..(end4 +
settling_time - 1)}: q13[end4 - 1] * (q3A[t] + q3B[t]) = 0;
subject to ARRIVAL_CDU_4_SETTLING_4 {t in (end4)..(end4 +
settling_time - 1)}: q14[end4 - 1] * (q4A[t] + q4B[t]) = 0;
subject to ARRIVAL_CDU_5_SETTLING_4 {t in (end4)..(end4 +
settling_time - 1)}: q15[end4 - 1] * (q5A[t] + q5B[t]) = 0;
subject to ARRIVAL_IN_BETWEEN {t in (end4)..(arrival5 - 1)}:
q11[t] + q12[t] + q13[t] + q14[t] + q15[t] + q16[t] + q17[t] + q18[t] = 0;
subject to ARRIVAL_Q11_5 {t in (arrival5 + 1)..(end5 - 1)}: q11[t] = 0;
subject to ARRIVAL_Q12_5 {t in (arrival5 + 1)..(end5 - 1)}: q12[t] = q12[t-1];
subject to ARRIVAL_Q13_5 {t in (arrival5 + 1)..(end5 - 1)}: q13[t] = q13[t-1];
subject to ARRIVAL_Q14_5 {t in (arrival5 + 1)..(end5 - 1)}: q14[t] = q14[t-1];
subject to ARRIVAL_Q15_5 {t in (arrival5 + 1)..(end5 - 1)}: q15[t] = q15[t-1];
subject to ARRIVAL_Q16_5 {t in (arrival5 + 1)..(end5 - 1)}: q16[t] = 0;
subject to ARRIVAL_Q17_5 {t in (arrival5 + 1)..(end5 - 1)}: q17[t] = 0;
```

```
subject to ARRIVAL_Q18_5 {t in (arrival5 + 1)..(end5 - 1)}: q18[t] = 0;
subject to ARRIVAL_Q11_6 {t in (arrival6 + 1)..(end6 - 1)}: q11[t] = 0;
subject to ARRIVAL_Q12_6 {t in (arrival6 + 1)..(end6 - 1)}: q12[t] = q12[t-1];
subject to ARRIVAL_Q13_6 {t in (arrival6 + 1)..(end6 - 1)}: q13[t] = q13[t-1];
subject to ARRIVAL_Q14_6 {t in (arrival6 + 1)..(end6 - 1)}: q14[t] = q14[t-1];
subject to ARRIVAL_Q15_6 {t in (arrival6 + 1)..(end6 - 1)}: q15[t] = q15[t-1];
subject to ARRIVAL_Q16_6 {t in (arrival6 + 1)..(end6 - 1)}: q16[t] = 0;
subject to ARRIVAL_Q17_6 {t in (arrival6 + 1)..(end6 - 1)}: q17[t] = 0;
subject to ARRIVAL_Q18_6 {t in (arrival6 + 1)..(end6 - 1)}: q18[t] = 0;
subject to ARRIVAL_Q11_7 {t in (arrival7 + 1)..(end7 - 1)}: q11[t] = 0;
subject to ARRIVAL_Q12_7 {t in (arrival7 + 1)..(end7 - 1)}: q12[t] = q12[t-1];
subject to ARRIVAL_Q13_7 {t in (arrival7 + 1)..(end7 - 1)}: q13[t] = q13[t-1];
subject to ARRIVAL_Q14_7 {t in (arrival7 + 1)..(end7 - 1)}: q14[t] = q14[t-1];
subject to ARRIVAL_Q15_7 {t in (arrival7 + 1)..(end7 - 1)}: q15[t] = q15[t-1];
subject to ARRIVAL_Q16_7 {t in (arrival7 + 1)..(end7 - 1)}: q16[t] = 0;
subject to ARRIVAL_Q17_7 {t in (arrival7 + 1)..(end7 - 1)}: q17[t] = 0;
subject to ARRIVAL_Q18_7 {t in (arrival7 + 1)..(end7 - 1)}: q18[t] = 0;
subject to ARRIVAL_CDU_2_ {t in (arrival5)..(end7 - 1)}: q12[t] * (q2A[t] + q2B[t]) = 0;
subject to ARRIVAL_CDU_3_ {t in (arrival5)..(end7 - 1)}: q13[t] * (q3A[t] + q3B[t]) = 0;
subject to ARRIVAL_CDU_4_ {t in (arrival5)..(end7 - 1)}: q14[t] * (q4A[t] + q4B[t]) = 0;
subject to ARRIVAL_CDU_5_ {t in (arrival5)..(end7 - 1)}: q15[t] * (q5A[t] + q5B[t]) = 0;
subject to ARRIVAL_CDU_2_SETTLING_5 {t in (end5)..(end5 +
settling_time - 1)}: q12[end5 - 1] * (q2A[t] + q2B[t]) = 0;
subject to ARRIVAL_CDU_3_SETTLING_5 {t in (end5)..(end5 +
settling_time - 1)}: q13[end5 - 1] * (q3A[t] + q3B[t]) = 0;
subject to ARRIVAL_CDU_4_SETTLING_5 {t in (end5)..(end5 +
settling_time - 1)}: q14[end5 - 1] * (q4A[t] + q4B[t]) = 0;
subject to ARRIVAL_CDU_5_SETTLING_5 {t in (end5)..(end5 +
settling_time - 1)}: q15[end5 - 1] * (q5A[t] + q5B[t]) = 0;
subject to ARRIVAL_CDU_2_SETTLING_6 {t in (end6)..(end6 +
settling_time - 1)}: q12[end6 - 1] * (q2A[t] + q2B[t]) = 0;
subject to ARRIVAL_CDU_3_SETTLING_6 {t in (end6)..(end6 +
settling_time - 1)}: q13[end6 - 1] * (q3A[t] + q3B[t]) = 0;
subject to ARRIVAL_CDU_4_SETTLING_6 {t in (end6)..(end6 +
```

```
settling_time - 1)}: q14[end6 - 1] * (q4A[t] + q4B[t]) = 0;
subject to ARRIVAL_CDU_5_SETTLING_6 {t in (end6)..(end6 +
settling_time - 1)}: q15[end6 - 1] * (q5A[t] + q5B[t]) = 0;
subject to ARRIVAL_CDU_2_SETTLING_7 {t in (end7)..(end7 +
settling_time - 1)}: q12[end7 - 1] * (q2A[t] + q2B[t]) = 0;
subject to ARRIVAL_CDU_3_SETTLING_7 {t in (end7)..(end7 +
settling_time - 1)}: q13[end7 - 1] * (q3A[t] + q3B[t]) = 0;
subject to ARRIVAL_CDU_4_SETTLING_7 {t in (end7)..(end7 +
settling_time - 1)}: q14[end7 - 1] * (q4A[t] + q4B[t]) = 0;
subject to ARRIVAL_CDU_5_SETTLING_7 {t in (end7)..(end7 +
settling_time - 1)}: q15[end7 - 1] * (q5A[t] + q5B[t]) = 0;
subject to ARRIVAL_Q11_8 {t in (arrival8 + 1)..(end8 - 1)}: q11[t] = q11[t-1];
subject to ARRIVAL_Q12_8 {t in (arrival8 + 1)..(end8 - 1)}: q12[t] = 0;
subject to ARRIVAL_Q13_8 {t in (arrival8 + 1)..(end8 - 1)}: q13[t] = 0;
subject to ARRIVAL_Q14_8 {t in (arrival8 + 1)..(end8 - 1)}: q14[t] = 0;
subject to ARRIVAL_Q15_8 {t in (arrival8 + 1)..(end8 - 1)}: q15[t] = 0;
subject to ARRIVAL_Q16_8 {t in (arrival8 + 1)..(end8 - 1)}: q16[t] = q16[t-1];
subject to ARRIVAL_Q17_8 {t in (arrival8 + 1)..(end8 - 1)}: q17[t] = q17[t-1];
subject to ARRIVAL_Q18_8 {t in (arrival8 + 1)..(end8 - 1)}: q18[t] = q18[t-1];
subject to ARRIVAL_CDU_1_ {t in (arrival8)..(end8 - 1)}: q11[t] * q1C[t] = 0;
subject to ARRIVAL_CDU_6_ {t in (arrival8)..(end8 - 1)}: q16[t] * q6C[t] = 0;
subject to ARRIVAL_CDU_7_ {t in (arrival8)..(end8 - 1)}: q17[t] * q7C[t] = 0;
subject to ARRIVAL_CDU_8_ {t in (arrival8)..(end8 - 1)}: q18[t] * q8C[t] = 0;
subject to ARRIVAL_CDU_1_SETTLING_8 {t in (end8)..(end8 +
settling_time - 1)}: q11[end8 - 1] * q1C[t] = 0;
subject to ARRIVAL_CDU_6_SETTLING_8 {t in (end8)..(end8 +
settling_time - 1)}: q16[end8 - 1] * q6C[t] = 0;
subject to ARRIVAL_CDU_7_SETTLING_8 {t in (end8)..(end8 +
settling_time - 1)}: q17[end8 - 1] * q7C[t] = 0;
subject to ARRIVAL_CDU_8_SETTLING_8 {t in (end8)..(end8 +
settling_time - 1)}: q18[end8 - 1] * q8C[t] = 0;
subject to ARRIVAL_END {t in (end8)..(T-1)}:
q11[t] + q12[t] + q13[t] + q14[t] + q15[t] + q16[t] + q17[t] + q18[t] = 0;
#############################################################################
```

```
# CDU flow
#############################################################################################
subject to CDU_FLOW_A {t in 0..T-1}:
(q2A[t]+ q3A[t]+ q4A[t]+ q5A[t]) <= feedflow;
subject to CDU_FLOW_B {t in 0..T-1}:
(q2B[t]+ q3B[t]+ q4B[t]+ q5B[t]) <= feedflow;
subject to CDU_FLOW_C {t in 0..T-1}:
(q1C[t] + q6C[t] + q7C[t] + q8C[t]) <= feedflow;
subject to CDU_FLOW_MIN_A {t in 0..T-1}:
(q2A[t]+ q3A[t]+ q4A[t]+ q5A[t]) >= min_feedflow;
subject to CDU_FLOW_MIN_B {t in 0..T-1}:
(q2B[t]+ q3B[t]+ q4B[t]+ q5B[t]) >= min_feedflow;
subject to CDU_FLOW_MIN_C {t in 0..T-1}:
(q1C[t] + q6C[t] + q7C[t] + q8C[t]) >= min_feedflow;
subject to CDU_VOL_A: cdu_vol = dt*(sum {t in 0..T-1}
(q2A[t]+ q3A[t]+ q4A[t]+ q5A[t]));
subject to CDU_VOL_B: cdu_vol = dt*(sum {t in 0..T-1}
(q2B[t]+ q3B[t]+ q4B[t]+ q5B[t]));
subject to CDU_VOL_C: cdu_vol = dt*(sum {t in 0..T-1}
(q1C[t] + q6C[t] + q7C[t] + q8C[t]));
#############################################################################################
# At most 2 tanks can feed CDU
#############################################################################################
subject to AT_MOST_TWO_TANKS_1C {t in 0..T-1}:
twocduflow_1C[t] = q1C[t]* (q6C[t] + q7C[t] + q8C[t]);
subject to AT_MOST_TWO_TANKS_6C {t in 0..T-1}:
twocduflow_6C[t] = q6C[t]* (q7C[t] + q8C[t]);
subject to AT_MOST_TWO_TANKS_7C {t in 0..T-1}:
twocduflow_7C[t] = q7C[t]* q8C[t];
subject to AT_MOST_TWO_TANKS_2A {t in 0..T-1}:
twocduflow_2A[t] = q2A[t]* (q3A[t] + q4A[t] + q5A[t]);
subject to AT_MOST_TWO_TANKS_3A {t in 0..T-1}:
twocduflow_3A[t] = q3A[t]* (q4A[t] + q5A[t]);
subject to AT_MOST_TWO_TANKS_4A {t in 0..T-1}:
```

```
twocduflow_4A[t] = q4A[t]* q5A[t];
subject to AT_MOST_TWO_TANKS_2B {t in 0..T-1}:
twocduflow_2B[t] = q2B[t]* (q3B[t] + q4B[t] + q5B[t]);
subject to AT_MOST_TWO_TANKS_3B {t in 0..T-1}:
twocduflow_3B[t] = q3B[t]* (q4B[t] + q5B[t]);
subject to AT_MOST_TWO_TANKS_4B {t in 0..T-1}:
twocduflow_4B[t] = q4B[t]* q5B[t];
subject to CDU_TANK_FLOW_DAY1_T1_C {t in 1..12}:
q1C[t] = q1C[t-1];
subject to CDU_TANK_FLOW_DAY1_T6_C {t in 1..12}: q6C[t] = q6C[t-1];
subject to CDU_TANK_FLOW_DAY1_T7_C {t in 1..12}: q7C[t] = q7C[t-1];
subject to CDU_TANK_FLOW_DAY1_T8_C {t in 1..12}: q8C[t] = q8C[t-1];
subject to CDU_TANK_FLOW_DAY1B_T1_C {t in 14..24}: q1C[t] = q1C[t-1];
subject to CDU_TANK_FLOW_DAY1B_T6_C {t in 14..24}: q6C[t] = q6C[t-1];
subject to CDU_TANK_FLOW_DAY1B_T7_C {t in 14..24}: q7C[t] = q7C[t-1];
subject to CDU_TANK_FLOW_DAY1B_T8_C {t in 14..24}: q8C[t] = q8C[t-1];
subject to CDU_TANK_FLOW_DAY2_T1_C {t in 26..36}: q1C[t] = q1C[t-1];
subject to CDU_TANK_FLOW_DAY2_T6_C {t in 26..36}: q6C[t] = q6C[t-1];
subject to CDU_TANK_FLOW_DAY2_T7_C {t in 26..36}: q7C[t] = q7C[t-1];
subject to CDU_TANK_FLOW_DAY2_T8_C {t in 26..36}: q8C[t] = q8C[t-1];
subject to CDU_TANK_FLOW_DAY2B_T1_C {t in 38..48}: q1C[t] = q1C[t-1];
subject to CDU_TANK_FLOW_DAY2B_T6_C {t in 38..48}: q6C[t] = q6C[t-1];
subject to CDU_TANK_FLOW_DAY2B_T7_C {t in 38..48}: q7C[t] = q7C[t-1];
subject to CDU_TANK_FLOW_DAY2B_T8_C {t in 38..48}: q8C[t] = q8C[t-1];
subject to CDU_TANK_FLOW_DAY3_T1_C {t in 50..60}: q1C[t] = q1C[t-1];
subject to CDU_TANK_FLOW_DAY3_T6_C {t in 50..60}: q6C[t] = q6C[t-1];
subject to CDU_TANK_FLOW_DAY3_T7_C {t in 50..60}: q7C[t] = q7C[t-1];
subject to CDU_TANK_FLOW_DAY3_T8_C {t in 50..60}: q8C[t] = q8C[t-1];
subject to CDU_TANK_FLOW_DAY3B_T1_C {t in 62..71}: q1C[t] = q1C[t-1];
subject to CDU_TANK_FLOW_DAY3B_T6_C {t in 62..71}: q6C[t] = q6C[t-1];
subject to CDU_TANK_FLOW_DAY3B_T7_C {t in 62..71}: q7C[t] = q7C[t-1];
subject to CDU_TANK_FLOW_DAY3B_T8_C {t in 62..71}: q8C[t] = q8C[t-1];
subject to CDU_TANK_FLOW_DAY4_T1_C {t in 73..83}: q1C[t] = q1C[t-1];
subject to CDU_TANK_FLOW_DAY4_T6_C {t in 73..83}: q6C[t] = q6C[t-1];
```

```
subject to CDU_TANK_FLOW_DAY4_T7_C {t in 73..83}: q7C[t] = q7C[t-1];
subject to CDU_TANK_FLOW_DAY4_T8_C {t in 73..83}: q8C[t] = q8C[t-1];
subject to CDU_TANK_FLOW_DAY4B_T1_C {t in 85..96}: q1C[t] = q1C[t-1];
subject to CDU_TANK_FLOW_DAY4B_T6_C {t in 85..96}: q6C[t] = q6C[t-1];
subject to CDU_TANK_FLOW_DAY4B_T7_C {t in 85..96}: q7C[t] = q7C[t-1];
subject to CDU_TANK_FLOW_DAY4B_T8_C {t in 85..96}: q8C[t] = q8C[t-1];
subject to CDU_TANK_FLOW_DAY5_T1_C {t in 98..108}: q1C[t] = q1C[t-1];
subject to CDU_TANK_FLOW_DAY5_T6_C {t in 98..108}: q6C[t] = q6C[t-1];
subject to CDU_TANK_FLOW_DAY5_T7_C {t in 98..108}: q7C[t] = q7C[t-1];
subject to CDU_TANK_FLOW_DAY5_T8_C {t in 98..108}: q8C[t] = q8C[t-1];
subject to CDU_TANK_FLOW_DAY5B_T1_C {t in 110..120}: q1C[t] = q1C[t-1];
subject to CDU_TANK_FLOW_DAY5B_T6_C {t in 110..120}: q6C[t] = q6C[t-1];
subject to CDU_TANK_FLOW_DAY5B_T7_C {t in 110..120}: q7C[t] = q7C[t-1];
subject to CDU_TANK_FLOW_DAY5B_T8_C {t in 110..120}: q8C[t] = q8C[t-1];
subject to CDU_TANK_FLOW_DAY6_T1_C {t in 122..130}: q1C[t] = q1C[t-1];
subject to CDU_TANK_FLOW_DAY6_T6_C {t in 122..130}: q6C[t] = q6C[t-1];
subject to CDU_TANK_FLOW_DAY6_T7_C {t in 122..130}: q7C[t] = q7C[t-1];
subject to CDU_TANK_FLOW_DAY6_T8_C {t in 122..130}: q8C[t] = q8C[t-1];
subject to CDU_TANK_FLOW_DAY6B_T1_C {t in 132..144}: q1C[t] = q1C[t-1];
subject to CDU_TANK_FLOW_DAY6B_T6_C {t in 132..144}: q6C[t] = q6C[t-1];
subject to CDU_TANK_FLOW_DAY6B_T7_C {t in 132..144}: q7C[t] = q7C[t-1];
subject to CDU_TANK_FLOW_DAY6B_T8_C {t in 132..144}: q8C[t] = q8C[t-1];
subject to CDU_TANK_FLOW_DAY7_T1_C {t in 146..T-1}: q1C[t] = q1C[t-1];
subject to CDU_TANK_FLOW_DAY7_T6_C {t in 146..T-1}: q6C[t] = q6C[t-1];
subject to CDU_TANK_FLOW_DAY7_T7_C {t in 146..T-1}: q7C[t] = q7C[t-1];
subject to CDU_TANK_FLOW_DAY7_T8_C {t in 146..T-1}: q8C[t] = q8C[t-1];
subject to CDU_TANK_FLOW_DAY1_T2_A {t in 1..24}: q2A[t] = q2A[t-1];
subject to CDU_TANK_FLOW_DAY1_T3_A {t in 1..24}: q3A[t] = q3A[t-1];
subject to CDU_TANK_FLOW_DAY1_T4_A {t in 1..24}: q4A[t] = q4A[t-1];
subject to CDU_TANK_FLOW_DAY1_T5_A {t in 1..24}: q5A[t] = q5A[t-1];
subject to CDU_TANK_FLOW_DAY2_T2_A {t in 26..48}: q2A[t] = q2A[t-1];
subject to CDU_TANK_FLOW_DAY2_T3_A {t in 26..48}: q3A[t] = q3A[t-1];
subject to CDU_TANK_FLOW_DAY2_T4_A {t in 26..48}: q4A[t] = q4A[t-1];
subject to CDU_TANK_FLOW_DAY2_T5_A {t in 26..48}: q5A[t] = q5A[t-1];
```

```
subject to CDU_TANK_FLOW_DAY3_T2_A {t in 50..71}: q2A[t] = q2A[t-1];
subject to CDU_TANK_FLOW_DAY3_T3_A {t in 50..71}: q3A[t] = q3A[t-1];
subject to CDU_TANK_FLOW_DAY3_T4_A {t in 50..71}: q4A[t] = q4A[t-1];
subject to CDU_TANK_FLOW_DAY3_T5_A {t in 50..71}: q5A[t] = q5A[t-1];
subject to CDU_TANK_FLOW_DAY4_T2_A {t in 73..96}: q2A[t] = q2A[t-1];
subject to CDU_TANK_FLOW_DAY4_T3_A {t in 73..96}: q3A[t] = q3A[t-1];
subject to CDU_TANK_FLOW_DAY4_T4_A {t in 73..96}: q4A[t] = q4A[t-1];
subject to CDU_TANK_FLOW_DAY4_T5_A {t in 73..96}: q5A[t] = q5A[t-1];
subject to CDU_TANK_FLOW_DAY5_T2_A {t in 98..120}: q2A[t] = q2A[t-1];
subject to CDU_TANK_FLOW_DAY5_T3_A {t in 98..120}: q3A[t] = q3A[t-1];
subject to CDU_TANK_FLOW_DAY5_T4_A {t in 98..120}: q4A[t] = q4A[t-1];
subject to CDU_TANK_FLOW_DAY5_T5_A {t in 98..120}: q5A[t] = q5A[t-1];
subject to CDU_TANK_FLOW_DAY6_T2_A {t in 122..144}: q2A[t] = q2A[t-1];
subject to CDU_TANK_FLOW_DAY6_T3_A {t in 122..144}: q3A[t] = q3A[t-1];
subject to CDU_TANK_FLOW_DAY6_T4_A {t in 122..144}: q4A[t] = q4A[t-1];
subject to CDU_TANK_FLOW_DAY6_T5_A {t in 122..144}: q5A[t] = q5A[t-1];
subject to CDU_TANK_FLOW_DAY7_T2_A {t in 146..T-1}: q2A[t] = q2A[t-1];
subject to CDU_TANK_FLOW_DAY7_T3_A {t in 146..T-1}: q3A[t] = q3A[t-1];
subject to CDU_TANK_FLOW_DAY7_T4_A {t in 146..T-1}: q4A[t] = q4A[t-1];
subject to CDU_TANK_FLOW_DAY7_T5_A {t in 146..T-1}: q5A[t] = q5A[t-1];
subject to CDU_TANK_FLOW_DAY1_T2_B {t in 1..24}: q2B[t] = q2B[t-1];
subject to CDU_TANK_FLOW_DAY1_T3_B {t in 1..24}: q3B[t] = q3B[t-1];
subject to CDU_TANK_FLOW_DAY1_T4_B {t in 1..24}: q4B[t] = q4B[t-1];
subject to CDU_TANK_FLOW_DAY1_T5_B {t in 1..24}: q5B[t] = q5B[t-1];
subject to CDU_TANK_FLOW_DAY2_T2_B {t in 26..48}: q2B[t] = q2B[t-1];
subject to CDU_TANK_FLOW_DAY2_T3_B {t in 26..48}: q3B[t] = q3B[t-1];
subject to CDU_TANK_FLOW_DAY2_T4_B {t in 26..48}: q4B[t] = q4B[t-1];
subject to CDU_TANK_FLOW_DAY2_T5_B {t in 26..48}: q5B[t] = q5B[t-1];
subject to CDU_TANK_FLOW_DAY3_T2_B {t in 50..71}: q2B[t] = q2B[t-1];
subject to CDU_TANK_FLOW_DAY3_T3_B {t in 50..71}: q3B[t] = q3B[t-1];
subject to CDU_TANK_FLOW_DAY3_T4_B {t in 50..71}: q4B[t] = q4B[t-1];
subject to CDU_TANK_FLOW_DAY3_T5_B {t in 50..71}: q5B[t] = q5B[t-1];
subject to CDU_TANK_FLOW_DAY4_T2_B {t in 73..96}: q2B[t] = q2B[t-1];
subject to CDU_TANK_FLOW_DAY4_T3_B {t in 73..96}: q3B[t] = q3B[t-1];
```

```
subject to CDU_TANK_FLOW_DAY4_T4_B {t in 73..96}: q4B[t] = q4B[t-1];
subject to CDU_TANK_FLOW_DAY4_T5_B {t in 73..96}: q5B[t] = q5B[t-1];
subject to CDU_TANK_FLOW_DAY5_T2_B {t in 98..120}: q2B[t] = q2B[t-1];
subject to CDU_TANK_FLOW_DAY5_T3_B {t in 98..120}: q3B[t] = q3B[t-1];
subject to CDU_TANK_FLOW_DAY5_T4_B {t in 98..120}: q4B[t] = q4B[t-1];
subject to CDU_TANK_FLOW_DAY5_T5_B {t in 98..120}: q5B[t] = q5B[t-1];
subject to CDU_TANK_FLOW_DAY6_T2_B {t in 122..144}: q2B[t] = q2B[t-1];
subject to CDU_TANK_FLOW_DAY6_T3_B {t in 122..144}: q3B[t] = q3B[t-1];
subject to CDU_TANK_FLOW_DAY6_T4_B {t in 122..144}: q4B[t] = q4B[t-1];
subject to CDU_TANK_FLOW_DAY6_T5_B {t in 122..144}: q5B[t] = q5B[t-1];
subject to CDU_TANK_FLOW_DAY7_T2_B {t in 146..T-1}: q2B[t] = q2B[t-1];
subject to CDU_TANK_FLOW_DAY7_T3_B {t in 146..T-1}: q3B[t] = q3B[t-1];
subject to CDU_TANK_FLOW_DAY7_T4_B {t in 146..T-1}: q4B[t] = q4B[t-1];
subject to CDU_TANK_FLOW_DAY7_T5_B {t in 146..T-1}: q5B[t] = q5B[t-1];
#######################################################
# FEED KEY COMPONENT LIMITS
#######################################################
var feed_key_A {0..T-1} <= feed_key_max_A * feedflow;
var feed_key_B {0..T-1} <= feed_key_max_B * feedflow;
var feed_key_C {0..T-1} <= feed_key_max_C * feedflow;
subject to CDU_KEY_A_T_MAX {t in 0..T-1}:
feed_key_A[t] <= (q2A[t]+q3A[t]+q4A[t]+q5A[t]) * feed_key_max_A;
subject to CDU_KEY_B_T_MAX {t in 0..T-1}:
feed_key_B[t] <= (q2B[t]+q3B[t]+q4B[t]+q5B[t]) * feed_key_max_B;
subject to CDU_KEY_C_T_MAX {t in 0..T-1}:
feed_key_C[t] <= (q1C[t]+q6C[t]+q7C[t]+q8C[t]) * feed_key_max_C;
subject to MIN_FEED_CRUDE_A {t in 0..T-1}:
feed_key_A[t]  >= feed_key_min_A * min_feedflow;
subject to MIN_FEED_CRUDE_B {t in 0..T-1}:
feed_key_B[t]  >= feed_key_min_B * min_feedflow;
subject to MIN_FEED_CRUDE_C {t in 0..T-1}:
feed_key_C[t]  >= feed_key_min_C * min_feedflow;
subject to CDU_KEY_A_T_MIN {t in 0..T-1}:
feed_key_A[t] >= (q2A[t]+q3A[t]+q4A[t]+q5A[t]) * feed_key_min_A;
```

```
subject to CDU_KEY_B_T_MIN {t in 0..T-1}:
feed_key_B[t] >= (q2B[t]+q3B[t]+q4B[t]+q5B[t]) * feed_key_min_B;
subject to CDU_KEY_C_T_MIN {t in 0..T-1}:
feed_key_C[t] >= (q1C[t]+q6C[t]+q7C[t]+q8C[t]) * feed_key_min_C;
# CDU A => T2-T5
subject to CDU_KEY_A_T0: feed_key_A[0] =
(q2A[0]*ini_key_T2 + q3A[0]*ini_key_T3 + q4A[0]*ini_key_T4 + q5A[0]*ini_key_T5);
subject to CDU_KEY_A_T {t in 1..T-1}: feed_key_A[t] =
(q2A[t]*key_T2[t] + q3A[t]*key_T3[t] + q4A[t]*key_T4[t] + q5A[t]*key_T5[t]);
# CDU B => T1, T6-T8
subject to CDU_KEY_B_T0: feed_key_B[0] =
(q2B[0]*ini_key_T2 + q3B[0]*ini_key_T3 + q4B[0]*ini_key_T4 + q5B[0]*ini_key_T5);
subject to CDU_KEY_B_T {t in 1..T-1}: feed_key_B[t] =
(q2B[t]*key_T2[t] + q3B[t]*key_T3[t] + q4B[t]*key_T4[t] + q5B[t]*key_T5[t]);
# CDU C => T1, T6-T8
subject to CDU_KEY_C_T0: feed_key_C[0] =
(q1C[0]*ini_key_T1 + q6C[0]*ini_key_T6 + q7C[0]*ini_key_T7 + q8C[0]*ini_key_T8);
subject to CDU_KEY_C_T {t in 1..T-1}: feed_key_C[t] =
(q1C[t]*key_T1[t] + q6C[t]*key_T6[t] + q7C[t]*key_T7[t] + q8C[t]*key_T8[t]);
#############################################################################
# Pipeline -> one Tank
#############################################################################
subject to ONE_PIPE_FLOW_1 {t in 0..T-1}:
q11[t]* (q12[t] + q13[t] + q14[t] + q15[t] + q16[t] + q17[t] + q18[t]) = 0;
subject to ONE_PIPE_FLOW_2 {t in 0..T-1}:
q12[t]* (q13[t] + q14[t] + q15[t] + q16[t] + q17[t] + q18[t]) = 0;
subject to ONE_PIPE_FLOW_3 {t in 0..T-1}:
q13[t]* (q14[t] + q15[t] + q16[t] + q17[t] + q18[t]) = 0;
subject to ONE_PIPE_FLOW_4 {t in 0..T-1}:
q14[t]* (q15[t] + q16[t] + q17[t] + q18[t]) = 0;
subject to ONE_PIPE_FLOW_5 {t in 0..T-1}:
q15[t]* (q16[t] + q17[t] + q18[t]) = 0;
subject to ONE_PIPE_FLOW_6 {t in 0..T-1}:
q16[t]* (q17[t] + q18[t]) = 0;
```

```
subject to ONE_PIPE_FLOW_7 {t in 0..T-1}:
q17[t]* (q18[t]) = 0;

#########################################################################
# Objective Function - components
#########################################################################
var switchover >= 0;
var twocdu >= 0;
var feed_diff >= 0;
var onepipe >= 0;
var settling >= 0;
param netback = 0;
param max_feed_diff;
let max_feed_diff := cdu_vol;
subject to MAX_FEED_DIFF: feed_diff <= max_feed_diff;
param max_onepipe;
let max_onepipe := 1;
subject to MAX_ONEPIPE: onepipe <= max_onepipe;
param max_settling;
let max_settling := 1;
subject to MAX_SETTLING: settling <= max_settling;
param max_switchover;
let max_switchover := 1000;
subject to MAX_SWITCHOVER: switchover <= max_switchover;
param max_twocdu;
let max_twocdu := 100;
subject to MAX_TWOCDU: twocdu <= max_twocdu;
subject to SWITCHOVER_COST:
switchover = sum {t in 1..T-1}
(
(q11[t-1] - q11[t])^2
+ (q12[t-1] - q12[t])^2
+ (q13[t-1] - q13[t])^2
+ (q14[t-1] - q14[t])^2
```

```
+ (q15[t-1] - q15[t])^2
+ (q16[t-1] - q16[t])^2
+ (q1C[t-1] - q1C[t])^2
+ (q2A[t-1] - q2A[t])^2
+ (q3A[t-1] - q3A[t])^2
+ (q4A[t-1] - q4A[t])^2
+ (q5A[t-1] - q5A[t])^2
+ (q2B[t-1] - q2B[t])^2
+ (q3B[t-1] - q3B[t])^2
+ (q4B[t-1] - q4B[t])^2
+ (q5B[t-1] - q5B[t])^2
+ (q6C[t-1] - q6C[t])^2
+ (q7C[t-1] - q7C[t])^2
+ (q8C[t-1] - q8C[t])^2
)/feedflow^2;
subject to FEED_COST: feed_diff = 0;
subject to ONEPIPE_COST: onepipe = 0;
subject to SETTLING_COST: settling = 0;
subject to TWOCDU_COST:
twocdu = sum {t in 0..T-1}
(twocduflow_1C[t] + twocduflow_2A[t] +
twocduflow_3A[t] + twocduflow_4A[t] +
twocduflow_2B[t] + twocduflow_3B[t] +
twocduflow_4B[t] + twocduflow_6C[t] +
twocduflow_7C[t])/feedflow^2;
param W_SW;
let W_SW := 0; # 1;
param W_FEED;
let W_FEED := 1;
param W_PIPE;
let W_PIPE := 1;
param W_CDU;
let W_CDU := 1;
param W_SET;
```

```
let W_SET := 1;
param W_PENALTY;
let W_PENALTY := 0;
param W_COST;
let W_COST := 0;
var cost;
subject to COST: cost =  W_SW * switchover +
W_FEED * feed_diff;
var penalty;
subject to PENALTY: penalty =  W_PIPE * onepipe +
W_CDU * twocdu + W_SET * settling;
minimize obj: W_COST * cost + W_PENALTY * penalty;
```