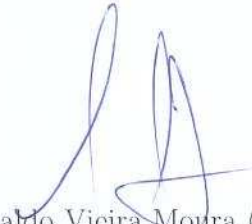


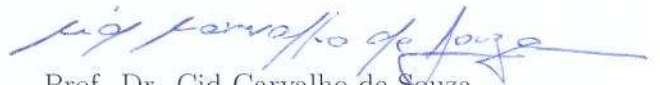
O Problema de Planejamento e Agendamento de Operações em uma Rede de Oleodutos

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Tony Minoru Tamura Lopes e aprovada pela Banca Examinadora.

Campinas, 29 de Julho de 2010.



Prof. Dr. Arnaldo Vieira Moura (Orientador)



Prof. Dr. Cid Carvalho de Souza
(Co-orientador)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

**FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DO IMECC DA UNICAMP**

Bibliotecária: Silvania Renata de Jesus Ribeiro Cirilo – CRB8 /6592

Lopes, Tony Minoru Tamura

L881p O problema de planejamento e agendamento de operações em uma rede de oleodutos/Tony Minoru Tamura Lopes-- Campinas, [S.P. : s.n.], 2010.

Orientador : Arnaldo Vieira Moura ; Cid Carvalho de Souza

Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Programação por restrições. 2. Programação Linear. 3. Oleodutos.. I. Moura, Arnaldo Vieira. II. Souza, Cid Carvalho de. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Título em inglês: The problem of planning and scheduling the operation of an oil pipeline.

Palavras-chave em inglês (Keywords): 1. Constraint programming. 2. Linear programming. 3. Oil pipeline.

Área de concentração: Ciência da Computação.

Titulação: Mestre em Ciência da Computação.

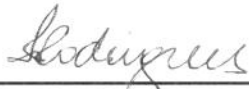
Banca examinadora: Prof. Dr. Arnaldo Vieira Moura - (IC-UNICAMP)
Prof. Dr. Cid Carvalho de Souza - (IC-UNICAMP)
Prof. Dr. Guilherme Pimentel Telles – (IC-UNICAMP)
Profª. Dra. Maria Teresa Moreira Rodrigues – (FEQ-UNICAMP)
Prof. Dr. Ricardo Dahab – (IC-UNICAMP)
Prof. Dr. Carlile Campos Lavor - (IMECC-UNICAMP)

Data da defesa: 29/07/2010

Programa de Pós-Graduação: Mestrado em Ciência da Computação.

TERMO DE APROVAÇÃO

Dissertação Defendida e Aprovada em 29 de julho de 2010, pela Banca examinadora composta pelos Professores Doutores:



Prof. Dr.ª Maria Teresa Moreira Rodrigues
FEQ / UNICAMP



Prof. Dr. Guilherme Pimentel Telles
IC / UNICAMP



Prof. Dr. Arnaldo Vieira Moura
IC / UNICAMP

O Problema de Planejamento e Agendamento de Operações em uma Rede de Oleodutos

Tony Minoru Tamura Lopes¹

Julho de 2010

Banca Examinadora:

- Prof. Dr. Arnaldo Vieira Moura (Orientador)
- Prof. Dr. Guilherme Pimentel Telles
Instituto de Computação, UNICAMP
- Profa. Dra. Maria Teresa Moreira Rodrigues
Faculdade de Engenharia Química, UNICAMP
- Prof. Dr. Ricardo Dahad (Suplente)
Instituto de Computação, UNICAMP
- Prof. Dr. Carlile Campos Lavor (Suplente)
Instituto de Matemática, Estatística e Computação Científica, UNICAMP

¹Suporte financeiro de: Fundação de Amparo à Pesquisa do Estado de São Paulo (processo 05/57343-0), 2006-2008.

Resumo

Um conjunto de órgãos distribuidores de derivados de petróleo, incluindo refinarias e terminais, possuem demandas locais e produções de diferentes produtos ao longo de um dado horizonte de tempo. No entanto, pode não haver estoque local de algum produto para satisfazer a demanda correspondente, ou pode não haver espaço nos tanques para estocar uma produção local. Isso leva à necessidade de transporte dos derivados de petróleo entre os órgãos.

Dentre os diversos modais, a rede de oleodutos é a melhor opção considerando-se custos e riscos ambientais. Em vista de sua grande complexidade operacional, um uso adequado da rede necessita de um planejamento tático composto mensalmente, e de um agendamento detalhado das operações, cobrindo poucos dias, e que deve ser atualizado diariamente. Tanto o planejamento mensal quanto o agendamento diário devem respeitar um grande conjunto de restrições, envolvendo a capacidade dos tanques, taxas de vazões nos oleodutos, níveis de estoques, dentre outras.

Esta dissertação apresenta uma formalização do problema, desenvolvida em dois estágios, representando o planejamento mensal e o agendamento diário. O problema de planejamento recebeu um tratamento inicial heurístico seguido de uma modelagem por fluxo em redes, enquanto o agendamento diário utilizou programação por restrições.

Os modelos foram testados sobre dados fornecidos pela companhia brasileira de petróleo PETROBRAS. Essas instâncias possuem uma das topologias mais complexas quando comparadas a outras redes encontrada na literatura aberta. Os resultados demonstram melhorias significativas sobre a resolução manual desses problemas.

Abstract

A set of oil derivative distribution depots, including refineries and terminals, have local demands for and productions of different products in a given time horizon. However, there may be not enough local stock of some product to satisfy the corresponding demand, or there may not be enough tank capacity to stock the local production. This brings the need for transportation of oil derivatives between the depots.

Among many transportation modes, the network of pipelines is one of the best options when considering cost and environment risks. In order to adequately operate the pipeline network, a two phase planning strategy is developed. First, a tactical pumping plan is composed monthly and, secondly, a more detailed operational schedule, spanning a few days, is updated daily. Both the tactical and the operational plannings must satisfy a large set of operation constraints, involving many restrictions, such as tanks capacities, pipeline flow rates, and stock levels.

This dissertation provides a formalization for the problem along with a decomposition of it in two stages, representing the monthly planning and operational schedule. The tactical stage is solved by applying a heuristic and then with a network flow model, while the operational schedule uses constraint programming.

Our model treats the oil pipeline network that is operated by the Brazilian oil company PETROBRAS. This is one of the most complex and large topologies when compared to other networks treated in the open literature. The model was tested with real-world instances and showed significant improvements over human planning.

Agradecimentos

Agradeço aos meus pais, José e Mitsuko, pelo carinho, amor, paciência e pelo apoio incondicional à minha educação, sem a qual não teria chegado até aqui.

Agradeço também aos meus orientadores, Prof. Arnaldo e Prof. Cid, pela oportunidade, amizade, por me iniciarem na área de pesquisa e por constituírem o modelo de profissional que almejo ser no futuro.

Agradeço meus irmãos e meus amigos pela convivência, pelos ótimos momentos nesses últimos anos e por todo o apoio que me deram. Em especial ao André, pela amizade, força, paciência e grandes idéias nas noites e madrugadas acordadas que passamos desenvolvendo este projeto.

Agradeço à FAPESP, pelo suporte financeiro ao meu projeto de pesquisa. À PETROBRAS, particularmente ao Fernando Marcellino, André Lima, Rômulo Albuquerque, Leandro Barcelos, Enio Medeiros, Décio, Márcio Audi e Rubens Oshiro pelo fornecimento dos dados e, principalmente, pela oportunidade de participar deste projeto. Também agradeço aos funcionários do Instituto de Computação da UNICAMP, sempre dispostos a ajudar nos momentos que precisamos.

Muito obrigado!

Sumário

Resumo	v
Abstract	vi
Agradecimentos	vii
1 Introdução	1
1.1 Contextualização	2
1.2 Organização do Texto	2
2 O Problema de Planejamento e Agendamento de Operações em uma Rede de Oleodutos	4
2.1 Caracterização do PPAORO	5
2.2 Parâmetros e Constantes Gerais	5
2.2.1 Restrições nos Órgãos	6
2.2.2 Restrições dos Dutos	8
2.2.3 Dados Dinâmicos da Rede	10
2.3 Solução e Objetivos	11
2.4 Modelo da Rede de Dutos	11
2.5 Topologia Brasileira	12
3 Revisão Bibliográfica	14
4 Arquitetura do Modelo de Resolução	20
4.1 Fase de Planejamento	21
4.2 Fase de Agendamento	22
5 Conceitos Básicos de Modelagem	24
5.1 Programação Lógica por Restrições	24
5.1.1 Definindo um modelo para satisfação de restrições	25

5.1.2	Encontrando uma solução	25
5.1.3	Melhorias na modelagem	26
5.2	Programação Linear Mista	27
5.2.1	Fluxo em Redes	28
6	Um modelo para o Problema de Agendamento	32
	Prólogo	32
1	Introduction	34
2	Problem Definition	37
2.1	Tank Restrictions	37
2.2	Pipeline Restrictions	38
2.3	Depot Restrictions	40
2.4	Inventory Constraints	41
2.5	Definition of a Solution	42
3	Related Work	43
4	Methodology	46
4.1	Planning and Routing	47
4.2	Scheduling Phase	51
4.3	The Sequencing Sub-problem	52
4.4	The Scheduling Sub-problem	68
4.5	A Running Example	70
5	Results	73
6	Conclusions	74
	Bibliografia	77
	Epílogo	80
7	Um modelo para o Problema de Planejamento	83
	Prólogo	83
1	Introduction	85
2	Problem Definition	87
2.1	Tank Restrictions	89
2.2	Pipeline Restrictions	89
2.3	Inventory Constraints	90
2.4	Input Instances and Solutions	90
3	A Network Flow Model for the Planning Phase	92
3.1	Network Model Definition	92
3.2	A Linear Programming Model	97
3.3	Network Flow Decomposition	101

4	Computational Results	103
5	Conclusions and Future Work	104
	Bibliografia	107
	Epílogo	110
8	Conclusões	112
	Bibliografia	114

Lista de Tabelas

3.1	Comparação entre abordagens anteriores.	15
6	Um modelo para o Problema de Agendamento	32
1	Solution Example	43
2	Comparison of the main approaches.	43
3	Solver and model statistics	75
7	Um modelo para o Problema de Planejamento	83
1	Initial values and constants for an input instance	91
2	Node description for the Pipeline Network flow Model	93
3	Execution Results	103
4	Manual planning vs Real planning - Total Network Flow Proportion	103
5	Manual planning vs Real planning - Total Network Flow Proportion by Product	103
6	Manual planning vs Real planning - Total Network Flow Proportion by Pipeline	104

Lista de Figuras

2.1	Modelo esquemático de uma rede de dutos.	12
2.2	Representação do sistema de dutos da região Sudeste de São Paulo (figura fornecida pela PETROBRAS).	13
6	Um modelo para o Problema de Agendamento	32
1	PETROBRAS pipeline network.	36
2	A pipeline network example.	38
3	Example of a flow reversal.	40
4	Solver Framework.	48
5	Activities for each pipeline in a delivery order's route.	55
6	Connection between two consecutive pipelines in an order's route.	56
7	Case when the <i>send</i> and <i>receive</i> activity sequences are different.	59
8	Example of edges in the pushing graph for a single pipeline.	62
9	Example of activity unification.	69
10	A small example.	73
11	Tankage Evolution for two instances.	74
7	Um modelo para o Problema de Planejamento	83
1	A Sample Pipeline Network.	86
2	The inland pipeline network.	88
3	Nodes and arcs representing pairs of depots d and products p . Items within parenthesis index the nodes enclosed inside the rectangle. The occluded rectangles represent the multiplicity of that network with regard to the inner index. .	95
4	Nodes and arcs for the stock of a pipeline with contents (p, qty, r) . Items within parenthesis index the nodes enclosed inside the rectangle. The occluded rectangles represent the multiplicity of that network with regard to the inner index. .	96

5	Nodes and arcs representing volumes traveling along routes. For a fixed route r and product p , at every period ti a new subnetwork is created so that the quantity will only be delivered after enough periods have passed. The subnetwork for period $t = 3$ was omitted to show more of the subnetwork for period $t = 2$. Items within parenthesis index the nodes enclosed inside the rectangle. The occluded rectangles represent the multiplicity of that network with regard to the inner index.	97
---	---	----

Capítulo 1

Introdução

A indústria de petróleo brasileira possui bases, também chamadas de órgãos, distribuidoras de derivados de petróleo, espalhadas por todo o país. Essas bases devem atender às demandas regionais. Entretanto, como grande parte da produção de petróleo e seus derivados é localizada, há necessidade de transporte dos produtos entre os órgãos. Dentre os meios de transporte disponíveis, o menos custoso e mais aceitável ambientalmente é aquele realizado usando redes de oleodutos. Nos Estados Unidos, por exemplo, as entregas por redes de oleodutos representam mais que 17% do volume total transportado, porém, menos de 2% do custo logístico [51].

As redes de oleodutos, para serem realmente efetivas, necessitam de controle e planejamento constante. Dois dos grandes problemas enfrentados, os quais são tratados nesta dissertação, são o planejamento mensal da operação dos dutos e o detalhamento diário da movimentação dos produtos pelos dutos. Na resolução de ambos é necessário respeitar restrições operacionais sobre as atividades nos órgãos e nos dutos. Exemplos de restrições estão nas capacidades dos tanques, taxas de vazões diferenciadas por produto, duto e sentido de fluxo, além de níveis de estoque.

Atualmente, o planejamento mensal e agendamento diário das operações são feitos manualmente. Inicialmente, em uma reunião entre os órgãos distribuidores, refinarias e os engenheiros de distribuição são discutidas as movimentações para atender as demandas e planejar as produções de forma a atingir um consenso. Isso formará o planejamento mensal das movimentações que será o alvo do agendamento diário. Assim, durante o mês planejado cada órgão propõe as operações necessárias para escoar sua produção e atender sua própria demanda. Em seguida, essas propostas são reunidas, e verifica-se a viabilidade de combiná-las. Quase sempre aparecem muitos conflitos, forçando os órgãos a se comunicar para resolver os conflitos e fechar uma solução que seja adequada para todos. Dada a extrema dificuldade e morosidade deste procedimento, têm-se aceitado como adequada qualquer solução que atenda a demanda, sem preocupações com a redução

de custos, entre outros aspectos.

Enquanto o problema de planejamento mensal mantém-se sem abordagens computacionais, já é conhecido que a dificuldade do problema de agendamento diário está diretamente ligada ao fato dele ser classificado como NP-Difícil [33]. Neste âmbito, é comum a utilização de ferramentas computacionais para fornecer soluções melhores e mais rapidamente para o problema. Outros trabalhos já trataram problemas semelhantes e a realidade brasileira também já foi alvo de outras pesquisas, como será abordado no capítulo 3. No entanto, dados os inúmeros detalhes a serem considerados, nenhuma dessas soluções poderia ser adotada para operar redes complexas.

Essa dissertação apresentará uma formalização para o problema de planejamento mensal da operação da rede de oleodutos e também uma nova abordagem para o agendamento diário dessas operações. O objetivo é fornecer uma ferramenta útil às empresas, tratando o máximo de especificidades, sem abusar de simplificações.

1.1 Contextualização

O problema em questão foi apresentado pela PETROBRAS [42], que possui uma rede de oleodutos com cerca de 11.300 km de extensão, contendo 15 refinarias e 54 terminais. A PETROBRAS forneceu todos os dados necessários para a definição das instâncias usadas na realização de testes e validação dos algoritmos. Há também uma interface gráfica, desenvolvida pela PETROBRAS, para visualização e verificação de viabilidade de soluções.

O projeto rendeu duas dissertações, esta apresentada aqui e outra apresentada em 2008 por André Ciré [10]. Os trabalhos foram desenvolvidos em conjunto e totalizaram 3 artigos publicados e um submetido. O capítulo 6 comparará os artigos apresentados por André Ciré com alguns dos resultados apresentados nesta dissertação. Além disso, por ser um problema comum em ambas as dissertações, os capítulos 2, 3 e 4 são compartilhados e apresentados aqui com revisões e alterações pontuais.

1.2 Organização do Texto

O Problema de Planejamento e Agendamento de Operações em uma Rede de Oleodutos (**PPAORO**) é explicado detalhadamente no capítulo 2. Uma revisão bibliográfica do problema é descrita no capítulo 3. No capítulo 4 é proposta a decomposição do problema em duas partes, o planejamento mensal e o agendamento diário. O capítulo 5 dá uma pequena introdução às notações e metodologias necessárias para as modelagens propostas. Em seguida, o capítulo 6 contém o artigo publicado no periódico *Constraints* com o modelo em programação por restrições para resolução do problema de agendamento

diário. O artigo também discute uma abordagem heurística inicial para o problema de planejamento mensal. O capítulo 7 expõe o artigo submetido ao periódico *Computer & Chemical Engineering* com o modelo em fluxo em redes para resolução do problema de planejamento mensal. Por fim, o Capítulo 8 apresenta uma conclusão e propõe possíveis trabalhos futuros.

Capítulo 2

O Problema de Planejamento e Agendamento de Operações em uma Rede de Oleodutos

Empresas de petróleo dispõem usualmente de um conjunto de *refinarias* e *terminais*, denominados *órgãos*, os quais são dedicados à produção e distribuição de derivados de petróleo e bio-combustíveis aos seus mercados consumidores. As campanhas de produção das refinarias e de entrega aos mercados são traduzidas, respectivamente, em valores estimados de produção e demanda de produtos em cada órgão da rede [6, 9].

Contudo, os órgãos não são auto-suficientes no suprimento de seus mercados locais. Torna-se necessário, portanto, o uso de algum meio de distribuição para evitar a falta de produto nos terminais e para escoar a produção em excesso nas refinarias. Tal transporte é predominantemente feito por meio de uma *rede de dutos*, a alternativa mais vantajosa em termos econômicos, operacionais e ambientais [43].

Os dutos devem ter respeitadas suas capacidades, seus sentidos permitidos de fluxo e suas vazões de bombeamento. Um duto pode ser usado para movimentar diferentes tipos de produto consecutivamente. Note que alguns dutos permitem vazão em ambos os sentidos, com inversões de fluxo. Outra característica operacional é que os dutos devem estar sempre completamente preenchidos, ou seja, a extração de um volume implica que necessariamente outro volume deve ser bombeado simultaneamente.

Os produtos, por sua vez, estão associados a um conjunto de *restrições de interface*, isto é, apenas podem ser bombeados consecutivamente em um duto se forem *compatíveis* entre si, condição essencial para a manutenção de um nível de qualidade aceitável. Para os produtos que circulam nos dutos, cada órgão também apresenta restrições locais de bombeamento e recebimento de produtos, decorrentes das conexões internas entre tanques e dutos.

O *Problema de Planejamento e Agendamento de Operações em uma Rede de Oleodutos* (**PPAORO**), desta forma, consiste em determinar as operações a serem realizadas em uma rede de dutos a fim de escoar a produção e atender um dado conjunto de demandas, considerando um horizonte temporal pré-estabelecido. Esta programação deve satisfazer todas as restrições operacionais específicas dos órgãos e dutos, além de buscar reduzir os custos de transporte pelo sistema.

Este capítulo é compartilhado com a dissertação de André Ciré [10] e apresentado aqui com revisões e alterações pontuais.

2.1 Caracterização do PPAORO

A seguir, são descritos os dados e restrições que formalizam o **PPAORO**, necessários para definir as instâncias e suas soluções válidas. Tal caracterização pode ser dividida em quatro classes: *parâmetros e constantes gerais do problema*, *restrições dos órgãos*, *restrições dos dutos* e *dados dinâmicos da rede*. As três primeiras classes estão relacionadas à topologia física e pouco mutável da rede, enquanto a última varia conforme o horizonte de planejamento.

2.2 Parâmetros e Constantes Gerais

Há quatro parâmetros fundamentais para o **PPAORO**, conforme relação abaixo.

1. *Unidade de Tempo (u.t.)*: designa a unidade de tempo real que será utilizada para a discretização da instância e precisão dos modelos desenvolvidos. Nos cenários reais, é comum que seja utilizado *minuto* como a u.t. básica.
2. *Unidade de Volume (u.v.)*: analogamente à u.t., designa a unidade de volume real que será utilizada para as vazões e capacidades. Usualmente são *metros cúbicos*.
3. *Horizonte (H)*: um número inteiro que designa o horizonte de execução em unidades de tempo, isto é, o tamanho da janela de tempo da instância na qual o planejamento e agendamento devem ser aplicados. Considera-se também que o instante inicial de planejamento é 1. Por exemplo, dados $H = 14400$ e u.t. em minutos, têm-se um horizonte de 10 dias em tempo real.
4. *Conjunto de Produtos (P)*: o conjunto $P = \{p_1, p_2, \dots, p_{np}\}$ designa os produtos que são armazenados e transportados pela rede. Um *grupo de produtos* é um subconjunto de P formado por produtos com certas características químicas semelhantes, tais como os grupos de *gasolinas*, *dieseis* e *álcoois*. Algumas restrições, tal como as

de interface, podem ser definidas sobre grupos ao invés de produtos individuais, facilitando a representação do problema.

2.2.1 Restrições nos Órgãos

Uma rede de dutos possui um conjunto de órgãos $Org = \{o_1, o_2, \dots, o_{no}\}$, responsáveis pela produção, armazenamento e distribuição de produtos. Um número significativo das restrições de um determinado órgão $o \in Org$ está relacionado com seu complexo de tanques, dado pelo conjunto $T(o) = \{t_1, t_2, \dots, t_{nto}\}$. Tais restrições são descritas a seguir, com $Tq = \bigcup_{o \in Org} T(o)$.

- C1: Para um dado tanque $t \in Tq$, deve-se respeitar sua capacidade máxima de armazenamento $cap(t)$ e nunca violar sua capacidade mínima, que é zero.
- C2: Um tanque $t \in Tq$ pode armazenar apenas um tipo de produto $prod(t) \in P$ durante todo o horizonte de planejamento, condição imposta como um requisito de qualidade. Além disso, usualmente há mais de um tanque por produto em cada órgão, mas não necessariamente um órgão contém tanques para todos os produtos da rede. Se $T(o) = \emptyset$ para um dado $o \in Org$, o órgão é denominado *entroncamento*, isto é, utilizado apenas para a passagem de produtos.

No cenário real, há casos em que certos tanques podem ser utilizados para o armazenamento de produtos diferentes dos originalmente atribuídos, o que envolve a aplicação de um custo fixo relativo à limpeza e a outros detalhes operacionais. Como são casos indesejáveis e muitas vezes provenientes da dificuldade do planejamento manual, a restrição C2 é inclusive requisitada pelos operadores da PETROBRAS.

- C3: Um produto pode ser tanto bombeado para um tanque quanto retirado dele, desde que essas operações não sejam simultâneas. Isto significa que, ao iniciar uma inserção ou retirada em um tanque, toda a operação deve ser completada antes que qualquer outra seja realizada neste tanque.

Uma exceção à este caso ocorre em órgãos que representam terminais portuários. Devido ao período restrito em que os navios podem ficar parados nos portos, muitas vezes volumes de produtos devem ser enviados diretamente dos dutos para os navios sem pré-estocagem. Para tanto, realiza-se uma operação *pulmão*, onde o produto é simultaneamente bombeado nos dutos para um tanque, e do tanque para o navio com uma vazão menor. No entanto, a operação pulmão é apenas possível para alguns órgãos que possuem esta capacidade hidráulica, definido pelo conjunto $Pulm \subseteq Org$.

Além disso, como hipótese, um tanque que esvaziou pode ser reabastecido sem que seja necessário considerar um tempo para limpeza e preparo para o armazenamento de volume adicional.

- C4: Se o produto a ser inserido em um tanque é produzido localmente, isto é, no mesmo órgão, o tanque só pode receber este produto se estiver completamente vazio, condição imposta também por questões de qualidade.
- C5: Caso ocorra a mistura de dois volumes produzidos em órgãos distintos num certo tanque, a qualidade do produto deverá ser reverificada antes da entrega ao mercado consumidor ou bombeamento para outros órgãos. Isto é representado aqui como um *tempo de certificação*, T_c , considerado entre o instante final em que se der a mistura e o instante inicial das demais operações no tanque.

Uma vez definidas as condições de armazenamento de produto nos órgãos, é possível listar as restrições referentes às necessidades de estocagem impostas pelos mercados, além daquelas relacionadas aos limites de envio e recebimento.

- C6: O valor de *estoque* de um órgão $o \in Org$, produto $p \in P$ e instante i é dado somando-se o volume em i de todos os tanques em $T(o)$ que armazenam p . Em todos os órgãos e instantes, tal estoque deve respeitar limitantes de *estoque mínimo*, $est_min(o, p)$, e *estoque máximo*, $est_max(o, p)$. Esta restrição é proveniente da necessidade de se manter estoques de segurança para forçar o escoamento da produção através da rede e para eventuais problemas na distribuição, tal como manutenções emergenciais de dutos.
- C7: Devido às restrições sobre o conjunto de bombas e válvulas hidráulicas em um certo órgão $o \in Org$, deve ser observado um *número máximo de operações simultâneas de envio*, $env_max(o)$. Como são relativas aos bombeamentos, não há uma restrição análoga para recebimento de produtos em o .
- C8: *Alinhamentos proibidos*: o complexo de conexões internas aos órgãos, principalmente refinarias, não permite que determinadas operações em oleodutos ocorram simultaneamente. Cada configuração não-passível de uso para transporte de produtos é associada a um *alinhamento proibido*, um código que agrupa as operações proibidas de serem exercitadas simultaneamente no órgão.

Supondo que $Align(o)$ sejam os alinhamentos proibidos de $o \in Org$, cada alinhamento $alin \in Align(o)$ é formado por um conjunto de duplas (p, s) , onde $p \in P$ e s (sentido) indica se a operação é relativa ao envio ou recebimento. Por exemplo, o alinhamento $alin = \{(gasolina, envio), (diesel, recebimento)\}$ significa que as

operações de envio de gasolina e recebimento de diesel não podem ser simultâneas. Um certo alinhamento pode envolver diversos produtos e diferentes direções de fluxo. Além disso, como está relacionado às conexões internas de um órgão, não depende dos dutos de entrada ou saída de produto.

Essa forma de representação facilita a modelagem do **PPAORO**, evitando a necessidade de explicitar detalhes da topologia dos circuitos hidráulicos e dutos internos ao órgão.

- C9: Devido principalmente ao custo mais alto da energia elétrica em certos períodos do dia, um órgão deve obedecer a *períodos de sazonalidade*, momentos em que nenhuma operação pode ser iniciada. Para um dado órgão $o \in Org$, períodos de sazonalidades são dados como um conjunto de intervalos de tempo, $saz(o) = \{(inicio_1, fim_1), \dots, (inicio_{nsaz(o)}, fim_{nsaz(o)})\}$.
- C10: Operações também não podem ser nem iniciadas, nem terminadas, durante as *trocas de turno* de trabalho nos vários órgãos. Estas trocas são dadas de forma análoga aos períodos de sazonalidade, $tt(o) = \{(inicio_1, fim_1), \dots, (inicio_{tt(o)}, fim_{tt(o)})\}$, $o \in Org$.

2.2.2 Restrições dos Dutos

A rede está associada a um conjunto $Dt = \{d_1, d_2, \dots, d_{nd}\}$ de dutos, onde cada $d \in Dt$ conecta um par de órgãos $orgs(d) = (o_i, o_j)$, com $o_i, o_j \in Org$. No caso, o_i é denominado *órgão* ou *extremidade de origem* e o_j , *órgão* ou *extremidade de destino* do duto d . O sentido de fluxo da extremidade de origem à de destino corresponde ao *sentido principal* do duto, ou $p(d)$. Já o sentido de fluxo contrário corresponde ao *sentido reverso* do duto, ou $r(d)$. Outra característica é que dois órgãos podem estar conectados por mais de um duto, o que é usual nas instâncias reais. Volumes também podem ser transmitidos diretamente de um duto para outro, sem a necessidade de serem armazenados temporariamente em órgãos intermediários.

A partir dos órgãos e dutos, define-se também o conceito de *rotas*. Uma *rota* r é composta por uma seqüência de pares de órgãos separados por dutos, representando um percurso válido através da rede. As rotas possíveis são prefixadas e informadas como parâmetro: se um certo volume de produto v for transmitido de um órgão o_i da rede até um órgão o_j , passando por um ou mais órgãos e dutos intermediários na seqüência $d_1, o_{i+1}, \dots, o_{j-1}, d_n$, $j = i + n$, necessariamente a rota $r = (o_i, d_1, o_{i+1}, \dots, o_{j-1}, d_n, o_j)$ deve existir. Note que os volumes são transmitidos diretamente entre os dutos intermediários de uma rota, sem perdas ou armazenamento nas conexões intermediárias.

As restrições referentes aos dutos são listadas a seguir.

- C11: Os dutos são pressurizados e, portanto, devem sempre estar completamente preenchidos com produtos. A capacidade volumétrica de um duto $d \in Dt$ é definida por $vol(d)$.

Também é considerada uma conservação de massa ideal: para se retirar um determinado volume v de uma extremidade do duto, deve-se inserir o mesmo volume v em sua outra extremidade. Tal volume pode ser proveniente de um tanque ou mesmo de um outro duto, interligado ao mesmo órgão através de uma de suas extremidades.

- C12: As *vazões de bombeamento* em um duto são limitadas por produto, devido às suas viscosidades, e pelo sentido de fluxo, decorrente das diferenças na capacidade das bombas entre os órgãos de origem e destino e inclinação do terreno. Usualmente dutos em terrenos inclinados possuem uma vazão de descida maior que a de subida. Os limites inferior e superior das vazões para um duto $d \in Dt$, produto $p \in P$ e sentido $s \in \{p(d), r(d)\}$ são dados, respectivamente, por $vz_inf(d, p, s)$ e $vz_sup(d, p, s)$.

Assim, para se bombear um certo volume em um duto, deve-se considerar todos os produtos e sentidos dos dutos nas rotas que serão movimentadas como consequência do bombeamento. A vazão máxima de bombeamento será, portanto, o mínimo das vazões superiores destes produtos nos seus respectivos sentidos. Já a vazão mínima de bombeamento será dada pelo máximo das vazões inferiores.

Os parâmetros $vz_inf(d, p, s)$ e $vz_sup(d, p, s)$ também indicam os sentidos possíveis de fluxo para os produtos em um duto. Por exemplo, se um produto p só puder trafegar no sentido principal do duto, então $vz_inf(d, p, r(d)) = vz_sup(d, p, r(d)) = 0$.

- C13: Devido à questões operacionais das bombas de cada órgão, deve-se respeitar também uma *quantidade mínima de bombeamento* ao se injetar produtos nos dutos. O volume bombeado em um duto também pode ser denominado de *batelada*, e esta restrição também é comumente referenciada como *tamanho mínimo de batelada* na literatura. Tal quantidade é definida por duto $d \in Dt$, produto $p \in P$ e sentido $s \in \{p(d), r(d)\}$, dada por $qtde_min(d, p, s)$.

- C14: Dois produtos em contato num certo duto devem ser *compatíveis*. Pares incompatíveis são dados pelo conjunto de duplas $Incomp = \{(g_{i_1}, g_{j_1}), \dots, (g_{i_n}, g_{j_n})\}$, onde $g_k \subset P$ é um grupo de produtos.

Esta restrição está ligada à manutenção da qualidade dos produtos, de forma a evitar a degradação decorrente da mistura (ou *interface*) entre tipos químicos diferenciados. Caso seja absolutamente necessário enviar dois produtos incompatíveis em seqüência, deve-se interpor entre eles um terceiro produto, compatível com ambos, chamado de *selo*. O volume de selo depende do oleoduto, dos produtos em

questão e do sentido do fluxo, e é dado por $selo(d, p, q, s)$, para $d \in Dt$, $p, q \in P$ e $s \in \{p(d), r(d)\}$.

C15: Os dutos possuem *períodos de manutenção* específicos, momentos em que são realizados reparos e limpeza. Os períodos de manutenção são dados por $manut(d) = \{mnut_1, mnut_2, \dots, mnut_{nm(d)}\}$, $d \in Dt$. Cada elemento $mnut_i = (inicio, fim)$ indica que, durante os instantes *inicio* e *fim*, não poderão ocorrer movimentações de produtos em nenhum sentido do duto.

2.2.3 Dados Dinâmicos da Rede

Os *dados dinâmicos* são parâmetros que não se referem à topologia física da rede, mas são específicos da instância a ser tratada pelo algoritmo. Podem ser divididos em três grupos: dados relativos ao estado inicial da rede, relativos à produção e relativos à demanda.

Os dados do estado inicial da rede são dois:

1. *Estoque inicial dos tanques*, indicando o produto e o volume inicial nos tanques.
2. *Conteúdo inicial dos dutos*, dado para todos os dutos por uma seqüência da forma $DIni(d) = \{(p_m, q_m, r_m), \dots, (p_n, q_n, r_n)\}$, $d \in Dt$. A tripla (p_j, q_j, r_j) representa, respectivamente, um produto $p_j \in P$, sua quantidade q_j e a rota que deve seguir, r_j . Desta forma, considera-se que os produtos inicialmente nos dutos já possuem um destino pré-estabelecido, que obrigatoriamente deve ser respeitado pelas soluções geradas. A ordem das tuplas na seqüência $DIni(d)$ refere-se ao sentido principal de fluxo no duto.

As campanhas de produção para cada órgão $o \in Org$ são definidas pelo conjunto $Pr(o) = \{pr_1, pr_2, \dots, pr_{npr}\}$, onde cada elemento $pr_i \in Pr(o)$ é formado pela tupla $(p, v, inicio_pr, fim_pr)$. No caso, $p \in P$ é o produto que será refinado, v um inteiro não-negativo que representa o volume produzido e, por fim, $[inicio_pr, fim_pr]$ representa o intervalo de tempo em que a produção ocorrerá no órgão.

O volume v produzido é distribuído uniformemente entre as $t = fim_pr - inicio_pr + 1$ unidades de tempo, isto é, a cada instante serão produzidas $\lceil v/t \rceil$ unidades de volume (u.v.). A produção pode ser distribuída em diferentes tanques durante este intervalo, contanto que estejam vazios imediatamente antes de a receberem (restrição C4).

Já as demandas são dadas de forma simétrica às produções. Para o órgão $o \in Org$, é definido o conjunto $Dem(o) = \{dem_1, dem_2, \dots, dem_{ndem}\}$, onde cada elemento $dem_i \in Dem(o)$ é formado pela tupla $(p, v, inicio_dem, fim_dem)$. Têm-se que $p \in P$ é o produto da demanda, v é um inteiro não-negativo que representa o volume demandado e, por fim, $[inicio_dem, fim_dem]$ representa o intervalo de tempo em que a demanda ocorrerá no

órgão o . O volume v demandado é distribuído uniformemente entre as $t = fim_dem - inicio_dem + 1$ unidades de tempo, isto é, a cada instante serão extraídas $\lceil v/t \rceil$ unidades de volume (u.v.). Tal como a produção, demandas podem ser absorvidas de diferentes tanques durante este intervalo.

2.3 Solução e Objetivos

Uma solução é formada por uma programação de *movimentos* de entrada nos dutos. Cada movimento é designado pela tupla $m = (p, r, v, inicio_m, fim_m, t_s, t_e)$, isto é, por um produto $p \in P$, uma rota r cadastrada, o volume v positivo do produto, os instantes de início, $inicio_m$, e fim, fim_m , de bombeamento do produto no primeiro duto da rota e, por fim, pelos tanques de saída $t_s \in Tq$ e de entrada $t_e \in Tq$, de onde o volume será extraído e onde será armazenado, respectivamente. O conjunto de movimentos deve ser tal que respeite todas as restrições nos órgãos e dutos, além de satisfazer as campanhas de produção e demandas em cada órgão.

No modelo proposto aqui, o foco será dado à satisfação de restrições ao invés da otimização de uma certa função objetivo, devido à alta complexidade de se obter soluções viáveis para o **PPAORO**. Assim, custos relativos à utilização dos dutos e tanques são desconsiderados. Isto é condizente com a prática atual realizada pela PETROBRAS, na qual o correto atendimento da demanda é prioritário frente aos demais custos.

2.4 Modelo da Rede de Dutos

A figura 2.1 apresenta um modelo esquemático simples de uma rede de dutos. Cada órgão D_i apresenta seu próprio conjunto de tanques, e note que mais de um duto pode ser utilizado para conectar dois órgãos distintos, tal como ocorre entre D_2 e D_3 .

Para que dois produtos fiquem em contato no duto, como em P_2 , ambos devem ser compatíveis entre si, obedecendo a restrição C14. Se o órgão D_0 possuir um número máximo de envios simultâneos igual a 1 (restrição C7), então produtos não podem ser injetados simultaneamente em P_0 e P_1 a partir de D_0 . Além disso, caso existir um alinhamento proibido (restrição C8) de recebimento de produtos F_1 (cor cinza) e F_2 (cor escura) no órgão D_3 , ele não poderá receber produtos dos dutos P_3 e P_4 simultaneamente.

Por fim, um exemplo de rota seria $r = (D_1, P_2, D_2, P_4, D_3)$. Se o destino do produto F_1 (cor cinza) do duto P_2 for o órgão D_3 pela rota r , a vazão aplicada para empurrar o duto P_2 , no caso a partir de D_1 , deve ser consistente com as vazões dos produtos tanto de P_2 como do duto P_4 , que também será empurrado.

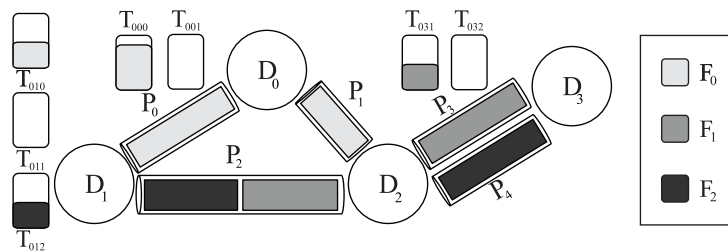


Figura 2.1: Modelo esquemático de uma rede de dutos.

2.5 Topologia Brasileira

A especificação do **PPAORO** aqui descrita agrega as principais restrições abordadas na literatura do problema, além de representar com certa fidelidade a realidade enfrentada pelos planejadores e operadores de dutos. Ela foi concebida a partir da análise bibliográfica do **PPAORO** e, principalmente, após diversas reuniões juntos aos gerentes da PETROBRAS.

A dissertação focará na rede de dutos da região Sudeste do Brasil, gerenciada pela PETROBRAS e representada na figura 2.2. Esta rede é composta por 14 órgãos, 29 dutos, cerca de 240 tanques e mais de 30 produtos. O horizonte de planejamento é de cerca de 10 dias.

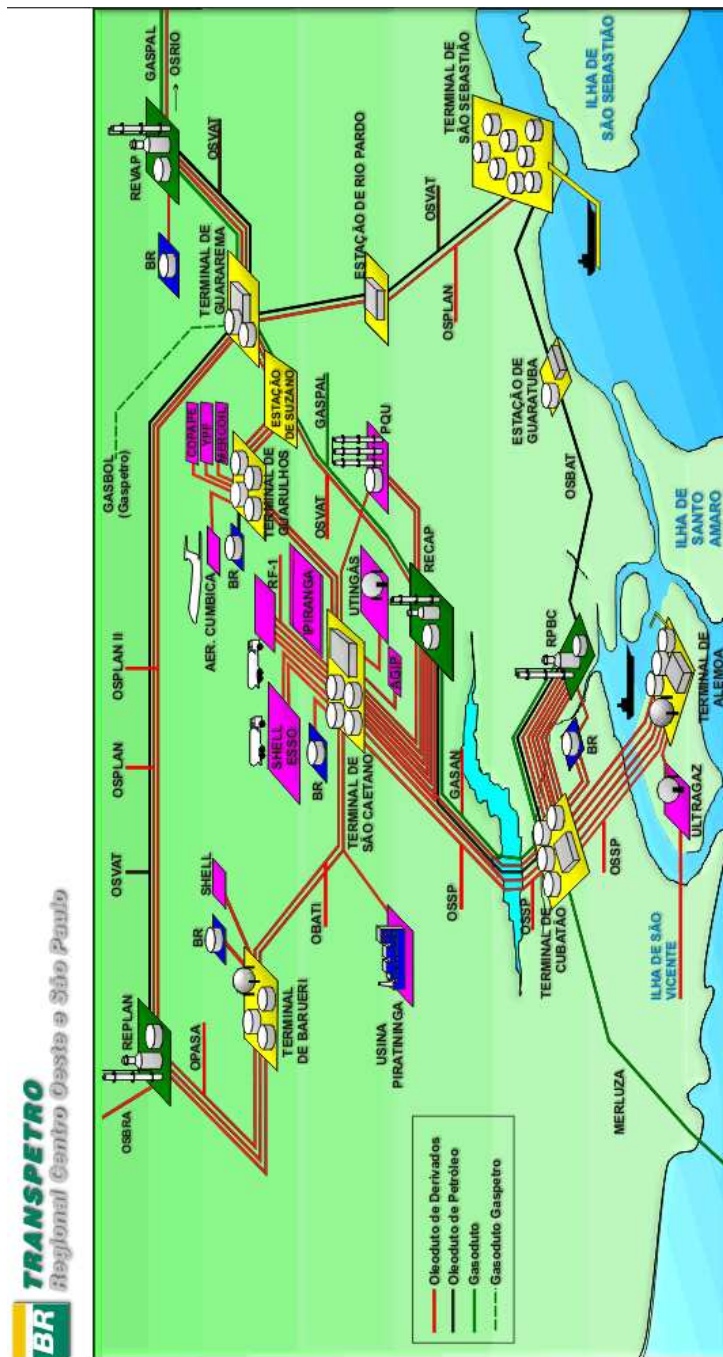


Figura 2.2: Representação do sistema de dutos da região Sudeste de São Paulo (figura fornecida pela PETROBRAS).

Capítulo 3

Revisão Bibliográfica

O problema abordado nesta dissertação engloba um conjunto de restrições bastante complexo e diversificado, relacionando fatores como capacidade de estocagem, sequenciamento de produtos nos dutos e condições de balanceamento de massa em sua especificação mais geral. O trabalho de [33] demonstra que basta considerar a restrição C14, de interfaces entre produtos, para que o **PPAORO** seja classificado como *NP-Difícil* [40].

Na literatura, o problema é usualmente conhecido como *Multiproduct Pipeline Scheduling* e diversos trabalhos se propuseram a apresentar abordagens heurísticas ou exatas para sua resolução. Contudo, em decorrência da grande aplicabilidade prática do problema, os trabalhos existentes diferem bastante na forma como definem e adaptam restrições às suas respectivas realidades. Desta forma, três pontos são essenciais para a compreensão de cada proposta de modelagem: a topologia da rede, o método de resolução aplicado e, por fim, os resultados computacionais e a operacionalidade das soluções.

Há pesquisas dedicadas à revisão bibliográfica do **PPAORO**, tais como [17, 50], que apresentam uma análise bastante detalhada das abordagens existentes na área. Assim, a revisão aqui descrita procurará destacar as propostas mais relevantes e que serviram como base para o desenvolvimento da abordagem aqui discutida. A tabela 3.1 sintetiza as características de tais propostas e a compara com este trabalho.

Este capítulo é compartilhado com a dissertação de André Ciré [10] e apresentado aqui com revisões e alterações pontuais.

Dentre tais trabalhos, o estudo mais completo da área é realizado por Camponogara [9], cujas características mais se aproximam da realidade quando comparadas aos demais trabalhos. Nele, define-se um *Problema de Transporte de Derivados de Petróleo Simplificado*, ou PTDPS, no qual as vazões são fixas por duto, todas as bases são capazes de armazenar todos os produtos (eliminando-se, assim, a transmissão de produtos duto-duto), e não há trocas de turno ou manutenções programadas. Também desconsideram-se estoques mínimos por base e quantidades mínimas por operação. Os tanques, por sua vez,

	[3, 2]	[7, 20, 44, 45]	[8, 35, 6]	[16, 15, 23]	[30]	[46, 48, 47]	Proposto
Órgãos	8	5	12	7	2	2	14
Dutos	8	4	28	7	1	1	29
Tanques	40	25	84	28	16	12	242
Produtos	5	4	7	4	8	6	32
Horizonte	14 dias	3 dias	5 dias	4 dias	5 dias	30 dias	10 dias
Tempo contínuo		✓	✓				✓
Vazão variável		✓			✓	✓	✓
Tanques individuais					✓		✓
Rede Genérica	✓		✓	✓			✓
Incompatibilidades		✓	✓		✓	✓	✓
Técnicas	GA	PLM,VNS, PLM,PLI	A-Teams, GRASP,PLM	AE+PLM	PLM+PR	PLM	PLM+PR
Soluções	Viável	Ótimo	Inviável Viável,Viável	Viável	Ótimo	Ótimo	Viável

Tabela 3.1: Comparação entre abordagens anteriores.

são *agregados*, isto é, há um tanque por órgão com capacidade equivalente à soma de todos os tanques que armazenam o produto em questão.

Para a resolução do PTDPs, o trabalho de [9] propôs duas técnicas distintas, uma exata e outra heurística. A primeira, modelada como uma Programação Linear Inteira (PLI) [52], buscava reduzir o PTDPs a um problema de *Fluxo em Redes Multi-períodos* [5], com dutos e órgãos como vértices e transmissões de quantidade como arestas. Apesar da existência de algoritmos com bom desempenho para problemas deste tipo, verificou-se que a resolução do modelo PLI do PTDPs era impraticável, como consequência do tamanho e densidade do grafo resultante da topologia da rede. Para as últimas versões dos resolvidores comerciais de PLI até a publicação desta dissertação, o tempo médio para se obter uma solução para a *relaxação linear* do modelo foi de 50 horas ¹.

A segunda técnica proposta em [9] caracterizava-se pela aplicação de *Times Assíncronos* ao PTDPs, heurística na qual diversos agentes autônomos tratam de partes distintas do problema, trocando de tempos em tempos informações relevantes para a sua resolução. Para viabilizar a aplicação da heurística, o problema foi modelado como um *Job Shop Scheduling* [9], composto por dois agentes: um para a geração de bombeamentos de quantidades entre bases e dutos, os *jobs*, e outro para o agendamento e simulação de tais bombeamentos nos dutos, equivalente ao processamento dos *jobs* em máquinas (dutos).

Apesar de mais eficiente, a aplicação de Times Assíncronos proposta por [9] não foi capaz de gerar soluções viáveis para as instâncias apresentadas, em decorrência das estratégias essencialmente gulosas atribuídas aos agentes. Com esta perspectiva, Laber [35] utilizaram os Times Assíncronos como a fase gulosa de uma metaheurística *GRASP* [49]

¹Usando a ferramenta **CPLEX 11**, em um computador *Pentium D - 3.4Ghz* com *2 GB RAM*

para o PTDPS, desenvolvendo uma busca local para o reparo das soluções. Como consequência, foram capazes de satisfazer todas as restrições consideradas no PTDPS, mas o tempo necessário para a ressimulação após cada aplicação da busca local inviabilizava sua utilização.

Posteriormente, Braconi [6] relaxou o modelo PLI de Fluxo em Redes Multiperíodos de [9], denominando a resolução deste novo modelo *Etapa de Planejamento*. Esta solução parcial era então utilizada como parte de uma segunda heurística, a *Etapa de Agendamento*, na qual atribuíam-se tempos aos fluxos criados na primeira etapa. Esta técnica permitiu obter soluções rapidamente, mas as relaxações intrínsecas à definição do PT-DPS, tal como tanques para todos os produtos em todos os órgãos, tornavam as soluções encontradas pouco aplicáveis na prática.

Outro trabalho fundamentado na aplicação de heurísticas é de Crane [13], no qual um Algoritmo Evolutivo [22] foi aplicado para uma versão bastante simplificada do **PPAORO**. Nela, a rede de dutos é tratada como uma árvore direcionada com 8 nós, e os estoques dos tanques possuem apenas três estados possíveis: alto, médio e baixo. Restrições de interface e tamanho mínimo de bateladas são também desconsideradas. O algoritmo proposto foi capaz de gerar soluções para horizontes de até 3 dias mas, devido à forma como as soluções eram representadas, o algoritmo era capaz de lidar apenas com instâncias pequenas com poucos tanques e órgãos.

A técnica desenvolvida por De La Cruz [15] também aplica um Algoritmo Evolutivo para uma versão simplificada do **PPAORO**, sem restrições de interface e com todos os dutos possuindo um mesmo diâmetro e vazão. No entanto, são considerados dutos bidirecionais e uma função multi-objetivo, buscando reduzir interfaces e prazo de entrega dos produtos. Adicionalmente em [16], o autor também implementa um modelo PLI para o problema, usado para gerar parte das soluções que compõe a população do Algoritmo Evolutivo. Em todos os casos, o algoritmo foi capaz de gerar soluções para instâncias pequenas rapidamente, especialmente com o uso de resolvidores PLI e como decorrência das simplificações.

Há ainda os trabalhos de Alves [3, 2], que também aplica um Algoritmo Evolutivo para uma variação do **PPAORO**. No entanto, o procedimento desenvolvido foca na utilização de uma rede específica da PETROBRAS denominada *rede de escuros*, utilizada para o tráfego de produtos de grande viscosidade. O problema é similar ao PTDPS de [9], com tanques agregados e vazões constantes, mas considerando também tamanho das bateladas e uso de selos para separar produtos incompatíveis. Além disso, a função objetivo buscava minimizar o não-atendimento da demanda. O algoritmo obteve bons resultados para um horizontes de 7 e 14 dias, discretizados em períodos de algumas horas.

Uma abordagem alternativa é descrita em Sasikumar [28], onde uma técnica baseada em Inteligência Artificial, denominada *Beam Search*, é agregada a heurísticas fortemente

baseadas na experiência humana. A técnica é aplicada em uma rede indiana, com restrições de sequenciamento de produtos e estoques nos órgãos. Apesar das simplificações, são geradas boas programações mensais para as instâncias em questão.

Ainda, os trabalhos de Liporace [19] e Milidiú e Liporace [34] tratam o **PPAORO** como um problema de *Planejamento em Inteligência Artificial* (PIA). Tal modelagem consiste em formular o problema genericamente como um conjunto de proposições lógicas, cuja composição de seus valores binários representam um *estado*. O planejador irá identificar uma sequência de ações válidas que leve o sistema do estado inicial ao estado-objetivo. Além de um planejador específico para o problema, a especificação do **PPAORO** como um PIA por Liporace foi incluída no *benchmark* oficial da *International Planning Competition* [11], uma competição internacional de planejadores genéricos. Contudo, numa perspectiva prática, os planejadores existentes resolvem apenas instâncias muito limitadas, com horizontes curtos, poucos tanques e poucas restrições.

Outro foco de pesquisa está em topologias simples, compostas por apenas um duto e um conjunto de órgãos conectados a ele. A motivação é que ela já representa diversos casos práticos existentes no mundo, incluindo sub-redes da própria PETROBRAS. Além disso, esta topologia permite considerar restrições mais complicadas referentes às vazões, inventário, perdas de volume decorrentes da criação de interfaces e custos de energia.

Neste contexto, encontram-se os trabalhos de Rejowski e Pinto [26, 32]. Ambos apresentam uma formulação PLI de tempo discreto para o sistema OSBRA da PETROBRAS, composto por 5 órgãos, 4 produtos e um duto que liga o estado de São Paulo à Brasília. São consideradas diversas restrições, tais como inventário nos órgãos e sequências permitidas de bombeamento, além de uma função objetivo referente à minimização de custos de estoque e de interface. Para um horizonte de 3 dias, os autores não obtiveram a solução ótima nos testes realizados. Contudo, em [44], o modelo é fortalecido pela introdução de cortes e se consegue, em um tempo computacional razoável, soluções ótimas para todas as instâncias anteriores.

Para a mesma especificação proposta por [44], Cafaro e Cerdá [7] apresentam uma formulação baseada em uma representação contínua do tempo e dos volumes. Conforme explicado pelos autores, esta representação permite uma diminuição significativa do tamanho da formulação e do número de variáveis binárias. Os resultados são comparados ao trabalho de [44], e mostram uma redução de cerca de três ordens de magnitude dos requisitos computacionais necessários para resolver o problema. A representação por tempos contínuos também foi implementada por Rejowski e Pinto [45], considerando adicionalmente vazões variáveis de bombeamento e decisões relativas ao controle de inventário. Para tanto, o problema foi modelado como uma Programação Não-linear Inteira, baseado na formulação em [32]. O algoritmo resultante gerou boas soluções para poucos dias.

Um outro direcionamento consiste em decompor o **PPAORO** em subproblemas mais

fáceis de serem tratados, ao invés de resolvê-lo completamente com um modelo único. O trabalho de Magatão [29] propõe uma divisão em que três submodelos são processados sequencialmente. O modelo lida com uma rede composta por um duto conectando uma refinaria a um terminal portuário. A execução inicia-se com um procedimento denominado *Tanque Bound*, caracterizado como um modelo em PLI responsável pela determinação dos recursos (*i.e.*, tanques) a serem utilizados em todo o agendamento. Em seguida, a *Rotina Auxiliar*, basicamente heurística, define alguns parâmetros de entrada para o próximo submodelo, como limites temporais que devem ser respeitados pelas tarefas no oleoduto. Por fim, o *Modelo Principal*, formulado como PLI e baseado nos parâmetros fornecidos pelo Tanque Bound e pela Rotina Auxiliar, determina o sequenciamento e a temporização das atividades de bombeamento no duto. A técnica foi testada com sucesso na topologia em questão, considerando restrições de tancagem e de interface, além de vazão constantes. Os autores propõem também em [30] uma abordagem integrada entre PLI e Programação por Restrições para resolver as mesmas instâncias, gerando boas soluções.

Mais recentemente, um conjunto de trabalhos de Relvas apresenta um estudo sobre uma rede de dutos em Portugal, formada por um duto e dois órgãos. Em [46, 48], os autores apresentam uma formulação PLI que considera diversas restrições de controle de inventário, baseada em [7]. Já em [47], o modelo anterior é melhorado de forma que sejam consideradas paradas em dutos e vazões variáveis, além do conceito de *re-agendamento*, em que o modelo é resolvido após perturbações na solução corrente.

Uma classificação importante para as instâncias encontradas na literatura é a topologia da rede de oleodutos. A tabela 3.1, apresenta algumas simples, com somente um duto ou com mais de um duto, porém, com somente uma rota. Esse segundo caso, é geralmente formado por único duto segmentado onde ocorrem bases de entrega. Há ainda alguns trabalhos [36, 28] que permitem o tratamento de redes em forma de árvore. Uma análise profunda desses trabalhos mostra que a topologia da rede é base das modelagens, o que limita uma possível extensão a outras instâncias.

Pode-se notar também da tabela 3.1 que essa dissertação tratará um número muito maior de restrições, além de uma rede completa, para fornecer uma solução útil aos operadores da rede de oleodutos. Por outro lado, não existe uma técnica padrão para o problema de agendamento em dutos. Assim, fez-se necessário um estudo de diversas técnicas de otimização para determinar, dada a realidade tratada, qual a mais promissora.

Nos capítulos 4 e 5 serão discutidas as técnicas de otimização escolhidas, os critérios de escolha e a aplicação delas ao problema. Essas discussões são compartilhadas com a dissertação de André Ciré [10].

A literatura estudada foca o problema de agendamento diário. Em geral, as soluções especificam exatamente os horários de movimentação e tratam um conjunto grande de restrições. Como será visto no capítulo 4, o planejamento mensal não necessita desse

detalhamento. Entretanto, o problema ainda precisa considerar toda a rede e um horizonte de planejamento de um mês pelo menos. Como pode ser visto na tabela 3.1, as abordagens balanceiam o uso de uma rede com grande número de dutos contra um horizonte mais extenso. Essa dissertação mostrará duas técnicas para resolução desse problema, uma heurística e outra utilizando modelagem por fluxo em redes, as quais tratarão o horizonte mensal com um número satisfatório de restrições.

Capítulo 4

Arquitetura do Modelo de Resolução

O estudo dos artigos e teses apresentados no capítulo 3 evidencia que, como decorrência da grande dificuldade do **PPAORO**, as abordagens existentes se apóiam fortemente em simplificações dos requisitos do problema. Por exemplo, são tratadas topologias de rede bastante simples na maioria dos casos, com apenas um duto de sentido único e terminais dispostos de forma sequencial [7, 28, 45]. Ou, ainda, não são consideradas incompatibilidade entre produtos [3, 16], ou os tanques são representados com capacidade agregada [6, 9].

Tais hipóteses permitem modelagens mais intuitivas e diretas do **PPAORO**, fundamentando seu tratamento pelas técnicas de otimização tradicionais, como mostradas no capítulo anterior. Em contrapartida, impossibilitam a representação de grande parte das restrições mais importantes para a viabilidade prática das soluções, o que torna as abordagens existentes muito pouco aplicáveis para os cenários reais. Um exemplo é quando se considera um tanque único de capacidade agregada. Apesar de simplificar as decisões relativas ao controle de inventário, inviabiliza a aplicação de restrições como não-simultaneidade (C3), essencial para um cálculo mais realista dos tempos de bombeamento. Note também que a maioria dos trabalhos existentes considera cerca de 6 órgãos, 4 produtos e poucos tanques. Por outro lado, a instância a ser tratada aqui agrega 14 órgãos com 242 tanques, além de 29 dutos, 32 produtos, u.t. em minutos e diversas outras restrições.

É clara, portanto, a necessidade de modelos mais abrangentes para a obtenção de soluções factíveis para apresentar aos operadores da rede de dutos. Contudo, constata-se que a resolução do **PPAORO** como um grande problema único e integrado é pouco promissora. Além do alto número de restrições complicadoras que devem ser satisfeitas simultaneamente, muitos dos trabalhos baseados em resoluções integradas não obtiveram resultados satisfatórios para uma topologia complexa, mesmo após diversas relaxações do modelo [9].

Desta forma, propõe-se uma arquitetura de resolução do problema composta por duas fases, executadas sequencialmente: a fase de *planejamento* e a fase de *agendamento*. Tal arquitetura é baseada nas decomposições recorrentes do **PPAORO** encontradas na literatura, as quais refletem os procedimentos manuais atualmente adotados. Cada fase é descrita em detalhes nas seções seguintes.

Este capítulo é compartilhado com a dissertação de André Ciré [10] e apresentado aqui com revisões e alterações pontuais.

4.1 Fase de Planejamento

O problema de *planejamento* trata da movimentação geral na rede de dutos, onde são programados quais órgãos irão receber e enviar produtos para o atendimento das demandas e escoamento das produções. Uma vez fixados os órgãos de destino e de origem, define-se então quais serão os dutos e as bases intermediárias pelos quais os produtos irão trafegar, determinando também os volumes que serão transmitidos nas rotas escolhidas.

As movimentações de produto geradas pela fase de planejamento são representadas por uma estrutura denominada *plano de entrega*, que representa uma transmissão de volume de um órgão origem a um órgão destino, passando por uma determinada rota. Cada plano contém um *prazo*, isto é, o instante máximo em que todo o volume já deve estar armazenado no órgão destino. Um plano de entrega é definido por uma tupla

$$pe = (t_i, o_i, t_d, o_d, p, v, r, pz),$$

onde $t_i \in T(o_i)$ é o tanque no órgão de origem $o_i \in Org$, $t_d \in T(o_d)$ é o tanque no órgão de destino $o_d \in Org$, v é o volume do produto $p \in P$, r é a rota de transmissão do plano e pz é o prazo do plano dentro do horizonte de programação. Considera-se que um plano de entrega é *indivisível*, ou seja, seu volume não pode ser fragmentado em dois ou mais planos. Contudo, um plano pode *parar* no duto, isto é, ser injetado de forma intermitente, o que acarreta também a parada de toda a rota envolvida no bombeamento. Operações de parada são essenciais, por exemplo, para satisfazer o número máximo de envios simultâneos (restrição C7) e os alinhamentos proibidos (restrição C8). Além disso, também são necessárias quando se necessita aguardar esvaziamento de tanques para que possam receber quantidades proveniente de dutos.

Assim, dadas as campanhas de produção e demanda em cada órgão, o objetivo da fase de planejamento é gerar um conjunto de planos de entrega de tal forma que, se todos forem *satisfeitos*, *i.e.*, entregues no prazo, garante-se uma solução que atenda todas as demandas da rede e escoe os volumes em excesso das produções.

Diferentes métodos heurísticos ou exatos podem ser aplicados para a criação dos planos para uma certa instância, os quais considerariam critérios específicos para a determinação das rotas, volumes e tanques. As técnicas heurísticas aplicadas neste trabalho baseiam-se em escolher os órgãos com as demandas mais *críticas*, isto é, com maior volume e mais próximas do início do horizonte. Uma vez escolhidos, definem-se então os órgãos que vão prover volume para satisfazê-los, utilizando como critério aqueles mais *próximos*, isto é, conectados por um menor número de dutos. Por fim, são selecionadas as rotas de maior vazão e os volumes para, então, fixar os demais parâmetros dos planos.

Além disso, visto que planos são também uma representação intuitiva do planejamento da rede, operadores dos dutos também são capazes de inserir novos planos ou alterar os existentes, muitas vezes para a representação de preferências de movimentação não previstos pelo algoritmo.

A fase de planejamento encapsula as decisões de quais rotas e volumes utilizar para a resolução de uma instância do **PPAORO**, mas não detalha as operações de bombeamento e seus respectivos tempos. O desafio da fase de planejamento é, portanto, criar conjuntos de planos com uma grande probabilidade de serem viáveis, sem contudo aumentar a complexidade do problema a ponto de torná-lo não-resolvível.

4.2 Fase de Agendamento

Uma vez definidos os órgãos que irão enviar e receber produtos e as respectivas rotas por onde os volumes irão trafegar, deve-se resolver o problema de *agendamento*, ou seja, determinar o número de operações de envios por plano de entrega e, principalmente, a ordem em que os bombeamentos serão realizados. Nesta fase, consideram-se as restrições de incompatibilidade e de capacidade dos tanques, além do cálculo de limitantes de tempo para envio e chegada dos volumes nos órgãos. Por fim, é necessário atribuir os exatos momentos em que ocorrerão as operações de bombeamento, compondo a solução final do problema. Isso deve ser feito de modo a garantir que sejam satisfeitas as restrições referentes às vazões de bombeamento, à troca de turno, à manutenção de dutos, e também outras restrições similares.

Dado um conjunto de planos de entrega $PE = \{pe_1, pe_2, \dots, pe_{npl}\}$, o objetivo da fase de agendamento é gerar movimentos $m_1, \dots, m_{nm}$, conforme definidos na seção 2.3, de forma que todos os planos sejam *cobertos* no prazo. Isto equivale a sequenciar e agendar os planos nas suas respectivas rotas e tanques, garantindo todos requisitos do problema relacionadas no capítulo 2, como restrição de interface, capacidade de tanques e não-simultaneidade de operações. Note que $nm \geq npl$, já que planos podem ser bombeados por uma ou mais operações de envio.

Apesar do grande número de restrições, a pré-atribuição de rotas da fase de planeja-

mento abre a possibilidade de modelagens mais intuitivas e compactas do problema de agendamento, já que as decisões ficam concentradas em como ordenar e temporizar os planos nos seus respectivos dutos.

Como consequência da dificuldade em garantir heurísticamente soluções operacionalmente viáveis, será utilizado um **modelo exato** como módulo resolvidor desta fase. Esse modelo, por sua vez, será implementado usando *Programação por Restrições* (PR) [31], sendo essa escolha motivada principalmente por duas razões. A primeira motivação decorre de restrições não-lineares inerentes ao sequenciamento e às vazões variáveis, impossibilitando a modelagem do problema usando Programação Linear. Já a segunda razão vem do fato de focarmos, prioritariamente, a obtenção de soluções viáveis ao invés de ótimas. As técnicas de PR têm obtido grande sucesso neste quesito. Mais detalhes sobre Programação por Restrição são apresentados no capítulo 5.

Capítulo 5

Conceitos Básicos de Modelagem

Na área de pesquisa operacional, surgem, a cada dia, novas aplicações de otimização. Muitas delas tratam de problemas NP-Difíceis, para os quais não se conhecem algoritmos polinomiais para sua resolução. Nesse âmbito, é comum a modelagem através de Programação Linear Mista [52] e de Programação por Restrições [31]. Essas técnicas possuem sólido embasamento teórico e garantidamente encontram soluções ótimas, se essas existirem.

Cada uma das técnicas abordadas nesta dissertação será descrita nas respectivas seções, e no decorrer serão discutidas algumas formas de hibridização entre elas.

5.1 Programação Lógica por Restrições

A Programação (Lógica) por Restrições (PR) [31, 4] é uma ferramenta muito poderosa para a modelagem de problemas combinatórios, principalmente na resolução de problemas de planejamento e agendamento. Com ela é possível criar um modelo usando variáveis e relações restritivas entre as variáveis. Durante a execução do modelo são atribuídos valores às variáveis de forma a satisfazer as restrições definidas. Também é possível especificar que os valores atribuídos sejam tais que minimizem uma função objetivo, possibilitando a modelagem de problemas específicos de otimização.

Sendo a PR uma linguagem bem abrangente, podem ser usados diversos tipos de restrições e, para cada tipo, existem formas diferentes de tratamento. Nesse ponto, a programação por restrições se divide em dois ramos, a *satisfação de restrições* e a *resolução de restrições*. Enquanto na *satisfação de restrições* as variáveis do modelo têm domínios finitos, a *resolução de restrições* trabalha sobre domínios infinitos ou complexos.

A *satisfação de restrições* é a mais utilizada em aplicações industriais e de otimização. Além disso, já foi utilizada anteriormente para resolução de problemas semelhantes ao proposto [30], com sucesso. Por estes motivos, ela será uma das técnicas principais usadas

nesta dissertação.

5.1.1 Definindo um modelo para satisfação de restrições

Ao se modelar um problema nesse ramo da programação por restrições, são necessários três passos:

- Definir um conjunto de variáveis $X = \{x_1, x_2, \dots, x_k\}$ e os respectivos domínios $D_i, i = 1, 2, \dots, k$, de cada variável. Esses domínios não são necessariamente intervalos inteiros e, às vezes, nem mesmo são numéricos.
- Definir o problema como restrições na forma $C_1 \wedge C_2 \wedge \dots \wedge C_n$, onde cada C_i representa uma restrição sobre as variáveis de X .
- Realizar a busca por soluções. Cada solução, por sua vez, é definida por uma valoração $V = \langle x_1 = v_1, x_2 = v_2, \dots, x_k = v_k \rangle$. A valoração V será válida se respeitar todo os domínios e as restrições impostas sobre as variáveis.

Um *objetivo* $G = \langle x_1 = g_1, x_2 = g_2, \dots, x_k = g_k \rangle$ é uma atribuição inicial de valores às variáveis de X . A atribuição pode ser irrelevante (“don’t care”) para algumas das variáveis, deixando-as livres sobre seu domínios D_i . Ou a atribuição pode fixar o valor de x_i em g_i . Para exemplificar, dada a restrição $x - y = z$, e os domínios de x, y, z como $[1 \dots 3]$, pode-se ter como *objetivo* $G = \langle x = 1, y, z \rangle$, sendo x fixado e y, z livres. A possibilidade de se fixar vários objetivos é uma das grandes vantagens da programação por restrições. Em outros paradigmas de programação, seriam necessárias implementações específicas para cada objetivo.

5.1.2 Encontrando uma solução

A maioria dos problemas NP-Difíceis é tratada na forma de *satisfação de restrições*, e a busca pela solução é feita de forma combinatória. Mais especificamente com uma estratégia de *backtracking*.

Sendo o *backtracking* sabidamente ineficiente, a PR fornece técnicas para redução do esforço de *backtracking*. Um exemplo disso é a diminuição dos domínios das variáveis. Um modo de diminuir os domínios, e conseqüentemente o espaço de soluções, é refletir as restrições sobre eles, o que é chamado de *propagação de restrições*. Para visualizar isso, retoma-se o exemplo de restrição anterior, e vê-se que os domínios viáveis podem ser reduzidos para: $x \in \{2, 3\}$, $y \in \{1, 2\}$ e $z \in \{1, 2\}$. Após a propagação, caso um dos domínios venha a se tornar vazio, o problema não terá solução. Nesse caso desfaz-se a última atribuição de valor e tenta-se outro valor (*backtracking*). Por outro lado, se todos

domínios possuírem um único valor, essa será a solução. Se nenhum dos dois ocorrer, deve-se realizar a busca combinatória pela solução.

Existem diversas técnicas de busca que podem ser usadas na PR. Cada técnica é escolhida para uma aplicação levando em consideração particularidades do problema. Em uma forma básica, porém, genérica, são utilizadas duas políticas de seleção. Na primeira, a seleção é sobre a ordem de valoração das variáveis. Pode-se, por exemplo, selecionar para a próxima atribuição de valores uma variável que seja referenciada por mais restrições ou que possua um domínio menor. A segunda política, vale a ordem de seleção dos valores nos domínios, podendo ir do menor para o maior, ou no sentido contrário, ou até mesmo dos valores do meio do domínio para as bordas. Outras técnicas, como a *split-domain*, não atribui valores diretamente, mas faz uma divisão e conquista por domínios menores e válidos.

5.1.3 Melhorias na modelagem

Assim como uma boa escolha de estratégia de busca pode influenciar drasticamente no tempo de execução, existem outras técnicas para tornar o modelo mais eficiente. Uma delas é a inclusão no modelo de restrições redundantes, e o uso de restrições complexas, como *all_distinct*, *element* e *cumulative* [31]. Esta última é especialmente importante para problemas de agendamento, como o problema em questão. Todas essas restrições mais complexas possuem um tratamento especial no ambiente de PR, sendo implementadas de forma eficiente. Pode-se ainda utilizar-se de técnicas alternativas [18, 4] ao *backtracking*, como o *backjumping*, que ao invés de retornar à variável antecessora na árvore de busca após uma falha, retorna à variável que gerou a inviabilidade, mais acima.

Ainda sobre o modelo, pode-se adicionar restrições para evitar a ocorrência de simetrias, ou seja, a existência de soluções equivalentes. Um exemplo de simetria está em um conjunto de variáveis que podem ter seus valores permutados. A identificação de simetrias pode ser difícil, mas é muito importante, e pode ser complementada por técnicas que consigam identificar um grande número de simetrias durante a busca [21].

Para problemas de otimização, deve-se considerar todas as melhorias citadas, e ainda levar em conta que a PR utiliza-se de estratégias como *Branch-and-Bound* [12] para otimizar uma dada função objetivo. Nesse contexto, podem ser combinadas heurísticas como a busca local, e até metaheurísticas [41] com o *branch-and-bound*, na tentativa de se obter boas soluções, e em menor tempo de execução.

Além da hibridização com outras heurísticas, existem diversas possibilidades de integrar PR com Programação Linear Mista (PLM). A PLM será descrita na seção seguinte.

5.2 Programação Linear Mista

Problemas modelados usando Programação Linear (PL) são problemas de otimização, onde a função objetivo e as restrições são todas lineares. A importância da PL está na existência de uma grande classe de problemas em pesquisa operacional que podem ser formulados dessa forma. E mais, historicamente, muitas das idéias de PL inspiraram conceitos de otimização como a *dualidade*, *decomposição* e a *convexidade*.

A diferença entre PL e Programação Linear Mista (PLM) está na existência de variáveis no modelo que possuem o seu domínio no conjunto dos inteiros. Assim, definimos um modelo de PLM na forma padrão como,

$$\begin{aligned} \text{minimizar } z = & \mathbf{c}_1\mathbf{x} + \mathbf{c}_2\mathbf{y} \\ \text{sujeito a: } & \mathbf{A}_1\mathbf{x} + \mathbf{A}_2\mathbf{y} = \mathbf{b} \\ & \mathbf{x}, \mathbf{y} \geq \mathbf{0}, \mathbf{x} \in \mathbb{R}^n, \mathbf{y} \in \mathbb{Z}^m \end{aligned}$$

onde $\mathbf{x}$ e $\mathbf{y}$ são os vetores de *variáveis de decisão*, $\mathbf{A}_1$ e $\mathbf{A}_2$ formam as matrizes de *coeficientes das restrições*, $\mathbf{b}$ é o vetor dos *termos independentes* e $\mathbf{c}_1$ e $\mathbf{c}_2$ são os vetores de custos da *função objetivo*, representada pela expressão $z = \mathbf{c}_1\mathbf{x} + \mathbf{c}_2\mathbf{y}$. Se não existirem os vetores $\mathbf{y}$, $\mathbf{c}_2$ e $\mathbf{A}_2$, tem-se um problema de PL.

Há atualmente duas famílias de técnicas para resolução de modelos de PL. Ambas caracterizam o problema como um *poliedro* delimitado pelos hiperplanos das equações lineares das restrições. Demonstra-se, daí, que a solução ótima está em um ou mais pontos extremos de tal poliedro. A primeira família, que engloba os métodos denominados *Simplex*, foi introduzida por Dantzig em 1949 [14] e esses métodos buscam a otimalidade “caminhando” iterativamente pelas arestas do poliedro, verificando as *soluções básicas* (ou pontos extremos) até encontrar a ótima. É demonstrado que a complexidade de pior caso para o algoritmo Simplex é exponencial, embora ele apresente um ótimo desempenho na prática. Já a segunda família engloba os métodos de *Pontos Interiores* ou *Barreiras*, estes sim, de complexidade polinomial. Eles exploram o poliedro a partir de pontos do interior da região viável, até encontrar um ponto extremo ótimo.

Apesar da pequena distância na formulação de problemas de PL para PLM, o problema geral na formulação PLM é NP-Difícil. É preciso, portanto, recorrer a algoritmos como *branch-and-bound* [12], aliados a outras técnicas, para se encontrar limitantes superiores e inferiores para as soluções, enquanto realizando uma busca mais eficiente no espaço de soluções.

Comparada com a PR, a modelagem PLM é geralmente mais complexa em consequência da dificuldade de se representar certas restrições como equações lineares. Contudo, alguns modelos PLM para problemas difíceis são resolvidos de forma eficiente e

exata, principalmente, quando utilizam técnicas como os *planos de corte* [52]. Ainda existem diversas maneiras de integrar PR e PLM [39], aproveitando-se, por exemplo, da capacidade de inferência fornecida pela PR e das relaxações na PL.

5.2.1 Fluxo em Redes

Os modelos de fluxo em redes [1] são uma subclasse muito importante dos problemas de programação linear. Existem inúmeras aplicações desses modelos para sistemas de distribuição e de comunicação. De uma forma abstrata, os modelos representam uma rede formada por localidades, cada qual com uma demanda e estoques de um recurso, e por ligações entre as localidades, cada qual com uma direção, capacidade e custo de transporte desses recursos. Busca-se minimizar o custo total de transporte satisfazendo as condições de viabilidade, as demandas e utilizando os estoques presentes.

Há diversos problemas clássicos modelados como fluxo em redes como, por exemplo, o problema do caminho de custo mínimo, o problema do fluxo máximo, o problema da circulação e o problema de alocação. Nesse contexto, estamos focando no mais geral deles, que é o problema do fluxo de custo mínimo. Para formulá-lo matematicamente precisamos das seguintes definições:

- $G = (N, A)$ é uma rede (grafo) direcionada, onde $N = (1, \dots, n)$ é o conjunto dos nós (vértices) e A é o conjunto de arcos (arestas) direcionados representados por pares $(i, j) \in N \times N$ e $i \neq j$, com sentido de i para j (ou seja $i \rightarrow j$).
- c_{ij} é o custo por unidade de fluxo que passa pelo arco $(i, j) \in A$.
- l_{ij} e u_{ij} são respectivamente o limite mínimo e máximo de fluxo que pode passar pelo arco $(i, j) \in A$.
- b_i é um número inteiro que representa demanda quando $b_i < 0$, e fornecimento, quando $b_i > 0$, no nó i .
- x_{ij} é o fluxo que passa pelo arco $(i, j) \in A$.

Sendo assim, o problema do fluxo de custo mínimo pode ser formulado como:

$$\begin{aligned}
&\text{minimizar } z = \sum_{(i,j) \in A} c_{ij} x_{ij} \\
&\text{sujeito a:} \\
&\quad \sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} x_{ji} = b_i \quad \forall i \in N \\
&\quad l_{ij} \leq x_{ij} \leq u_{ij} \quad \forall (i,j) \in A
\end{aligned}$$

Essa formulação é bastante abrangente. No entanto, para o **PPAORO** é necessário compreender uma outra variante, onde há diferentes recursos (*commodities*) passando pela rede e é necessário manter os fluxos de cada uma separadamente. Além disso, será necessário representar a rede em diferentes períodos, ou seja, o tempo será discretizado para demonstrar como o fluxo evolui durante o tempo.

Diferentemente do problema de fluxo de custo mínimo que possui algoritmos especializados para sua resolução, o problema de fluxo *multicommodities*, sendo mais geral, é resolvido usando o modelo de programação linear equivalente. Já os multiperíodos podem ser modelados com um fluxo de custo mínimo e, portanto, com um fluxo *multicommodities*.

Para formular o problema de fluxo *multicommodities* definimos:

- $G = (N, A)$ é uma rede (grafo) direcionada, como definido anteriormente.
- $K = (1, 2, \dots, k)$ é o conjunto de *commodities*.
- c_{ij}^k é o custo de transferir uma unidade de fluxo da *commodity* k pelo arco (i, j) .
- x_{ij}^k é o fluxo da *commodity* k passando pelo arco (i, j) .
- $\mathbf{c}^k$ e $\mathbf{x}^k$ são respectivamente os vetores de custos e fluxos da *commodity* k para todos os arcos.
- l_{ij}^k e u_{ij}^k são os limites mínimos e máximos, respectivamente, para o fluxo da *commodity* k pelo arco (i, j) .
- l_{ij} e u_{ij} são os limites mínimos e máximos, respectivamente, para o fluxo de todas as *commodities* pelo arco (i, j) .
- $\mathbf{M}$ é uma matriz $N \times A$ de incidência nó-arco.
- $\mathbf{b}^k$ é um vetor de inteiros que dá a demanda ou fornecimento da *commodity* k para todos os nós.

Tendo essa notação, podemos formular o problema do fluxo *multicommodities* como:

$$\begin{aligned}
 &\text{minimizar } z = \sum_{k \in K} \mathbf{c}^k \mathbf{x}^k \\
 &\text{sujeito a:} \\
 &\quad l_{ij} \leq \sum_{k \in K} x_{ij}^k \leq u_{ij}, \quad \forall (i, j) \in A \\
 &\quad \mathbf{M} \mathbf{x}^k = \mathbf{b}^k, \quad \forall k \in K \\
 &\quad l_{ij}^k \leq x_{ij}^k \leq u_{ij}^k, \quad \forall (i, j) \in A \text{ and } \forall k \in K
 \end{aligned}$$

Um outro resultado sobre fluxo em redes, importante para nosso trabalho, é a possibilidade de decompor um fluxo obtido em caminhos e ciclos. Para compreender isso, definimos $\mathcal{P}$ e $\mathcal{W}$ como os conjuntos de todos os caminhos e ciclos na rede G . Assim, podemos ter fluxos $f(P) : \forall P \in \mathcal{P}$ e $f(W) : \forall W \in \mathcal{W}$ de forma que:

$$x_{ij} = \sum_{P \in \mathcal{P}} \delta_{ij}(P) f(P) + \sum_{W \in \mathcal{W}} \delta_{ij}(W) f(W)$$

Na equação acima, $\delta_{ij}(P)$ e $\delta_{ij}(W)$ dizem se o arco está contido ($\delta_{ij} = 1$) ou não ($\delta_{ij} = 0$) no caminho ou ciclo em questão. Compreende-se, por essa equação, que o fluxo em um arco é composto da contribuição de fluxo de diversos caminhos e ciclos que contém aquele arco.

A equação acima conduz ao algoritmo de decomposição mostrado no Algoritmo 1. Esse algoritmo de decomposição clássico [1] será a base para a criação dos planos de entrega. Como os nós representarão demandas, produções, tanques e trechos de uma rota, um caminho na rede conterá todas as informações necessárias para criação de tais planos, após a decomposição do fluxo obtido. Como será visto, o algoritmo poderá ser simplificado, pois não haverá ciclos em nossa rede.

Algoritmo 1 Algoritmo de Decomposição de Fluxo

Entrada: $G = (N, A)$ com fluxo nos arcos x

Saída: $\mathcal{P}$ com $f(P), \forall P \in \mathcal{P}$, $\mathcal{W}$ com $f(W), \forall W \in \mathcal{W}$

NOTAÇÃO:

y - cópia do fluxo nos arcos

$A(y) = \{(i, j) \in A | y_{ij} > 0\}$ (Arcos com fluxo positivo em y)

$N(y) = \{i | (i, j) \in A(y) \text{ or } (j, i) \in A(y)\}$ (Nós incidentes nos arcos em $A(y)$)

$G(y) = (N(y), A(y))$

$\mathcal{S} = \{i \in N(y) | b_i > 0\}$ (nós fornecedores)

$\mathcal{D} = \{i \in N(y) | b_i < 0\}$ (nós de demanda)

s e t são os nós de início e fim do caminho P .

$\Delta(P) = \min\{b(s), -b(t), \min\{y_{ij} | (i, j) \in P\}\}$ (Capacidade do caminho P)

$\Delta(W) = \min\{y_{ij} | (i, j) \in W\}$ (Capacidade do ciclo W)

```

1: procedimento DECOMPOSICAODEFLUXO
2:    $y = x, \mathcal{P} = \emptyset, \mathcal{W} = \emptyset$ 
3:   enquanto  $A(y) \neq \emptyset$  faça
4:      $s = \text{SELECIONAR}(y)$ 
5:      $\text{BUSCAR}(s, y)$ 
6:     se Ciclo  $W$  encontrado então
7:        $\mathcal{W} = \mathcal{W} \cup \{W\}$ 
8:        $f(W) = f(W) + \Delta(W)$ 
9:        $y_{ij} = y_{ij} - \Delta(W), \forall (i, j) \in W$ 
10:    fim se
11:    se Caminho  $P$  encontrado então
12:       $\mathcal{P} = \mathcal{P} \cup \{P\}$ 
13:       $f(P) = f(P) + \Delta(P)$ 
14:       $y_{ij} = y_{ij} - \Delta(P), \forall (i, j) \in P$ 
15:       $b(s) = b(s) - \Delta(P)$ 
16:       $b(t) = b(t) - \Delta(P)$ 
17:    fim se
18:    Atualizar  $A(y), N(y), \mathcal{S}, \mathcal{D}$ 
19:  fim enquanto
20: fim procedimento
21: função SELECIONAR( $y$ )
22:   se  $\mathcal{S} \neq \emptyset$  então
23:     retornar  $s \in \mathcal{S}$ 
24:   senão
25:     retornar  $s \in N(y)$ 
26:   fim se
27: fim função
28: função BUSCAR( $s, y$ )
29:   Fazer uma busca em profundidade iniciando com o nó  $s$ 
   até achar um ciclo  $W$  em  $G(y)$ 
   ou um caminho  $P$  em  $G(y)$  terminando no nó  $t \in \mathcal{D}$ 
30:   retornar  $W$  ou  $P$ , de acordo com o encontrado
31: fim função

```

Capítulo 6

Um modelo para o Problema de Agendamento

Prólogo

O artigo deste capítulo foi publicado em um dos periódicos internacionais mais importantes na área de Programação por Restrições, o *Constraints* em 2010 [27]. O artigo pode ser visto como uma evolução do artigo apresentado nos anais do congresso *The 14th Principles and Practice of Constraint Programming* (CP'08), na série *Lecture Notes in Computer Science* [38]. Este por sua vez é uma evolução de outro artigo publicado nos anais da conferência, *Proceedings of the 11th IEEE International Conference on Computational Science and Engineering - CSE'08* [37], e eletronicamente, pela editora do *IEEE*. Os dois artigos de 2008 compõe a dissertação de André Ciré [10].

O artigo apresenta o Problema de Planejamento e Agendamento de Operações em uma Rede de Oleodutos (*Pipeline Planning and Scheduling Problem*) com grande detalhamento cobrindo as restrições já apresentadas no capítulo 2. A seção de trabalhos relacionados (*Related Work*) do artigo foi atualizada no capítulo 3 dessa dissertação e a arquitetura utilizada é a mesma mostrada no capítulo 4.

A fase de planejamento é tratada por uma heurística gulosa incremental que foi brevemente introduzida nos trabalhos anteriores [38, 37]. Esse procedimento cria os planos de entrega um após o outro, conforme funções de avaliação que utilizam estimativas sobre a rede até aquele momento. Por exemplo, o produto, órgão de destino e prazo são escolhidos analisando todas as combinações e escolhendo a base onde há uma demanda mais urgente para o produto.

O foco do artigo, no entanto, é a fase de agendamento (*Scheduling Phase*). Essa fase foi decomposta em dois problemas: o sub-problema de sequenciamento (*Sequencing Sub-problem*) e o sub-problema de escalonamento (*Scheduling Sub-problem*).

Primeiramente, o sequenciamento definirá a ordem das operações dos planos nos tanques de origem e destino e nos dutos trafegados que formam a rota. Após isso, o escalonamento dará os tempos exatos de início e término dessas operações. Cada um desses problemas é tratado por meio de um modelo de programação por restrições próprio que faz uso extenso de *restrições globais* específicas para agendamento. Além disso, novos métodos de busca de solução, especializados para a estrutura da rede de dutos, também são propostos.

Outra contribuição estendendo os trabalhos anteriores [38, 37] é o grafo de empurramento (*Pushing Graph*), equivalente a uma restrição global, aumentando a inferência de precedências entre operações ocorrendo em diferentes dutos. Essa estrutura é específica para problemas de agendamentos em dutos, pois supõe que os dutos estão sempre preenchidos completamente e a entrada de um volume em uma extremidade só é possível com a saída de um mesmo volume na extremidade oposta. Ela pode ser vista como uma representação do problema, permitindo a aplicação de mais restrições globais tais como a restrição *flow* [24].

Os procedimentos foram testados sobre 4 instâncias reais, as quais utilizavam uma mesma rede com 14 órgãos, 29 dutos e 32 produtos diferentes armazenados em 242 tanques. Um exemplo de execução sobre uma instância artificial ilustra o procedimento.

A Hybrid Model for a Multiproduct Pipeline Planning and Scheduling Problem

Tony M. T. Lopes, Andre A. Cire, Arnaldo V. Moura, Cid C. de Souza
 Institute of Computing - University of Campinas
 13081-970, Campinas, SP
 {arnaldo, cid}@ic.unicamp.br, {andre.cire, tony.lopes}@gmail.com

Abstract

Brazilian PETROBRAS is one of the world largest oil companies. Recurrently, it faces a very difficult planning and scheduling problem: how to operate a large pipeline network in order to adequately transport oil derivatives and biofuels from refineries to local markets. In spite of being more economical and environmentally safer, the use of a complex pipeline network poses serious operational difficulties related to resource allocation and temporal constraints. The current approaches known from the literature only consider a few types of constraints and restricted topologies, hence they are far from being applicable to real instances from PETROBRAS. We propose a hybrid framework based on a two-phase problem decomposition strategy. A novel Constraint Programming (CP) model plays a key role in modelling operational constraints that are usually overlooked in literature, but that are essential in order to guarantee viable solutions. The full strategy was implemented and produced very adequate results when tested over large real instances.

1 Introduction

Scheduling problems have been receiving increasing attention in the last years, mainly due to the need to deal efficiently with very large real scenarios in order for companies to persevere in highly competitive markets. A number of notable examples stems from the oil industry, where oil prospection and refined product transportation are major sources of costs. As such, PETROBRAS, the 14th largest oil company in the world¹, faces a very difficult transportation problem in which ethanol and several petroleum derivatives, like gasoline, diesel, and naphtha must be transported from refineries to depots where consumer markets are located. Such complex transportation problems usually must be solved in the presence of very complicated facility operational restrictions.

¹See www.energyintel.com.

Pipeline networks are the preferred way for transporting oil refined products. In contrast to tanks and waterborne alternatives, their costs do not escalate sharply with distance. In an example cited in [1], assuming each truck holds 200 barrels (8,400 gallons) and can travel 500 miles per day, it would take a fleet of 3000 trucks, with one truck arriving and unloading every 2 minutes, to replace a 150,000-barrel per day, 1,000-mile pipeline. In the United States of America, oil pipeline shipments account for more than 17% of the transported volume but less than 2% of the country freight cost [2]. As another example, the Brazilian pipeline network owned and operated by PETROBRAS. It has an extension of 7,000 kilometers, comprising 30 individual interconnecting pipelines, all with very different physical characteristics, and through which more than 30 different types of products circulate. There are 14 distribution depots that can store up to 10 millions cubic meters of these products, stocked in more than 200 tanks located at such depots. A partial illustration of the Brazilian southeastern network is shown in figure 1.

Pipelines must always be completely filled with products, meaning that a volume must be *pushed* into a pipeline in order to pump out the same volume at the other extremity. Flow directions can also change dynamically. Moreover, certain products are chemically incompatible and can not make contact with each other in a pipeline, so as to ensure their required quality level. Also, flow rates depend on product, flow direction and on the particular pipeline being used. At the depots, not all departing and arriving operations can be simultaneous, due to restrictions imposed both by the internal valve and duct layouts, as well as by the number of local pumps. Tanks can store just one type of product. Extraction or injection of volumes from or into a tank can not be simultaneous, and must always obey the tank capacities. As can be inferred, the operation of such a network is subjected to a complex set of physical and operational constraints.

Due to its size and complexity, as well as to its financial impact, the efficient operation of this large oil pipeline network is one of the most strategic problems faced by logistics at PETROBRAS today. Yet, nowadays, the problem is basically solved manually, by executing a trial-and-error process with the aid of a proprietary simulator that checks whether some simple physical constraints are being satisfied. This process is very time consuming and, not rarely, the final results still violate some of the more complex restrictions. Clearly, this manual process is far from optimal and limits the efficiency of the network operation, as down-level operators must frequently apply corrections in delivery orders as a means to make the system work appropriately. In fact, it is common for the company to use trucks for transporting pending volumes, thus increasing the overall transportation costs, a situation that could be avoided by a more intelligent use of the pipeline network.

Studies from the literature usually focus on more restrictive and smaller pipeline topologies, each with a reduced set of operational constraints that, despite making the problem more tractable, are far from representing real-world scenarios. This is a conse-

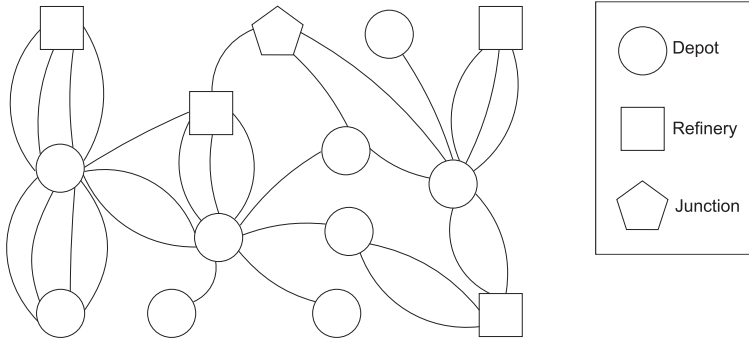


Figure 1: PETROBRAS pipeline network.

quence of the high number of hard sequencing and timing constraints involved, complicated when large and more generic topologies are considered.

We propose new algorithms for generating feasible solutions for a very large pipeline planning and scheduling problem, where most of the hardest real-world constraints are taken into account. Our approach has two phases: the *planning phase* is implemented as a constructive heuristic that generates *delivery orders*, representing transfers between two depots; and in the *scheduling phase*, a Constraint Programming (CP) model, is used to establish an ordering among the delivery orders, at each pipeline and each tank. The resulting algorithm, specially tailored to deal with large instances, generates more reliable delivery orders and can also be used to validate production and demand trial scenarios. This strategy proved to be very flexible, permitting easy addition or removal of operational requirements, and also facilitating the test of new search heuristics. As will be discussed later, CP was at the core of the computational model devised, being capable of finding good operational solutions for real problem instances in an adequate amount of computer time. Several reasons motivated the use of CP. Primarily, the scheduling problem is highly over-constrained and has several non-linear constraints, which can be easily modelled in the CP paradigm. Besides, the main goal was to search for a feasible solution, and so the choice of variables for value assignment in the CP model benefits from special restart strategies.

This project was a joint work with PETROBRAS, the Brazilian state-owned oil company, which provided both problem specifications and instances, as well as analyzed the proposed solutions. The prototype developed is being considered for use as an auxiliary tool to aid planners at PETROBRAS.

This article is an extension of our earlier works on this topic [3, 4]. It contains a much more detailed description of the problem and of the new algorithms. The text is organized as follows. Section 2 describes the problem. Section 3 presents some previous works related to pipeline scheduling. Section 4 discusses in detail the heuristic and CP

models that comprise the planning and scheduling phases. Finally, section 4 discusses our computational results while section 6 presents our conclusions and points to further work.

2 Problem Definition

A pipeline network system is composed of three sets: *tanks*, *depots*, and *pipelines*. Tanks are used for product storage. Depots are geographically dispersed. They represent refinery facilities and oil-derivative local markets. Each depot has its own subset of tanks. Pipelines interconnect the depots and are used for product transportation. A pipeline connects only two depots.

An illustration of a pipeline system is presented in figure 1, in which products (or *fluids*) F_0 , F_1 , and F_2 can circulate. The figure shows 4 depots, D_0, D_1, D_2 , and D_3 , connected by pipelines P_0, P_1, P_2, P_3 , and P_4 . A label T_{ijk} refers to the i -th tank in depot D_j for stocking product F_k . Note that two pipelines connect depots D_2 and D_3 , which is a common situation in practice.

Volumes must be extracted from tanks before pumped into a pipeline. On being pumped out of a pipeline, volumes can either enter a tank or move directly into another pipeline. The sequence of pipelines traversed by a volume when moving from its origin to its destination comprises its route. More precisely, we define a *route* as an alternating sequence of depots and non-repeating connecting pipelines. For example, the sequence $D_0P_1D_2P_3D_3$ represents a valid route in figure 1. A pipeline in a route is called a *segment*. All product volumes in circulation must have a pre-defined route assigned to it. Also, a volume cannot be stocked at intermediate depots in any route; it can only be deposited in a tank at its destination depot.

The problem consists of scheduling *pumping operations* in order to satisfy the depots' inventory constraints. Individually, a pumping operation is taken as a continuous and atomic product injection into a pipeline at its origin depot. Each operation specifies information about the pumped product, volume, route, origin, and destination tanks, as well as about its start and end pumping times. The operations must also obey a complex set of operational restrictions over a given planning time period, or *horizon*. Restrictions regarding tanks, pipelines, and depots are described in the following three sections, respectively. Inventory constraints are described in section 2.3.

All volumes are given in standardized metric units.

2.1 Tank Restrictions

- (1) Tanks have a fixed maximum capacity which must always be respected. Minimum capacities are all set to zero units.

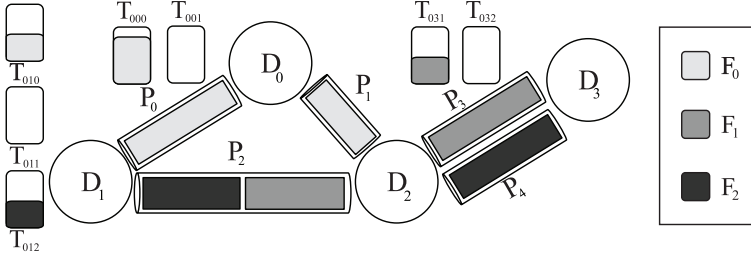


Figure 2: A pipeline network example.

- (2) A tank can only store one type of product during the whole planning horizon. This constraint, required by field operators, ensures adequate product quality by avoiding possible mixtures. A depot can contain more than one tank per product, but does not necessarily contain tanks for all products. It is possible that a depot contains no tanks at all, being used solely as an intermediate transmission facility between two pipelines. In figure 1, depot D_2 depicts such a situation.
- (3) All injection and extraction operations targeting a common tank must be time disjoint. This is due to the internal connections of subsidiary pipes and valves inside a depot. Therefore, a new operation at a tank can only begin when all previous scheduled operations at that tank are already completed.
- (4) The initial product stock level at each tank must be respected.

Problem specifications in which constraint 3 is satisfied are said to have the *individual tank* property. Some alternative models [5] relax this restriction by considering virtual tanks for each product at each depot. Such virtual tanks aggregate the capacities of all real tanks for a product in a given depot.

2.2 Pipeline Restrictions

- (5) Pipelines must be completely filled with products at all times, since they are pressurized. Hence, in order for a certain product to leave a pipeline, it is necessary to push it out by injecting the same volume at the other pipe extremity. For example, in figure 1 suppose that a certain volume of product F_2 , at the extremity of pipeline P_4 connecting into D_3 , is to be pumped out. If we had enough volume of another product, say F_0 , at the extremity of pipeline P_2 facing depot D_2 , and if its route suffix were $P_2D_2P_4D_3$, we could schedule a pumping operation with origin at D_1 making active the route $D_1P_2D_2P_4D_3$. Executing now this operation would push F_2 into a tank at D_3 , as desired. Note that it will be necessary to have enough volume of some product to insert into P_2 at D_1 . Further, the pumping operation

with origin at D_1 is just another operation to be scheduled in the network, among many others. So it must have its product, volume, route, origin and destination tanks already defined, as well as its start and end times.

- (6) Pipelines must be able to operate in an intermittent fashion, *i.e.*, some time may elapse between the end of a pumping operation and the beginning of the next one in a certain pipeline. Even though longer continuous transfers are more economical and thus preferred, a pipeline flow can be temporally suspended for several reasons. For instance, as pipelines in a depot may share the same pumps, they might not be able to function simultaneously (see also constraint 12). Another common scenario is when three pipes form a “Y” connection. Suppose that the flow is now downward from the right segment into the lower common segment. If in a route containing the left segment there are products with a more pressing deadline, as soon as the volume of the present operation that is now flowing from the right is completely inserted into the common segment, the present operation on the right side may be stopped thus allowing the flow from the left segment to start. If this situation repeats itself, we will see the right branch operating in an intermittent fashion, giving way so that more urgent products can circulate first into the common segment. Note however that once a volume is inserted in a pipeline it can not be separated in two parts by the insertion of other volumes in between.
- (7) All injection and extraction operations targeting a common pipeline must be time disjoint. In other words, it is not possible to simultaneously inject products into a pipeline from two different sources, for instance, to increase the resulting flow rate. Similarly, a product leaving a pipeline can not be injected into different tanks or pipelines.
- (8) Some pairs of products, so called *incompatible pairs*, can not make contact within a pipeline, otherwise a quality loss will ensue. For example, both products inside pipeline P_2 in figure 1 must be compatible. Such operational restriction is also known as the *interface constraint*.
- (9) As there is a limited number of pumps per depot, and given that products have different densities, a maximum flow rate must be observed per pipeline, per product and per flow direction. Thus, the pumping operation flow rate can take any real value between 0 and the minimum value of the maximum flow rates of all products in the current route. We do not consider minimum flow rates due to the lack of accurate data. However, operators accepted this as a valid simplification since pressing deadlines naturally require products to be pumped as fast as possible.

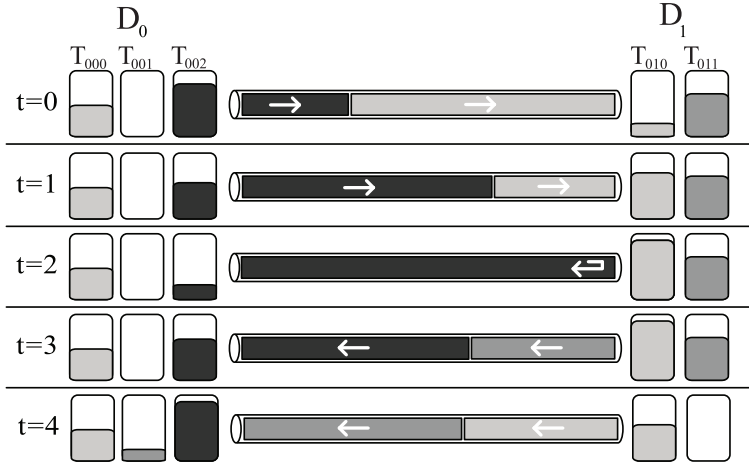


Figure 3: Example of a flow reversal.

- (10) Flows can change direction dynamically, an event called *pipeline flow reversal*. An example of a reversal is illustrated in figure 3 for a network with a single pipeline topology. During the time period from $t = 0$ to $t = 2$, product F_2 is extracted from tank T_{002} in depot D_0 , and is used to push another product into tank T_{010} at depot D_1 . As soon as the first product is completely injected into its destination tank at $t = 2$, product F_2 must return to the first depot, since there is no tank for this kind of product at depot D_1 . This is done by using product F_1 from tank T_{011} , at D_1 , to push F_2 back to its original tank, at D_0 , thus forcing a change of flow direction in the pipeline at $t = 3$ and $t = 4$.
- (11) Although there is no restriction barring the creation of new routes, the most common choices as dictated by previous human experience should be preferred. These routes are given as input and will be denoted by a set R throughout the text. Note that we are using a fixed set of routes, and we are not exploring all possible routes through the network. In some cases, we might use a route from this set which was not used before in manual solutions to transport specific products, thus deviating from the usual routing that appear in manual solutions for those products.

2.3 Depot Restrictions

- (12) Each depot has an upper limit on the number of outgoing pumping operations, which depends on the number of locally available pumps. Therefore, it may not be possible for all the pipelines at a depot to be in operation simultaneously.
- (13) At each depot, tanks are connected to external pipes by a mesh of shared structures,

like valves and auxiliary pipes. Depending on the internal arrangement at a depot, certain sending and receiving operations can not be active simultaneously. Such sets of operations are called *forbidden alignment configurations*, or simply *forbidden sequences*. For instance, in a certain depot one may not be able to send gasoline through any pipeline while diesel is being received by any other pipeline.

2.4 Inventory Constraints

The *inventory constraints* are related to tank capacities and the maintenance of desired stock levels at each depot. The stock of a product at a certain depot at time t is obtained by summing the volumes of that product at time t in all tanks that can stock the product at the given depot. In this work, we only consider inventory constraints related to the satisfaction of tank capacities. Nevertheless, the pipeline operators also provide the desired maximum and minimum level of product stock levels per depot, which are being used solely as a guide to the search procedures.

The stock levels in a depot change mainly due to *production* and *demand* operations. Productions represent volume creation at refineries, while demands represent volume consumption by local markets. Both are planned in advance, using market estimates and other data such as raw products availability and refinery capabilities. Usually, a refinery depot produces more than it can actually store within the given planning horizon, so as to satisfy market demands at other locations. This, of course, requires products to circulate throughout the pipeline network. In addition, an excess of inventory in each depot is also preferred to prevent against emergency situations, such as the repair of a damaged pipeline.

The consumption rate of each product may vary greatly according to monthly seasonal markets. Moreover, it is difficult to foretell exactly the local market needs for a long time horizon. As a result, pipeline operators are required to constantly update the network schedule to accommodate new demands, guaranteeing they will be satisfied on time. Given that, and in order to represent productions and demands in a reliable and computationally feasible way, the following model was devised, and accepted by the pipeline operators as an adequate simplification. Each production and demand, given as input, is composed of a volume, a product, its corresponding depot, and a time frame that defines when the production or demand must be serviced. The production (demand) volume can be injected (extracted) instantly at any time within its time frame. Also, this volume must be necessarily injected into (extracted from) one or more tanks, the selection of which being responsibility of the algorithms. In this way, productions and demands are regarded as common tank operations. Hence by constraint 3 they can not occur simultaneously with other tank operations. Usually, the time frame given for a demand or production

corresponds to one day, *i.e.*, its volume can be injected or extracted at any time during the day.

Although instantaneous volume injection and extraction is not physically realizable, the operators believe this is a reasonable model to capture variations about local markets, given the inherent uncertainties about the real data. Operators also have some flexibility to adjust the timing when products are delivered to clients and, to a smaller degree, they can also reschedule the production planning at refineries. Furthermore, this model gives the operators flexibility to drive the algorithms under various scenarios, allowing them to simulate different production and demand curves.

2.5 Definition of a Solution

The problem input comprises a *static* and a *dynamic* component. The static component is described by the sets of tanks, pipelines, depots, and their operational parameters, such as the maximum number of simultaneous outgoing pumping operations at each depot. The dynamic component is given by a set of productions and demands (see section 2.3) that must be satisfied. The dynamic component also contains the *initial network state*. The initial state is defined by the sequence of product volumes inside each pipeline, the assigned route for each of these volumes, and the stock level in each tank at the beginning of the horizon. It also gives the set of ongoing pumping operations at the initial instant.

Given the problem input, a feasible *solution* is defined by a set of pumping operations which ensures that all demands and productions will be fulfilled in their given time frames. That is, all product volumes from productions (demands) are to be correctly stored (extracted) without violating tank capacities. In addition, the pumping operations must obey all operational constraints stated in sections 2.1, 2.2, and 2.3. Finally, it is also necessary to make explicit the exact times and tanks allocated for each production and demand. This is done by representing the associated injections and extractions, respectively, as pumping operations that have a null route.

The program output is a table containing a list of all pumping operations. This list is then submitted to a proprietary software that executes simple consistency checks. Passing the consistency checks, the list is then handed to logistic engineers for use as a guide during the actual operation of the pipeline network. An example of such list is shown in table 1. Column headings are as follows: T_i and T_f are the start and end times, respectively; Vol is the volume pumped; Pd is the product code; Tk_{Or} and Tk_{Dt} are origin and destination tank codes, respectively; and $Route$ is the route code. A full solution table would contain several hundred similar lines.

In this work, no objective function are considered, as more accurate data about energy and operational costs are still being gathered by PETROBRAS. However, as we will see in

Table 1: Solution Example

T_i	T_f	Vol	Pd	Tk _{Or}	Tk _{Dt}	Route
2075	2362	858	F_1	T_{011}	T_{031}	$D_1 P_0 D_0 P_1 D_2 P_3 D_3$
3456	4021	722	F_1	T_{032}	T_{001}	$D_3 P_3 D_2 P_1 D_0$
4857	4869	30	F_0	T_{010}	T_{000}	$D_1 P_2 D_2 P_4 D_3 P_3 D_2 P_1 D_0$
...	...	...	...	...	...	...

Table 2: Comparison of the main approaches.

References	[9]	[10, 11, 12]	[5, 13]	[14, 15]	[16, 17]	[18, 19, 20]	Our work,[3]
Depots	8	5	12	7	2	2	14
Pipelines	8	4	28	7	1	1	29
Tanks	40	25	84	28	16	12	242
Products	5	4	7	4	8	6	32
Horizon	14 days	3 days	5 days	80 units	5 days	30 days	10 days
Continuous time		✓	✓				✓
Variable Flow rate		✓			✓	✓	✓
Individual Tanks					✓		✓
General Topology	✓		✓	✓			✓
Forbidden Sequences		✓	✓		✓	✓	✓
Main Technique	GA	MILP,VNS	A-Teams,MILP	GA,MILP	MILP,CP	MILP	Heuristic+CP
Results	Feasible	Optimum	Infeasible	Feasible	Optimum	Optimum	Feasible

section 3, the apparently simpler task of finding a solution satisfying such a huge number of intricate constraints is a great challenge in itself.

3 Related Work

The pipeline operation problem comprises a highly complex and diversified set of constraints. Examples of difficult operational restrictions include product sequencing inside a pipeline, interface constraints, mass balance in tanks and various inventory management conditions. Complexity analysis of some relaxed versions of the pipeline scheduling problem can be found in [6, 7, 8].

Due to the problem's economical and practical importance, a number of methods have been proposed in the literature to automate the planning of pipeline network operation. The characteristics that distinguishes such approaches from each other are mainly related to the network topology and the granularity of the scheduling unit of time. As real-world scenarios require the satisfaction of a high number of constraints, the level of detail is an indication of the effectiveness of their solutions in practice. Table 2 summarizes some important previous approaches and their most important characteristics.

The most studied topology consists of one pipeline connecting a single origin to multiple destinations, which are distributed along the pipeline. Usually, the origin represents

an oil refinery, the destinations are consumer markets, and the flow in the pipeline is unidirectional. The study of simpler networks are motivated by the fact that many real scenarios are composed of a single pipeline. In addition, it is easier to model some of the problem restrictions, specially those involving variable flow rates, product storage, and energy costs. An early work along this line appeared in [21], where an artificial intelligence technique called *Beam Search* was applied to an Indian network. Although the procedure strongly relies on human intervention, it is capable of handling tank capacities and interface constraints.

Also, several Mixed-Integer Linear Programming (MILP) optimization models were developed for this type of topology. The work in [22] deals with a network composed of 5 depots, where 4 products must be transported among them. The proposed model is based on an uniform and discrete time representation and on disjunctive programming. It handles a large number of hard constraints, such as mass balances and product demands, besides minimizing storage and interface costs. However, the authors were not able to obtain optimal solutions, even considering a simple 3-day planning horizon. For this reason, in [23] the model was improved with the addition of integer cuts and redundant constraints, which enhanced the MILP computational performance.

In [10], a new continuous time and volume MILP formulation was proposed for the same problem presented in [23]. The continuous representation was effectively smaller in terms of the number of binary variables and constraints. As a result, optimal solutions were obtained in about three orders of magnitude faster than in [23]. The continuous representation was also used in [11], where the authors additionally modelled variable pumping flow rates and stock management decisions, formulating the problem also as a Mixed-Integer Non-linear Programming (MINLP) model.

As described in [11], many studies focus on decomposition techniques for single topologies. The work in [16] presents a 3-stage decomposition strategy for a network composed of a refinery, a pipeline, and a depot with a set of tanks. The first stage defines the tanks for each pumping activity. The second stage heuristically creates temporal constraints taking into account the demand requirements. Finally, the last stage is an MILP model to decide upon the pumping times, while having fixed values for the previous parameters. All the MILP models were based on an uniform time discretization. Later, the work in [17] proposed an integrated approach, using both CLP and MILP models, in order to solve the same instances, generating good solutions.

More recent works [18, 19] study a Portuguese pipeline network, with a topology similar to that considered in [16]. The authors propose an MILP model with new inventory constraints, based on [10]. In [20], the model is improved to consider intermittent pipelines and variable flow rates. Moreover, a rescheduling strategy is implemented in order to deal with possible perturbations on the instance data.

There are also studies regarding multiple origins, multiple depots, and various pipelines interconnecting them. They represent more realistic networks, since the production of oil derivatives is usually distributed in several geographic locations. In these scenarios, the problem complexity is much higher than with simpler topologies. In addition to inventory management at each depot and product sequencing in the pipelines, it is also necessary to take into account operational constraints that limit the simultaneous usage of different pipelines. Also, the models must ensure the correct synchronization of direct flow transmission between pipelines, which is common in such scenarios. In general, the current approaches for more complex topologies neglect most of the operational constraints in order to make the problem tractable.

Among such approaches, [13] proposed an MILP based on a network flow model to solve a relaxed version of the problem. The specification included about 14 depots and 29 pipelines, but considered that all depots contained tanks for all products. Furthermore, the pumping flow rate was fixed per pipeline and only simple inventory constraints were considered. Despite these relaxation, it took more than 50 hours of computer time just to find the LP initial basis. In [5], the authors propose the use of an heuristic method known as *A-Teams*, adapting the pipeline scheduling to a *job shop scheduling*. However, they were still unable to find feasible solutions.

The work from [24] implements a Genetic Algorithm for a directed tree network topology with 8 depots and with only a few restrictions, focusing mostly on the basic pipeline operations. The authors further improved the model in [15], hybridizing the evolutionary algorithm with simple MILP models. A more recent application of meta-heuristics to more general topologies, but with relaxed operational constraints, can be found in [9, 12].

Another research direction is discussed in [25], which addressed the pipeline scheduling problem as a general-purpose planning problem. A new domain specification and a planner were created in order to deal with general topologies and some tank constraints. Although not effective for large instances, the work established a new benchmark at the *International Planning Competition* [26].

The use of Constraint Programming for general topologies was first proposed in [4]. The work proposed a two-phase algorithm composed of an heuristic and a CP model. It also considered individual tanks, intermittent pipelines, direct pipeline flow transmissions, as well as flow reversal operations at any pipeline. This work yielded feasible solutions for real instances and motivated a novel CP model for the same problem. In [3] the heuristic was enhanced using a CP approach composed of two sub-models. In addition, it considered further difficult constraints related to forbidden alignment sequences and simultaneous pumping operations. This article is an extension of these previous works, and also describes in full detail procedures considered therein.

4 Methodology

The procedure currently applied by PETROBRAS to operate its pipeline system is based on a *trial-and-error* process, in which manually constructed solutions are continuously tested and adjusted with the help of a proprietary simulator. This method is not only time and people demanding, but the lack of more detailed long-term foresight forces the solution to be rebuilt everyday by the pipeline operators.

Manual procedures are still used due to the great difficulty in solving the problem in its full setting, as mentioned in section 3. Consequently, decompositions methods are almost mandatory in order to cope with the problem size, a topic more fully discussed in [11]. We propose a novel decomposition method inspired on previous work [16] and on the manual procedures. This new framework is capable of dealing with larger and more complex topologies and can also accommodate additional constraints which are essential to guarantee the operational feasibility of solutions.

The problem is divided into two parts, the *planning phase* and the *scheduling phase*. The execution main cycle is schematically presented in figure 4. The planning phase decides the necessary volume transfers among depots. Each of such transfers must be scheduled in order to satisfy all demands, and distribute all volumes produced at refineries and so ensure an adequate stock level at each depot. Furthermore, the planning phase must also define the pipeline routes along which such transfers will occur, while minimizing traffic and possible product interfaces in each pipeline. Each transfer between depots is deemed a *delivery order*, a notion to be made precise shortly. The output of the planning phase is a set of delivery orders.

The scheduling phase represents the core decision process in our framework. Once the orders have been created, this phase must generate and sequence the necessary pumping operations so as to make it possible to carry out the complete set of orders. The scheduling phase must ensure that the orders are scheduled while observing all operational restrictions, such as tank capacities, forbidden sequences and interface constraints, as described in section 2.

The output of the scheduling phase is a complete solution to the problem, as detailed in section 2.4.

The planning phase takes a high level view of the pipeline network. The set of orders it generates in order to satisfy productions and demands involves relatively few scheduling details. This phase is similar to the current initial procedures used by field operators at PETROBRAS when constructing a manual solution. In this vein, heuristic techniques proved quite suitable to this planning phase. First of all, only a small subset of pipeline constraints are analyzed in this sub-problem, and previous works showed that heuristic approaches have a good performance in these cases [9, 14]. Moreover, when putting

such techniques to the test, one can take advantage of previous human expertise when designing their search criteria, thus enhancing the chances of finding a feasible solution. In this work we propose a randomized greedy heuristic to generate the set of delivery orders. This heuristic is described in section 4.1.

The scheduling phase demands a much higher level of detail. Investigation shows that modelling it as a simple MILP is not viable, since the number of variables and constraints would grow at an unacceptable rate when considering more complex network topologies. On the other hand, pure heuristic techniques were also greatly impaired when applied to the scheduling phase. Slight modifications in a solution can create serious collateral perturbations over the problem structure. For example, since products can flow directly from one pipeline to another, changing a single pumping start time may delay the arrival of a number of other products that pass through connected pipelines. Consequently, it is difficult to define a representation in which it is possible to execute a computationally affordable number of local modifications in order to find a viable solution.

On the other hand, the CP paradigm is well-suited to the scheduling phase. It provides a powerful modelling language that permits the implementation of operationally crucial constraints, besides providing enough flexibility to extend the model if new restrictions were deemed necessary by pipeline field operators. Most importantly, it allows the exploitation of special problem patterns explicitly. This is done, for instance, by modelling multiple sub-problem representations in order to use specialized and adequate constraint propagation mechanisms to solve each of the sub-problems. When modelling specific sub-problems one can take advantage of good representations proposed in previous works, such as the continuous time representation in [10]. In fact, such multiplicity of perspectives played an important role in the final model, greatly improving crucial domain reductions. Furthermore, a preliminary study [4] already indicated that CP would be flexible and powerful enough to treat the real problem faced by PETROBRAS. Finally, the use of CP was further fostered by its well-known good performance when treating scheduling problems [27]. In particular, at this stage of development, CP suited the application's needs due to its notable ability to find feasible solutions. The CP model for the scheduling phase is presented in detail in section 4.2.

4.1 Planning and Routing

The planning and routing phase essentially represents a global view of the problem. The main objective is to define the necessary product transmissions between depots, in a way that all demands are fulfilled and production volumes are stored without violating any inventory constraints. On the other hand, it does not define how the transmissions will be performed in an operational level, i.e., their sequence or schedule.

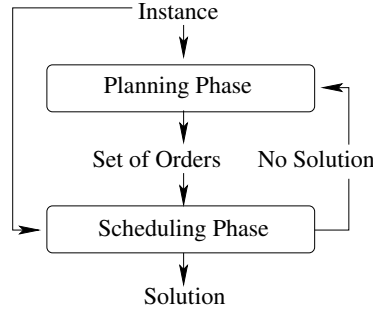


Figure 4: Solver Framework.

These transmissions between depots are formally modeled as structures called *delivery orders*, or *orders* for short. An order O represented by the tuple

$$O = (OriginTank, DestinationTank, Product, Volume, Route, Deadline)$$

enforces that some *Volume* of a *Product* must be extracted from the *OriginTank*. After that, it will travel through a *Route* that starts at the depot of the *OriginTank* and ends at the depot of the *DestinationTank*. At this last depot, the *Volume* will be inject into the *DestinationTank*. All of these operations must have ended before the *Deadline*. The planning and routing phase must generate a set of delivery orders as its main output. The input consists of both static and dynamic components of the problem, as described in section 2.4.

In this context, an order is considered to be *indivisible*, i.e., once a volume from an order starts to be pumped into its first route's segment, no volume from any other order can also be pumped into the segment while the first volume is not completely injected. Nevertheless, orders are not necessarily pumped in continuous streams, and, thus, several pumping operations might be used to pump an order into a pipeline, as long as it does not interleave with any other orders (see restriction 6, Subsection 2.2). This restriction was imposed in order to reduce product interface in the pipelines which, as said before, may result in some quality loss.

To generate orders, we created a randomized constructive heuristic that uses the accumulated human experience on the operation of the network at PETROBRAS. One of the purposes of the randomization is to produce diversified sets of orders in case the previous sets of orders were infeasible, increasing the chance of finding one or more solutions. Also, the heuristic takes into consideration other criteria that are difficult to handle manually, such as estimating the time for product volumes to arrive at depots.

Delivery orders are created incrementally. A group of functions evaluates values for the order properties, based on the input and previously created orders. This way, for each

order these functions will give different values as the network estimates changes. These estimates includes satisfied demand, pipelines, and tanks utilization. For each order its properties are determined as follows:

1. *Choosing Product, Destination Depot and Deadline:* For every triple of product p , destination depot bd and day t , define

$$V_{p,bd,t} = H - (LM(t) - TN_{p,bd}),$$

where H is the last minute in the planning horizon, $TN_{p,bd}$ is the minute when enough of product p is available in all depots that are able to deliver product p to depot bd in time to fully satisfy the demand at day t , and $LM(t)$ returns the last minute of day t . The $(LM(t) - TN_{p,bd})$ component gives an estimate of how much time there is left to satisfy the demand. All times are computed relative to the beginning of the planning horizon. We randomly select some triple (p, bd, t) within the best α_1 percent triples with the highest, *i.e.* most critical, $V_{p,bd,t}$ values. The idea behind this strategy is to satisfy any pressing demands first.

2. *Choosing Origin Depot and Route:* Given a selected triple of product p , destination depot bd and due date t , proceed as follows. For every pair of origin depot bo and route r starting at bo and ending at depot bd , define

$$V_{bo,r} = (1 - CG(r)) \times \frac{LM(t) - (TD_{p,bo} + Tmin_{r,p})}{(1 + NR_r)},$$

where $TD_{p,bo}$ is the minute when there is any positive quantity of the product p available at bo , and $Tmin_{r,p}$ gives the minimum amount of minutes for product p to travel along route r . The component $(TD_{p,bo} + Tmin_{r,p})$ gives the minimum time when one unit of product could have arrived at the destination depot. The term NR_r captures the number of such occurrences if route r is used and $CG(r)$ gives an estimated fraction of use for all the pipelines in route r , based on previously created orders. A higher number of possible necessary reversals or overloading of pipelines are unfavored. Considering all these factors, a pair among the best, *i.e.* largest, α_2 percent $V_{bo,r}$ values is randomly selected. Hence, the route with the greatest difference between earliest arrival and due times, for any volume occurring in it, will be selected. Note that this strategy also strives to minimize the number of flow reversals, a possible yet expensive maneuver.

3. *Choosing Origin, Destination Tank, and Volume:*

Given a destination depot bd , an origin depot bo and a product p , proceed as follows. For every pair of tanks to and td , both holding the product p and located at depots

bo and bd , respectively, let

$$V_{to,td,vol} = \left(1 - \max \{F_{to,bo,p}, F_{td,bd,p}\}\right) \times vol,$$

where $F_{t,b,p} = UR_t / MUR_{t,b,p}$. Here, $UR_t = TAlloc_t / Cap_t$ gives the fraction between the volume so far allocated to a tank t ($TAlloc_t$) and the tank capacity (Cap_t) *i.e.*, it measures how much a tank t has been used by force of previous orders, and the term $MUR_{t,b,p}$ is defined as the maximum of UR_t among all tanks t at depot b that can store product p . We choose the maximum fraction between the origin and destination tanks in order to capture the one most relatively used between them.

Now, by letting

$$vol = \min\{TQde_{to}, Cap_{td} - TQde_{td}\},$$

where $TQde_t$ is the volume residing at a tank t after all previous orders have been executed, we get the maximum volume that can be transferred from the origin tank to the destination tank.

We then select randomly within the α_3 percent tanks with largest $V_{to,td,vol}$ values. This gives a pair of tanks that can accommodate one of the largest volumes that can be transferred between the origin and destination depots.

As soon as there is no more demand to choose from, the planning phase terminates. After the procedure ends all demands are guaranteed to be satisfied if the resulting orders can be scheduled to arrive by their respective deadlines.

At this point, network operators can interfere in the planning of deliver orders by adding, modifying, or removing orders according to their particular needs. This flexibility is interesting since sudden needs might unexpectedly arise. For example, there might be exceptional cases where the operators want to empty certain tanks for emergency maintenance purposes. This could be achieved by issuing new orders that remove products from those particular tanks. Or the pipeline operators might dislike the proposed set of orders, for some unexpected reason.

Note that the whole planning phase can always be activated again. By doing so, the randomization built into the procedure will, most certainly, give rise to a different set of orders, for the same input data. In any case, the focus of the heuristic is to generate a set of orders that is feasible for the next scheduling phase, described in section 4.2.

Free Delivery Orders

In certain scenarios, the orders created by the planning phase are not enough to guarantee a valid solution. For instance, suppose that only two orders need to be scheduled along the

same route, and they have incompatible products. Since we can not push both products successively into a same pipeline, a third product must be inserted between them.

In order to cope with such situations, *free delivery orders* are created before entering the scheduling phase. In contrast to regular orders, their volumes, products, and origin or destination tanks are treated as variables instead of constants. Moreover, such free orders do not have a deadline parameter and may have a null volume associated with them. Furthermore, their routes are previously determined by choosing among those typically used by pipeline operators. Operators can change such routes by editing a configuration file.

Free orders are also used to represent products that remain in the pipeline at the end of the planning horizon in order to ensure that all pipes are always completely full. These orders do not have a destination tank, and constraints are used to indicate they are the last ones to be pumped into the pipelines. Further explanations about free delivery orders are given in section 4.5, which also considers a running example.

4.2 Scheduling Phase

The scheduling phase treats the problem of generating a number of pumping operations so as to satisfy a given set of delivery orders. These pumping operations must observe the network operational constraints, as explained in section 2. Otherwise, the solver must prove that the present set of orders is *infeasible*, *i.e.*, either the deadline of one or more orders cannot be met, or there is some operational constraint, such as tank capacities, would always be violated, for any possible schedule. In the latter case, a new set of orders can be requested from the planning phase.

Our research indicated that it would not be practical to create a single CP model to tackle the scheduling phase in its entirety. In a typical horizon of 5 to 10 days, the planning phase is expected to generate hundreds of orders. Such orders involve dozens of products that leave from and arrive at several tanks, circulate through many common pipelines, and are subjected to thousands of interrelated constraints. The resulting problem size would be excessively large as shown in [4], thus preventing the use of more realistic operational restrictions.

As a result, the scheduling phase was further decomposed in two sub-problems: the *sequencing* and the *scheduling sub-problems*, solved in succession. The sequencing sub-problem consists of defining the *ordering* according to which delivery orders will be satisfied at each pipeline and at each tank. This ordering must primarily observe interface and inventory constraints, as well as satisfy time bounds present in each delivery order. Once a valid ordering has been found, the scheduling sub-problem treats the problem of dividing each delivery order into smaller individual pumping operations, as well as as-

signing their respective start and end times. Such pumping operations must enforce the remaining scheduling restrictions, respecting the fixed ordering already established by a solution to the sequencing sub-problem.

The sequencing and scheduling sub-problems are described in sections 4.3 and 4.4, respectively. We note that each sub-problem considers a restricted set of constraints, although some constraints will be represented in the models designed for both sub-problems.

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

4.3 The Sequencing Sub-problem

The input for the sequencing sub-problem is the set of delivery orders and the problem static and dynamic components, described in section 2.4. A solution is an ordered list that defines the *sequence* of delivery orders to be satisfied at each tank and pipeline. This sequence must take into account ordering-related conditions, such as interface constraints, tank capacities, and flow direction in each route segment. Additionally, other essential operational restrictions are also treated, such as valid time bounds for the pumping operations.

The single CP model devised to solve this problem will be referred to as the *sequencing model*. Due to its complexity, more than one representation, or *viewpoint*, was used so that different problem restrictions could be more easily modelled. A viewpoint contains specific variables and constraints to deal with the corresponding restriction they focus on, as well as defines additional redundancy used to enforce constraint propagation [28]. The interconnection between viewpoints is done by defining *channelling variables* and posting additional *channelling constraints* [29]. In our model, it suffices to label only the channelling variables, as the constraint propagation in each viewpoint ensures that their variables are bound and, as a result, the corresponding restriction is satisfied.

Our model interrelates two different viewpoints. The *order viewpoint* provides a global view of the problem, dealing mainly with time and routing constraints for individual orders, and also with tank capacities. In contrast, the *operations viewpoint* captures a local view of the problem, representing only the operational behavior of each pipeline. Finally, the channelling variables indicate the proper sequencing of the orders in the corresponding route segments and also in the appropriate origin and destination tanks. Both viewpoints are inspired on the previous MILP formulations cited in section 3, and,

specially, in the pipeline structure presented in [10, 11, 20].

A notation for the input data and other specific details is described in section 4.3. The order viewpoint is detailed in section 4.3, while the operations viewpoint is described in section 4.3. Finally, channeling variables and constraints are presented in section 4.3. For the sake of clarity, the models are presented without considering free delivery orders. Section 4.3 describes the necessary changes to accommodate them, and Section 4.3 explains the search strategy that was used.

Some notation

Let $\mathbf{P}$ be the pipeline set, $\mathbf{T}$ be the tank set, and $\mathbf{O} = \{o_1, \dots, o_n\}$ be the set of delivery orders received from the planning phase. For each $o_i \in \mathbf{O}$, we define $\text{origin}(o_i) \in \mathbf{T}$ and $\text{destin}(o_i) \in \mathbf{T}$ as the origin and destination tanks for order o_i , respectively. Also, let $\text{route}(o_i) = (p_l, \dots, p_m)$ be the sequence of pipelines that o_i must traverse, *i.e.* its route. The intermediate depots are not considered in this notation, as products are stocked only in its destination tank. The volume corresponding to order o_i will follow a pre-defined flow *direction* in each pipeline $p \in \text{route}(o_i)$, given by the constant $\text{direction}_{i,p}$ which can be inferred from $\text{route}(o_i)$. We consider four possible flow directions in a pipeline: N , if the order follows the *normative*, or preferred, pipeline flow direction; R , if it follows the reverse direction; NR , if it starts in the normative direction, but later changes to the reverse direction, thus leaving the pipeline through the same extremity it was pumped into; and RN , similarly to NR , but with its flow starting in the reverse direction.

As explained in section 4.1, orders are indivisible. Furthermore, volumes must strictly follow the direction specified in the corresponding orders. For instance, if $\text{direction}_{i,p} = N$, then the volume corresponding to order o_i will never follow direction R once inside pipeline p . Hence, the volume corresponding to order o_i , once inside pipe p , must be pushed out of p before any volume that moves along the R direction in p can be injected into p . This also implies that volumes are never injected to or extracted from the same route segment more than once.

Let $\mathbf{Pr}$ and $\mathbf{Dem}$ be, respectively, the sets of productions and demands received from the planning phase, *i.e.* with their tanks already assigned. In this model, productions and demands are represented as delivery orders with routes designated as empty. Formally, $\mathbf{Pr}, \mathbf{Dem} \subseteq \mathbf{O}$ and for all o_i in $\mathbf{Pr} \cup \mathbf{Dem}$, we have $\text{route}(o_i) = \emptyset$. In addition, production orders have their destination tanks specified and their origin tanks left as empty. Demand orders have their origin tanks specified, and their destination tanks left as empty. In this way, the relations between production, demand and delivery orders can be handled in a transparent and uniform manner.

The Order Viewpoint

The order viewpoint handles the problem globally, focusing on constraints that define valid time bounds for each order. These bounds are related to maximum flow rates, disjunction of pipeline operations and tank capacities. If any of these time bounds violate an order's deadline or an inventory constraint, the delivery order set is clearly infeasible. Otherwise, they will help ruling out invalid order sequences.

We define the concept of an *activity* [30] before describing the model variables. An activity is a structure composed of two variables: a *start time* variable and a *duration* variable. For clarity, we will also consider that activities contain an *end time* variable, which is clearly the sum of its start time and its duration variables. A variable *var* of an activity *act* will be represented here as $Var(act)$.

Due to the indivisibility and strict flow direction properties, each order will be represented by two activities for each segment of its route, symbolizing the time intervals in which the order occupies the extreme points of the pipeline. Formally, for each $o_i \in \mathbf{O}$ and $p \in \mathbf{route}(o_i)$, we define two activities, $snd_{i,p}$ and $rcv_{i,p}$. The first activity represents the time interval during which the volume from order o_i was pumped into p , while the second represents the time interval during which the volume from order o_i was pumped out of p . In this way, we have explicit variables that describe the times when an order's volume is being injected in and extracted from a pipeline. This representation is depicted in figure 5.

Note that such activities do not necessarily represent continuous operations. In other words, the start time represents the initial instant in which the first volume unit was pumped into, or out of, the pipeline and, similarly, the end time is the instant in which the last volume unit was pumped into, or out of, the pipeline. Therefore, it is possible to create more than one pumping operation for each activity, as long as their start and end times observe the time interval imposed by the activity. This concept accommodates for the possibility of an intermittent behavior during the execution of an individual order. As mentioned earlier, breaking activities into pumping operations is not dealt with at this stage.

Simple linear constraints can be used to ensure bounds on flow rates and to state delivery deadline constraints. There is no explicit variables for flow rates, since we are only interested in a lower bound for the duration of activities (restriction 7 in section 2). Let $\mathbf{max_flow_rate}_{p,d,pr}$ be a constant representing the maximum flow rate in pipeline p , for direction d , and traversing product pr . For each $o_i \in \mathbf{O}$ and each $p \in \mathbf{route}(o_i)$, we

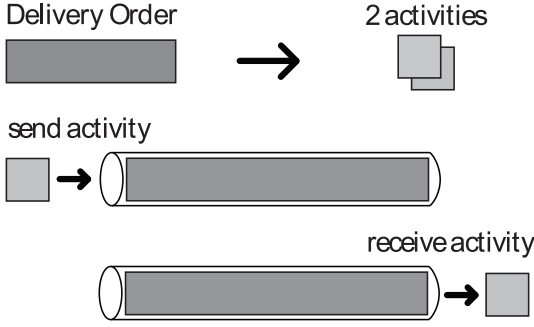


Figure 5: Activities for each pipeline in a delivery order's route.

let:

$$EndTime(rcv_{i,p}) \leq \text{deadline}(o_i), \quad (1)$$

$$Duration(snd_{i,p}) \times \text{max_flow_rate}_{p,\text{direction}_{i,p},\text{product}(o_i)} \geq \text{volume}(o_i), \quad (2)$$

$$Duration(rcv_{i,p}) \times \text{max_flow_rate}_{p,\text{direction}_{i,p},\text{product}(o_i)} \geq \text{volume}(o_i). \quad (3)$$

Before an order exits a pipeline, it must first traverse the full pipeline extension. Therefore, for a given order, the time elapsed between its send and receive activities in a pipeline must be at least the time necessary for its volume to traverse the full pipeline extension at the maximum flow rate. Let $\text{product}(o_i)$ be the product associated with order o_i . For each $o_i \in \mathbf{O}$ and $p \in \text{route}(o_i)$, we require:

$$StartTime(rcv_{i,p}) \geq StartTime(snd_{i,p}) + \left\lfloor \frac{\text{volume}(p)}{\text{max_flow_rate}_{p,\text{direction}_{i,p},\text{product}(o_i)}} \right\rfloor. \quad (4)$$

By the route definition, when an order is being pumped out of one of its intermediate segments it must be immediately pumped into its next segment, and without volume loss. Equivalently, while a *receive* activity is being performed in an intermediate segment, a *send* activity must be performed in the subsequent segment at the same time. This relationship can be easily modelled with the activity variables, as illustrated in figure 6. It consists of *unifying* [29] consecutive send and receive activity variables in the following way. For each $o_i \in \mathbf{O}$ and for each pair (p_l, p_{l+1}) of consecutive pipelines in $\text{route}(o_i)$, simply let:

$$StartTime(rcv_{i,p_l}) = StartTime(snd_{i,p_{l+1}}), \quad (5)$$

$$EndTime(rcv_{i,p_l}) = EndTime(snd_{i,p_{l+1}}). \quad (6)$$

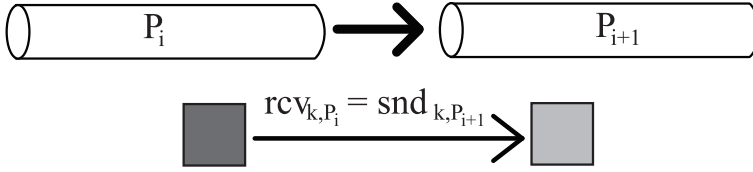


Figure 6: Connection between two consecutive pipelines in an order's route.

Order activities related to a same pipeline must all be disjunctive with respect to time (restriction 7 in section 2). That is, a send (or receive) activity from a certain order must not overlap with the send (or receive) activities of other orders that share common pipelines. In order to guarantee this, we use the global constraint *disjunctive*. For all $p \in \mathbf{P}$, let

$$\mathbf{disjunctive}(\{snd_{i,p} : \forall o_i \in \mathbf{O}, p \in \mathbf{route}(o_i)\}), \quad (7)$$

$$\mathbf{disjunctive}(\{rcv_{i,p} : \forall o_i \in \mathbf{O}, p \in \mathbf{route}(o_i)\}). \quad (8)$$

Next, we define two new activities: ext_i and inj_i , representing volume extraction and injection, respectively, at the origin tank $\mathbf{origin}(o_i)$ and destination tank $\mathbf{destin}(o_i)$ associated with order o_i . Volumes injected into and extracted from tanks must not overlap in time as well. For all $t \in \mathbf{T}$, let

$$\mathbf{disjunctive}(\{ext_i : \forall o_i \in \mathbf{O}, \mathbf{origin}(o_i) = t\} \cup \{inj_i : \forall o_i \in \mathbf{O}, \mathbf{destin}(o_i) = t\}). \quad (9)$$

Production and demand orders must occur within their corresponding time frames, satisfying the inventory constraints described in section 2.3. For each $o_i \in \mathbf{Pr}$, let

$$StartTime(inj_i) \geq \mathbf{ProductMinStartTime}(o_i), \quad (10)$$

$$EndTime(inj_i) \leq \mathbf{ProductMaxEndTime}(o_i), \quad (11)$$

and for each $o_i \in \mathbf{Dem}$, post

$$StartTime(ext_i) \geq \mathbf{DemandMinStartTime}(o_i), \quad (12)$$

$$EndTime(ext_i) \leq \mathbf{DemandMaxEndTime}(o_i). \quad (13)$$

But note that tank capacities must also be taken into account. This is achieved by using a global constraint called a *reservoir* [30]. Reservoirs are resources that contain a minimum, a maximum, and an initial level. Activities can both increase the level (volume injection) or decrease the level (volume extraction) from reservoirs over time. These levels can be used to implement a tank's initial volume as well as its capacity. A link

between activities and reservoirs is defined using the keywords *consumes(capacity)* and *produces(capacity)*. For a reservoir $TkRes_t$, where $t \in \mathbf{T}$, we must have for each $o_i \in \mathbf{O}$:

$$ext_i \quad \mathbf{consumes}(\mathbf{volume}(o_i)) \quad TkRes_{\mathbf{origin}(o_i)}, \quad (14)$$

$$inj_i \quad \mathbf{produces}(\mathbf{volume}(o_i)) \quad TkRes_{\mathbf{destin}(o_i)}. \quad (15)$$

Finally, the relationship between send (receive) variables and tank activities is analogous to those for pipeline volume transmissions. While an order's volume is being extracted from a tank, it must be immediately pumped into its first route segment. Similarly, a volume is immediately injected into its destination tank while it is being pumped out of its last route segment. For each $o_i \in \mathbf{O}$, letting p_0 and p_m be the first and last pipeline along $\mathbf{route}(o_i)$, respectively, we state:

$$StartTime(snd_{i,p_0}) = StartTime(ext_i), \quad (16)$$

$$EndTime(snd_{i,p_0}) = EndTime(ext_i), \quad (17)$$

$$StartTime(rcv_{i,p_m}) = StartTime(inj_i), \quad (18)$$

$$EndTime(rcv_{i,p_m}) = EndTime(inj_i). \quad (19)$$

In summary, the constraints posed in the order viewpoint ensure the time consistency of orders, force deadline satisfactions, and also guarantee that operations at each pipeline and at each tank are always disjunctive. The inventory constraints are enforced in this viewpoint as well.

The Operations Viewpoint

The main intuition for this viewpoint relates to the rules for volume injection and extraction at pipeline extremities. Although some time bounds and disjunctions were established in the order viewpoint, it is still necessary to model the fact that, for a certain volume to leave a pipeline, the same volume must be pumped into the other extremity. This restriction imposes precedence relations between orders, and also further restricts activity time bounds. To capture these conditions, we state additional restrictions among the *send* and *receive* activities of different orders. These restrictions, basically, impose that a *send* activity is performed if and only if a *receive* activity occurs at the exact same time. Restrictions such as variable flow rates, which depend on the sequence of products inside a pipeline must also be considered. The model that implements these relations is described next.

We introduce a new activity structure for this viewpoint by defining *operation activities*. Besides start time, end time and duration variables, an operation activity also has other variables associated with it, namely a *volume*, a *product*, a *direction*, and an

order variable. Operation activities will be *bound* to orders: the order variable can be seen as a pointer to the corresponding order in the $\mathbf{O}$ set, while the other operation order variables will represent values for the remaining order constants (the order's product, volume, directions, and so on). Moreover, operation activities will be created in a way that accommodates a more intuitive and compact representation of a pipeline operational behavior.

For each pipeline $p \in P$, we define two sequences of operation activities, $SndPipe_p$ and $RcvPipe_p$. The i -th element of the sequence $SndPipe_p$ is denoted by $sndOp_{i,p}$ and, similarly, $rcvOp_{i,p}$ denotes the i -th element of the sequence $RcvPipe_p$. The size of each sequence is the number of orders that traverse the corresponding pipeline, that is, $|SndPipe_p| = |RcvPipe_p| = |\{o_i : o_i \in \mathbf{O}, p \in \mathbf{route}(o_i)\}|$. Moreover, we consider that both sequences are *time ordered*, as enforced by the following constraints.

$$\begin{aligned} StartTime(sndOp_{i,p}) &\geq EndTime(sndOp_{j,p}), \\ &\forall sndOp_{i,p}, sndOp_{j,p} \in SndPipe_p, i > j, \end{aligned} \quad (20)$$

$$\begin{aligned} StartTime(rcvOp_{i,p}) &\geq EndTime(rcvOp_{j,p}), \\ &\forall rcvOp_{i,p}, rcvOp_{j,p} \in RcvPipe_p, i > j. \end{aligned} \quad (21)$$

The intended meaning of the operation activities in $SndPipe_p$ and in $RcvPipe_p$ is similar to the *send* and *receive* activities in the order viewpoint. Their timing variables represent the period of time during which the corresponding volume is occupying the extreme points of a pipeline. That is, the time variables of an activity $sndOp \in SndPipe_p$ represent the time interval during which the volume specified by its *volume* variable is being injected into p . In the same vein, the time variables of an activity $rcvOp \in RcvPipe_p$ give the time interval during which the volume, specified by the corresponding *volume* variable, is being extracted from p . Note that constraints (20) and (21) already enforce the precedence relation between such orders, *i.e.* the volume corresponding to operation activity $sndOp_{i,p}$ will necessarily be pumped into p before the volume corresponding to activity $sndOp_{j,p}$, if $i < j$. Similarly for the operation activities in a $RcvPipe_p$ sequence. As a result, by binding orders to operation activities at each pipeline, we can enforce the pipeline order sequences.

For instance, consider the example showed in the figure 7. Suppose we have six orders, labeled A to F , all with volumes traversing the pipeline as depicted in the example. The $SndPipe_p$ and $RcvPipe_p$ sequences will have each 6 operation activities, labeled 1 to 6. As order A is bound to send operation activity 1 and B is bound to activity 2, order A will have its volume pumped first into the pipeline. In a similar way, order C , bound to the receive operation activity 5, will have its volume pumped out first than will order D , which is bound to receive activity 4.

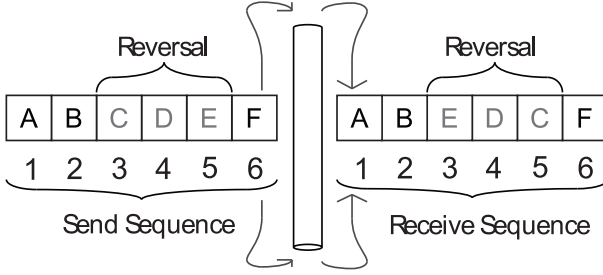


Figure 7: Case when the *send* and *receive* activity sequences are different.

Most importantly, note that a $SndPipe_p$ sequence is *not* necessarily the same as a $RcvPipe_p$ sequence, due to the fact that the flow direction in a pipeline may change. In the previous example, if activities C , D and E indicate flow reversals, *i.e.* they flow along directions NR or RN , then they will be pumped out of the pipeline in reverse order. See also figure 3.

Using the previous representations for operation activities we can model pipeline flow direction, interface constraints, and the pipeline operational behavior. For that, we make use of a special structure, called a *pushing graph*, to be detailed next.

Flow Direction. Flow directions in a pipeline must be consistent. For instance, if an activity has its direction attribute set to N , then the next activity along the same pipeline must necessarily have its direction attributes set to N or NR . Direction attributes such as R and RN are only consistent after a sequence of NR activities whose total volume is equal to the pipeline volume, as required by restriction 10 in section 2. Attribute RN is treated in a similar way.

In order to ensure flow direction consistency, we use the global constraint *Table Constraint* [30]. First, we define a set of pairs that represent valid directions for any two consecutive activities. In our case, the set is

$$\text{ValidPairs} = \{ (N, N), (R, R), (NR, NR), (RN, RN), (N, NR), (NR, R), \\ (R, RN), (RN, N), (NR, RN), (RN, NR) \}.$$

Next, we invoke table constraints over the direction variables in order to enforce that only valid pairs of consecutive operation activities can occur. For each $p \in \mathbf{P}$ and each $i \in \{1, \dots, |SndPipe_p| - 1\}$, we post

$$\text{table_constraint} \left(\text{Direction}(sndOp_{i,p}), \text{Direction}(sndOp_{i+1,p}), \text{ValidPairs} \right), \quad (22)$$

$$\text{table_constraint} \left(\text{Direction}(rcvOp_{i,p}), \text{Direction}(rcvOp_{i+1,p}), \text{ValidPairs} \right). \quad (23)$$

As for pipeline reversals, a special global constraint **reversal** was created, encapsulating the rules for flow direction reversal. It ensures that maximum sequences of NR or RN operations have their total volume sum equal to the corresponding pipeline volume. It will also ensure the proper sequence of send and receive orders during flow reversals, as illustrated in figure 7.

Due to space considerations, this global constraint will not be described in full detail. Instead, we give intuition on the general mechanism of such procedures. For each operation activity $op \in SndPipe_p \cup RcvPipe_p$, a new integer variable, called a *group*, is created. We state that two or more variables have the same value for the group variable if and only if they are part of the same maximum sequences of NR or RN operations. This property is implemented for the sequences $SndPipe_p$ and $RcvPipe_p$ separately. For each $p \in \mathbf{P}$ and each $i \in \{1, \dots, |SndPipe_p| - 1\}$,

$$\begin{aligned} Group(sndOp_{i,p}) = Group(sndOp_{i+1,p}) &\iff Direction(sndOp_{i,p}) = Direction(sndOp_{i+1,p}) \\ &\quad \wedge Direction(sndOp_{i,p}) \in \{RN, NR\}, \end{aligned} \quad (24)$$

$$\begin{aligned} Group(rcvOp_{i,p}) = Group(rcvOp_{i+1,p}) &\iff Direction(rcvOp_{i,p}) = Direction(rcvOp_{i+1,p}) \\ &\quad \wedge Direction(rcvOp_{i,p}) \in \{RN, NR\}, \end{aligned} \quad (25)$$

$$Group(sndOp_{i,p}) \leq Group(sndOp_{i+1,p}), \quad (26)$$

$$Group(sndOp_{i,p}) \leq Group(rcvOp_{i+1,p}). \quad (27)$$

The following conditions, implemented using *set constraints* [30], ensure that the maximum sequences must sum to the pipeline volume. Let $G_k = \{op_{i,p} : Group(op_{i,p}) = k, Direction(op_{i,p}) \in \{RN, NR\}\}$. For each $p \in \mathbf{P}$ and each possible value k for the group variable,

$$\sum_{sndOp_{i,p} \in G_k \cap SndPipe_p} Volume(sndOp_{i,p}) = Volume(p), \quad (28)$$

$$\sum_{rcvOp_{i,p} \in G_k \cap RcvPipe_p} Volume(rcvOp_{i,p}) = Volume(p). \quad (29)$$

The group variables are also used to ensure the correct inverse bind sequence in flow reversals. First, it is necessary to guarantee that a send and receive operation activity have the same group value if and only if they are bound to the same order. For each $p \in \mathbf{P}$ and each distinct $i, j \in \{1, \dots, |SndPipe_p|\}$,

$$Order(sndOp_{i,p}) = Order(rcvOp_{j,p}) \iff Group(sndOp_{i,p}) = Group(rcvOp_{j,p}). \quad (30)$$

If two send activities have the same group variable value, then the receive binding

sequence is the inverse of the send bind sequence. Thus, we have to impose the following conditions, for each $p \in \mathbf{P}$ and each $i, j \in \{1, \dots, |SndPipe_p|\}$.

$$\begin{aligned} & Group(sndOp_{i,p}) = Group(sndOp_{j,p}) \wedge Direction(sndOp_{i,p}) \in \{RN, NR\} \iff (31) \\ & \exists k < l \text{ s.t. } Order(rcvOp_{k,p}) = Order(sndOp_{j,p}) \wedge Order(rcvOp_{l,p}) = Order(sndOp_{i,p}). \end{aligned}$$

The global constraint **reversal** implements condition 31 in a more compact and efficient way. Furthermore, additional conditions are posted so as to enforce propagation, thus linking the channelling variables discussed in section 4.3 with the above flow direction restrictions.

Interface Constraints. The interface constraint (see restriction 8 of section 2) requires that we identify which volumes are in contact inside a pipeline. Contact between volumes corresponding to consecutive orders in a time ordered sequence can occur in two scenarios. First, if the orders specify the same pipeline flow direction. Secondly, if a reversal is about to begin (in a send operation activity sequence) or is about to end (in a receive operation activity sequence). Note that, in the latter case, the pipeline ordering will be inverted. Thus, if **CompatiblePairs** is a set of compatible product pairs, the interface constraints can be enforced as follows. For each pipeline p and each $i \in \{1, \dots, |SndOp_p| - 1\}$,

$$\begin{aligned} & Direction(sndOp_{i,p}) = Direction(sndOp_{i+1,p}) \vee (32) \\ & \left(Direction(sndOp_{i,p}), Direction(sndOp_{i+1,p}) \right) \in \{(N, NR), (R, RN), (NR, RN), (RN, NR)\} \\ & \implies \mathbf{table_constraint} \left(Product(sndOp_{i,p}), Product(sndOp_{i+1,p}), \mathbf{CompatiblePairs} \right), \\ & Direction(rcvOp_{i,p}) = Direction(rcvOp_{i+1,p}) \vee (33) \\ & \left(Direction(rcvOp_{i,p}), Direction(rcvOp_{i+1,p}) \right) \in \{(NR, R), (RN, N), (NR, RN), (RN, NR)\} \\ & \implies \mathbf{table_constraint} \left(Product(rcvOp_{i,p}), Product(rcvOp_{i+1,p}), \mathbf{CompatiblePairs} \right). \end{aligned}$$

The Pushing Graph. The main idea of the operations viewpoint is to take advantage of time ordered sequences for the purpose of modelling the pushing of volumes inside pipelines. To clarify, let $rcvOp_{j,p} \in RcvPipe_p$. The volume associated with $rcvOp_{j,p}$ can only be received when an activity $sndOp_{i,p} \in SndPipe_p$ is being simultaneously injected at the other extremity of pipe p . As such, we say that activity $sndOp_{i,p}$ *pushes* activity $rcvOp_{j,p}$ out of p . As a consequence, the intersection between the corresponding time intervals must not be empty. In order to correctly establish this relationship, a special structure, called a *pushing graph*, is introduced. It largely explores the pre-defined precedences between operation activities in both sending and receiving sequences.

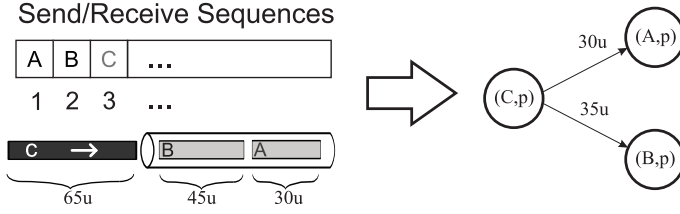


Figure 8: Example of edges in the pushing graph for a single pipeline.

A pushing graph is a digraph $G = (V, E)$ where each vertex $v \in V \subseteq \mathbf{O} \times \mathbf{P}$ represents an order $o \in \mathbf{O}$ and a pipeline $p \in \mathbf{P}$. For simplicity, we will use the notation $v = (o, p)$ for vertices. Clearly, we can only have a vertex $v = (o, p)$ if $p \in \text{route}(o)$. Now, let $v = (o_i, p_i)$ and $w = (o_j, p_m)$. In the pushing graph, there is an edge $(v, w) \in E$ if and only if an operation activity sndOp_{x,p_i} , bound to order o_i , pushes an operation activity rcvOp_{y,p_m} , bound to order o_j , out of pipeline p_m . From now on, we will use the notation $v \rightarrow w$ when $(v, w) \in E$.

Figure 8 shows an example of a pushing graph for a network composed of a single pipeline p having a volume of 75 units. In the figure, orders A , B , and C need to be sequenced, where $\text{volume}(A) = 30$ units, $\text{volume}(B) = 45$ units, and $\text{volume}(C) = 65$ units. Three vertices are thus created: (A, p) , (B, p) , and (C, p) . Suppose that both the sending and receiving operation sequences are exactly A , B , and C . After activities $\text{sndOp}_{1,p}$ and $\text{sndOp}_{2,p}$ are executed, we can infer that the pipeline is entirely filled with the volumes corresponding to orders A and B . As such, when $\text{sndOp}_{3,p}$ is executed, the volume associated to order C will push out of the pipeline a certain amount of the volumes corresponding to orders A and B . This shows that there is a time relationship between activity $\text{sndOp}_{1,p}$ and the activities $\text{rcvOp}_{1,p}$ and $\text{rcvOp}_{2,p}$. Therefore, a direct edge from vertex (C, p) to vertices (A, p) and (B, p) is inserted in the pushing graph. Furthermore, it is also possible to calculate the actual volumes associated with each of these edges, a volume that will be pushed out of the pipeline when activity $\text{sndOp}_{3,p}$ is executed. As the volume from order A leaves the pipeline first, it will consume 30 units from order C . The remaining volume associated to order C namely, 35 units, will push out of the pipeline part of the volume associated with order B . In the figure, these numbers are represented above the corresponding edges.

The pushing graph is initialized with the complete set of vertices and by letting $E = \emptyset$. The edges are dynamically created by monitoring the corresponding volumes and variable domains when operation activity bindings occur during the labelling process. This relation is bidirectional: if new edges are added, a propagation mechanism is invoked to update the variable domains accordingly.

The dynamic update of the pushing graph structure occurs as follows. Let $p \in \mathbf{P}$

be a pipeline. A new variable $acc_{i,j,p}$ is created for each operation activities $sndOp_{i,p} \in SndPipe_p$ and $rcvOp_{j,p} \in RcvPipe_p$, where $i \geq j$. It represents the volume in p *before* inserting the total volume associated to activity $sndOp_{i,p}$ and *after* extracting the total volume corresponding to $rcvOp_{j,p}$. This volume is known as the *accumulated* value between the activities. For instance, $acc_{3,1,p} = 35$ units in the example depicted in figure 8.

As the send and receive sequences are time-ordered, it can be shown that $sndOp_{i,p}$ never pushes $rcvOp_{j,p}$ out of p , if $i < j$. Thus, for each $i \geq j$ the $acc_{i,j,p}$ variables can be represented by simple constraints. We let:

$$acc_{i,j,p} = \sum_{k \leq j} volume(rcvOp_{k,p}) - \sum_{k < i} volume(sndOp_{k,p}). \quad (34)$$

If $acc_{i,j,p} \geq volume(p)$, then it is not possible for the volume associated with $sndOp_{i,p}$ to push the volume corresponding to $rcvOp_{j,p}$ inside p since a quantity greater than or equal to the pipeline volume was already injected in the pipeline between both such volumes. On the other hand, if $acc_{i,j,p} + volume(sndOp_{i,p}) + volume(rcvOp_{j,p})$ is at most $volume(p)$, then the volume in $sndOp_{i,p}$ does not suffice to completely push the volume of $rcvOp_{j,p}$ out of the pipeline. We state the following edge-existence constraints:

$$\begin{aligned} \exists \text{ edge } (Order(sndOp_{i,p}), p) \rightarrow (Order(rcvOp_{j,p}), p) & \quad (35) \\ \iff \\ [acc_{i,j,p} < volume(p)] \wedge [acc_{i,j,p} + volume(sndOp_{i,p}) + volume(rcvOp_{j,p}) > volume(p)] & \end{aligned}$$

We also define a function $pvol : E \rightarrow \mathbb{R}_+$. For an edge $(o_i, p_i) \rightarrow (o_j, p_j)$, $pvol$ represents the volume of order o_i in pipeline p_i that was effectively used to push the volume corresponding to order o_j out of pipeline p_j . In figure 8, $pvol(e)$ is the number above edge e . Note that such volume is not necessarily the total volume of one of the orders; o_j can push several other orders out of a certain pipeline and, similarly, o_j can be pushed out of a pipeline by several other orders as well. The function $pvol$ can be calculated by the following relation, where $e = ((Order(sndOp_{i,p}), p) \rightarrow (Order(rcvOp_{j,p}), p))$:

$$\begin{aligned} pvol(e) = \min \Big\{ & volume(rcvOp_{j,p}), volume(rcvOp_{j,p}) \\ & + volume(sndOp_{i,p}) + acc_{i,j,p} - volume(p) \Big\} \\ & - \max \Big\{ 0, volume(rcvOp_{j,p}) + acc_{i,j,p} - volume(p) \Big\}. \end{aligned} \quad (36)$$

Using constraints (34), (35), and (36), extended for all $p \in \mathbf{P}$ and their operational activities, the pushing relations at individual pipelines can all be enforced. A simple

procedure is used to guarantee edge creation in the pushing graph, thus providing a *global* view of all pushing operations. It is described next.

First, we define two new functions $invol : E \rightarrow \mathbb{R}$ and $outvol : E \rightarrow \mathbb{R}$. Take an edge $e = (o_i, p_i) \rightarrow (o_j, p_j)$. The value $invol(e)$ represents the volume of order o_i that has already been injected into pipeline p_i *immediately before* it starts pushing the volume corresponding to order o_j out of pipeline p_j . Similarly, the value $outvol(e)$ represents the volume of order o_j that has already left the pipeline p_j *immediately before* the volume associated with order o_i starts to push it out of p_j . For instance, in figure 8, let $e = (C, p) \rightarrow (B, p)$. Then, we have $invol(e) = 30$ and $outvol(e) = 0$. The $invol$ and $outvol$ functions can be easily calculated by changing equation (36) appropriately.

For clarity, we define two similar functions: $afterinvol$ and $afteroutvol$. The first represents the volume of order o_i that has been injected into pipeline p_i *after* it pushed volume $pvol(e)$ of o_j out of pipeline p_j . The second represents the volume of order o_j that has left pipeline p_j *after* the volume $pvol(e)$ was pushed out of p_j by the volume associated with order o_i . Clearly, $afterinvol(e) = invol(e) + pvol(e)$ and $afteroutvol(e) = outvol(e) + pvol(e)$. In figure 8, if $e = (C, p) \rightarrow (B, p)$, then we have $afterinvol(e) = 65$ and $afteroutvol(e) = 35$.

Let p_m, p_{m+1} be two consecutive pipelines along the route of an order o . Also, let $v, w \in V$, and with edges $v \rightarrow (o, p_m)$ and $(o, p_{m+1}) \rightarrow w$ in E . In direct volume transmissions between pipelines, a situation enforced by constraints (5) and (6), we can post

$$\begin{aligned} & \left[outvol(v \rightarrow (o, p_m)), afteroutvol(v \rightarrow (o, p_m)) \right] \\ & \cap \left[invol((o, p_{m+1}) \rightarrow w), afterinvol((o, p_{m+1}) \rightarrow w) \right] \neq \emptyset \\ \implies & v \rightarrow w. \end{aligned} \tag{37}$$

The pushing relations are *transitive* along a route, provided there is an intersection among the $invol$ and $outvol$ intervals. This transitivity is a mathematical view of the following fact. While a volume associated with v is pushing a certain volume of o out of pipeline p_m , this same amount of volume is being directly transmitted to pipeline p_{m+1} . As such, it will also push some volume out of pipeline p_{m+1} , corresponding to order w in our example. Therefore, a volume corresponding to v in pipeline p_m is pushing a volume associated with w in pipeline p_{m+1} . So, an edge must be created between vertices v and w . Hence, every time a new edge is dynamically inserted in G , the propagation mechanism will try to add new edges using condition (38), thus defining a global precedence relation between pipelines and their activities. The $pvol$ of the new edge will be the minimum of the $pvol$ of both previous edges. We post the following constraint when a new edge e is added to the graph of a pipeline $p \in \mathbf{P}$, to state the time dependency between activities

$sndOp_{p,i}$ and $rcvOp_{p,i}$, both related to edge e :

$$\left[StartTime(sndOp_{i,p}), EndTime(sndOp_{i,p}) \right] \cap \left[StartTime(rcvOp_{j,p}), EndTime(rcvOp_{j,p}) \right] \neq \emptyset. \quad (38)$$

Also, let $MaxFl_i$ be a variable representing the maximum flow rate for activity $sndOp_{p,i}$, related to variable $Product(sndOp_{p,i})$, and let $MaxFl_j$ stand similarly for activity $rcvOp_{p,j}$. We state:

$$\left\lfloor \frac{volume(sndOp_{i,p})}{EndTime(sndOp_{i,p}) - StartTime(sndOp_{i,p})} \right\rfloor \leq MaxFl_j \quad (39)$$

$$\left\lfloor \frac{volume(rcvOp_{j,p})}{EndTime(rcvOp_{j,p}) - StartTime(rcvOp_{j,p})} \right\rfloor \leq MaxFl_i. \quad (40)$$

The pushing graph is capable of ensuring complex precedence relations, as it provides a global view of the pumping conditions of all activities. Moreover, it is clear that it can be seen as a network flow graph as well: the *pvol* sum for the edges that push the volume of an order o in a pipeline p must be equal to the *pvol* sum for the edges representing the orders pushed by the volume of o . The value of this sum is precisely $\mathbf{volume}(o)$. Clearly, then, in order to enhance domain reductions, we can use a **flow** global constraint [27] involving sequences of send and receive variables.

Channeling Variables and Constraints

Finally, it is necessary to bind orders to operation activities, thereby connecting the order viewpoint to the operations viewpoint. This can be done by defining positional variables $sndPos_{i,p}$ and $rcvPos_{i,p}$, accounting for the positions of, respectively, the send and receive activities of order o_i at pipeline $p \in \mathbf{route}(o_i)$.

First, two different orders can not be associated to the same pipeline position. We use *alldifferent* global constraints [29] to enforce this restriction. For each $p \in \mathbf{P}$, we let

$$\mathbf{all_diff}(\{sndPos_{i,p} : o_i \in \mathbf{O}, p \in \mathbf{route}(o_i)\}), \quad (41)$$

$$\mathbf{all_diff}(\{rcvPos_{i,p} : o_i \in \mathbf{O}, p \in \mathbf{route}(o_i)\}). \quad (42)$$

The order and operation viewpoints can be easily connected using the *element* constraint [29] together with positional variables, as follows. For an element constraint $z = x_y$, where x, y are variables and z is either a constant or variable, we use the notation

$\text{element}(z, x, y)$. We state

$$\text{element}(\text{StartTime}(\text{snd}_{i,p}), \text{StartTime}(\text{sndOp}_{k,p}), k = \text{sndPos}_{i,p}), \quad (43)$$

$$\text{element}(\text{EndTime}(\text{snd}_{i,p}), \text{EndTime}(\text{sndOp}_{k,p}), k = \text{sndPos}_{i,p}), \quad (44)$$

and, for all $o_i \in \mathbf{O}$ and all $p \in \text{route}(o_i)$, we state further:

$$\text{element}(o_i, \text{Order}(\text{sndOp}_{k,p}), k = \text{sndPos}_{i,p}), \quad (45)$$

$$\text{element}(\text{volume}(o_i), \text{Volume}(\text{sndOp}_{k,p}), k = \text{sndPos}_{i,p}), \quad (46)$$

$$\text{element}(\text{direction}(o_i), \text{Direction}(\text{sndOp}_{k,p}), k = \text{sndPos}_{i,p}), \quad (47)$$

$$\text{element}(\text{product}(o_i), \text{Product}(\text{sndOp}_{k,p}), k = \text{sndPos}_{i,p}). \quad (48)$$

A similar set of constraints is applied to the receive sequences.

It is also necessary to enforce $\text{sndPos}_{i,p} = \text{rcvPos}_{i,p}$ if and only if $\text{direction}(o_i) \in \{N, R\}$, that is, the order is not related to any pipeline flow reversal. The other cases are handled by the global constraint **reversal**, described in section 4.3. At last, we note there are positional variables for sequencing orders in tanks as well, which are modelled in a similarly way. But note that, as tanks do not involve operation activities, the corresponding restrictions only involve variables from the order viewpoint model. Since they are simpler and declared analogously, they will be omitted due to space considerations.

Free Delivery Orders

Only minor changes to the previous modelling are necessary in order to handle free orders. Among them, in the order viewpoint, volumes and products should be changed from constants to variables in all constraints. Also, *Alternative Resource Set* [30] constraints can be used to guarantee that origin and destination tanks are assigned to all free orders. The operations viewpoint model remains unchanged.

Search Strategy

The search strategy refers to the algorithms responsible for finding a feasible solution to the CP problem. In the sequencing model, a solution is found by labelling only the positional variables. The constraint propagations in the order and operations viewpoint models ensure the satisfaction of the problem restrictions. Also, time variables do not need to be labelled since we are only interested in valid time bounds for order satisfaction, given by the minimum and maximum values of the resulting domains.

Different types of search strategies were tested in order to solve the sequencing model. The currently implemented version is shown in Algorithm 2. It combines a *backtracking*

mechanism [29] with a special variable ordering strategy. The algorithm has two consecutive parts: *disjunctive components determination* and *adaptive backtracking*. More details about each part now follow.

Algoritmo 2 Procedure for search strategy

```

1: procedimento
2:   Identify network disjunctive components  $C$ 
3:   Para each  $c \in C$  faça
4:     Build resource graph  $G(c)$  and sort it topologically, obtaining order  $N$ 
5:      $N' := N$ ;  $k := \text{initial\_}k$ 
6:     enquanto  $N' \neq \emptyset$  faça
7:        $p :=$  first element from sequence  $N'$ ;  $N' := N' / \{p\}$ 
8:       Label positional variables, and volumes/tanks in case of free orders
9:       se fails in labeling  $\geq k$  and not cyclic condition então
10:         $k := k + \text{incremental factor}$ ;  $N' := N$ 
11:        Move  $p$  to the beginning of sequence  $N'$ 
12:       fim se
13:     fim enquanto
14:   fim Para
15: fim procedimento

```

Disjunctive Components Determination. A disjunctive component is defined as a subset of the network which can be sequenced separately, without affecting other regions. Two pipelines belong to the same component if they are both contained in a route associated to some order. The same reasoning applies to tanks. Therefore, it is possible to deal with smaller search spaces, as each component has its own backtrack tree.

The disjunctive component determination also gives some room for parallelism during execution, although it was not implemented in the current experiments.

Adaptive Backtracking. We implemented backtracking using the positional variables for each pipeline and each tank. The term *adaptive* stems from the fact that we used a *restart* strategy [31].

The labelling process is done for each pipeline and each tank separately, both treated as *resources*. Given an initial resource ordering N , all the positional variables of a resource are labelled before the labelling of the next resource. In case the number of fails (*i.e.*, failed value assignment to positional variables) reaches a parameter k during the labelling of a resource, the ordering N is changed dynamically, as described shortly. The process is then restarted.

The values of positional variables are randomly chosen, giving higher probabilities to orders with the earliest deadlines. For free orders, volumes, products and tanks are set after their respective positional variables are labelled.

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

After the occurrence of k fails involving a resource, the backtrack tree is reinitialized with that resource as the first element in the topological ordering, and k is incremented by a constant. This implementation was motivated by the fact that, during test runs, it was observed that a fair number of fails were caused by earlier decisions taken when instantiating variables associated to pipelines in the given sequencing. We empirically determined that $k = 150$ was a good increment. Also, some precautions were taken to avoid the repetition of pipeline sequences that have already been tried, as indicated in step 9 of Algorithm 2.

4.4 The Scheduling Sub-problem

The second sub-problem of the complete CP model is solved by a simpler constraint model which is responsible for assigning the exact times to pumping operations, respecting forbidden alignment configurations and avoiding simultaneous pipe usage. Once a solution to the sequencing model is found, we already have the sequencing of all orders at each pipeline and at each tank.

The pumping operations are created by simply checking the edges of the pushing graph, as explained in section 4.3. They already represent the minimum number of pumping operations required, given the way the graph is generated. For each edge, a new type of *send* and *receive* operation is created, with the corresponding volume set to $pvol$. Additionally, a procedure for *unifying* such activities was built in order to reduce the number of variables in the model. Basically, it analyzes paths in the pushing graph, checking if activity attributes such as volumes and times are the same. A simple example can be seen in figure 9. If activity A_i pushes activities A_j and A_{j+1} , four operations will be created. However, only two operations are truly required, since Op_1^i is the same as Op_1^j , and similarly for Op_2^j and Op_1^{j+1} . This procedure can be efficiently implemented by simply decomposing paths in the pushing graph.

Let **Dep** be the set of depots, and let $PumpOps_d$ give the pumping operations that

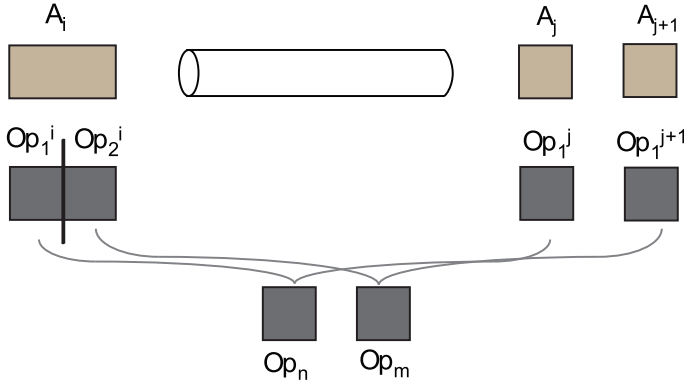


Figure 9: Example of activity unification.

will start at depot d , for each $d \in \mathbf{Dep}$. The simultaneous sending constraints can be enforced using a *discrete resource*, $DiscSending_d$. Such a resource limits the number of simultaneous operations to a certain maximum number [30]. Thus, we associate each operation in $PumpOps_d$ with its respective resource $DiscSending_d$, and with an input value limit set to $DepotMaxSimultaneousOperations_d$. The constraints are, for all $d \in \mathbf{Dep}$:

$$pumpOp \text{ requires } DiscSending_d : \forall d \in \mathbf{Dep}, pumpOp \in \mathbf{PumpOps}_d \quad (49)$$

$$MaxCapac(DiscSending_d) = DepotMaxSimultaneousOperations_d. \quad (50)$$

Similarly, the forbidden alignment configurations are enforced using discrete resources $AlignDisc_{a,d}$, one for each alignment restriction a and depot d . The operations associated with each resource are easily identifiable by checking their product type and flow direction. Assume that the forbidden alignment configuration set at depot d is given by $\mathbf{Align}_d$. For each $a \in \mathbf{Align}_d$, let $OpAlign_{d,a}$ be the set of pumping operation tuples of size $|a|$, starting in d , that might violate the alignment conditions. This latter case is easily identifiable by just checking product types and flow directions. We then state, for all $d \in \mathbf{Dep}$ and all $a \in \mathbf{Align}_d$:

$$pumpOp \text{ requires } AlignDisc_{d,a} : pumpOp \in OpAlign_{d,a} \quad (51)$$

$$MaxCapac(AlignDisc_{d,a}) = |a| - 1. \quad (52)$$

The time variables are labeled with the least possible value in their domains. This forces pumping to start as soon as possible. In case a failure ensues, a new sequencing solution is requested, most certainly a different one, due to the randomizations present in the previous model.

4.5 A Running Example

In this section we use a small running example to illustrate important aspects of the planning and scheduling phases. To this end, consider the net topology depicted in figure 10.

We have six depots: $D_0, \dots, D_5$. Products F_0 and F_1 need to be pumped from D_4 and D_5 to D_1 and D_0 , respectively. Right next to each depot we have two tanks. For example, at depot D_0 we have tanks T_{000} and T_{001} for products F_0 and F_1 . At the initial instant, tank T_{000} contains 5 units of volume, indicated by writing $5u$ inside the rectangle representing that tank. At time 2, tank T_{010} at depot D_1 contains $15u$ of volume, and so on. Note that depots D_2 and D_3 are just junctions, there is no storage tanks next to them.

There are 5 pipelines depicted in the figure: $P_0, \dots, P_4$. We assume, for simplicity, that all pipelines have capacity of $5u$ and have all the same length. Flow rates are assumed to be constant all over the network.

In order to move products from D_4 and D_5 to D_1 and D_0 , respectively, we will use four routes. They are:

1. $R_0 = D_4 P_2 D_2 P_4 D_3 P_1 D_1$
2. $R_1 = D_5 P_3 D_3 P_4 D_2 P_0 D_0$
3. $R_2 = D_4 P_2 D_2 P_4 D_2 P_0 D_0$
4. $R_3 = D_5 P_3 D_3 P_4 D_3 P_1 D_1$

Inside the rectangle representing each pipeline, we indicate the product it contains, as well as the route assigned to that product. For example, at time 1 pipeline P_2 contains $5u$ of product F_0 on route R_2 . Note that R_0 is a direct route from D_4 to D_1 ; it will be used to move $5u$ of product F_0 from D_4 to D_1 . Also, R_1 goes directly from D_5 to D_0 and will be used to move $5u$ of F_1 from D_5 to D_0 . Suppose that the planning phase generates, for some reason, orders to send $5u$ of fuel F_0 on route R_0 and another $5u$ of fuel F_1 on route R_1 . Note that both these routes include pipe P_4 , and so the flow will have to reverse direction at that pipe, so as to accommodate for the passage of F_0 and F_1 through it, but in opposite directions. We can use the routes R_2 and R_3 to accomplish this reversal but, as a result, the planning phase will have to generate free delivery orders to make this happen.

At the beginning of the planning horizon all pipes are completely filled with products already routed to their destinations. For one such product, the route starts at the pipe where the product is at the beginning of the planning horizon and describes the sequence of depots and pipes to be traversed by the product until it reaches its destination depot. In other words, for products that start in some of the pipes, their routes are suffixes of

one of the four routes R_i . We can see this by examining figure 10 at time 0. At that time, pipe P_4 contains $5u$ of fuel F_0 on the shortened route $S_4 = P_4D_3P_1D_1$. Note that S_4 is a suffix of the full route R_0 . The shortened routes that concern us in the example are:

1. Suffixes of R_0 : (i) $S_1 = P_1D_1$, (ii) $S_2 = P_2D_2P_4D_3P_1D_1$, and (iii) $S_4 = P_4D_3P_1D_1$.
2. Suffixes of R_1 : (i) $S_0 = P_0D_0$, and (ii) $S_3 = P_3D_3P_4D_2P_0D_0$.

Now, let's follow some actions that could happen at the planning phase. Suppose that, at the beginning, we have a demand of $15u$ of fuel F_1 at depot D_0 at time 10. The planning algorithm notices that we already have $10u$ of F_1 in the pipelines, namely, at P_3 and at P_0 . Since it favors shortened routes because, potentially, they have smaller travelling times, it generates two delivery orders: $O_0 = (\emptyset, T_{001}, F_1, 5u, S_0, 10)$ and $O_1 = (\emptyset, T_{001}, F_1, 5u, S_3, 10)$. Notice the shortened associated routes. Since these volumes have no origin depots, that data is written as $\emptyset$ in these orders. We need some more $5u$ of F_1 to be pushed to D_0 . The shorter route from D_4 or D_5 to D_0 is R_1 (R_2 also fits, but it is longer and involves a flow reversal at P_4). So, the algorithm generates one more delivery order: $O_2 = (T_{131}, T_{001}, F_1, 5u, R_1, 10)$. By this order, we request that $5u$ of F_1 from tank T_{131} at D_4 to be sent to tank T_{001} at D_0 , via route R_1 . In order to satisfy the demand for $15u$ of F_1 at D_0 , the algorithm then generates another delivery order: $O_3 = (T_{001}, \emptyset, 15u, \emptyset, 10)$. Note that this order just discharges $15u$ of F_1 from tank T_{001} at time 10.

Further, suppose that at we have a production of $15u$ of F_1 at D_5 at time 5. Although this volume need not be used to satisfy the demand, it may be necessary at D_5 in order to push other products that are already waiting in pipes along routes R_1 and R_3 , and that have more pressing deadlines to arrive at their destinations. In response, the planning algorithm might generate a delivery order like $O_4 = (\emptyset, T_{131}, 15u, \emptyset, 5)$, saying that tank T_{131} , at D_5 , receives $15u$ of fuel by time 5.

The scheduling phase will distribute and allocate the orders in a way that all tanks and pipelines capacities are respected. A typical sequence of movements, commented as appropriate, could be as follows (see figure 10.):

From time 0 to time 1: Pumping from D_4 , with active route R_0 .

1. A *free order* injects $5u$ of fuel F_0 from tank T_{120} (at D_4) into P_2 , on route R_2 . We denote this injection move by m_0 . Note that this volume is scheduled for flow reversal at P_4 , later on. This free order would have been generated by the planning algorithm.
2. Move m_0 pushes F_0 from P_2 into P_4 , on route S_2 . It is easy to check that this new move, denoted by m_1 , is compatible with route S_2 .

3. Move m_1 , in turn, pushes F_0 from P_4 into P_1 , on route S_4 and is compatible with route S_2 .
4. A *free order* generated by the planning algorithm extracts $5u$ of fuel F_0 from P_1 into tank T_{010} (at D_1).

From time 1 to time 2: Pumping from D_4 , with active route R_0 .

1. A *free order* injects $5u$ of fuel F_0 from tank T_{020} (at D_4) into P_2 , on route R_0 . This injection move is denoted by m_2 and would have been generated by the planning algorithm.
2. Move m_2 pushes F_0 from P_2 into P_4 , on route R_2 . The latter move is denoted by m_3 and can be easily checked to be compatible with route R_2 . This volume on P_4 will reverse its flow on the next move.
3. Move m_3 pushes F_0 from P_4 into P_1 , on route S_2 . The corresponding move, denoted by m_4 , is compatible with route S_2 .
4. A *free order* generated by the planning algorithm extracts $5u$ of fuel F_0 from P_1 into tank T_{010} (at D_1).

From time 2 to time 3: Pumping from D_5 , with active route R_1 .

1. A *free order* injects $5u$ of fuel F_1 from tank T_{131} (at D_5) into P_3 , on route R_1 . This free order would have been generated by the planning algorithm and is denoted by m_5 .
2. Move m_5 pushes F_1 from P_3 into P_4 , on route S_3 . The resulting move, denoted by m_6 , is clearly compatible with route S_3 .
3. Move m_6 pushes F_0 from P_4 into P_0 , on route R_2 . This move is compatible with route R_2 , and its volume reverses the flow direction in P_4 .
4. Order O_4 extracts $5u$ of fuel F_1 from P_0 into tank T_{001} (at D_0). This is a free order generated by the planning algorithm.

Note that we had five free orders generated by the planning algorithm. Also note that order O_0 has been serviced already. Further, the volume of F_1 , of order O_1 is now in pipe P_4 and the the volume of F_1 , of order O_2 is now in pipe P_3 . Hence, we have now $5u$ of fuel F_1 in tank T_{001} , and we have 10 more units of F_1 in the pipeline. If they arrive by time 10, we will guarantee enough of F_1 at D_0 to satisfy the demand specified by order O_3 . Finally, production order O_4 can now inject $15u$ of F_1 in tank T_{131} , at D_5 . We see that tank T_{131} is already empty by time 3 and, if it remains empty by time 5, it will be able to accommodate the injection specified by order O_4 .

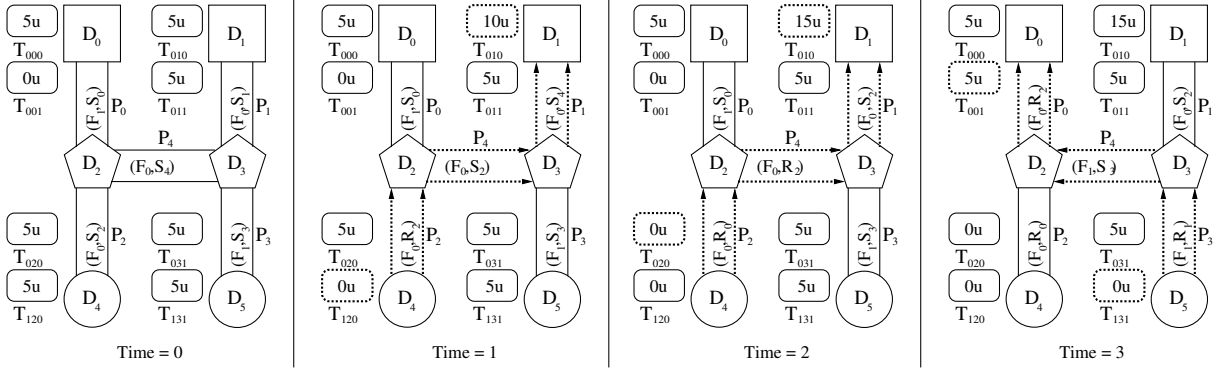


Figure 10: A small example.

5 Results

Solutions were obtained on a *Intel Pentium D* 3.40 Ghz CPU platform, with 4GB of memory. The planning and scheduling phases were coded in C++ and compiled using *GCC-4.0*. The CP model was solved using *ILOG Solver 6.2* and *ILOG Scheduler 6.2*, with medium to high propagation enforcement.

We used four real field instances to test the models. The first two rows in Table 3 indicate the planning horizon and the number of deliver orders generated by the planning phase, respectively, for each of the test instances. The remaining lines give details of typical runs. All instances share the same network topology of 14 depots, 30 pipelines, 32 different product types and 242 tanks distributed among the depots. Pipelines volumes range from very small 30 m³ capacity pipes up to 8,000 m³ volume pipes. Most of the tank capacities are between 4,000 and 30,000 m³.

Figure 11 shows the total tankage in all depots for product groups LPG, Diesel, Gasoline, and Naphtha in two different instances. Here, a group is formed by gathering products with the same composition but different quality. For instance, in the Gasoline group we have normal and premium quality gasoline. This grouping is used often by pipeline operators. The figure displays volume variations over time in a typical solution, covering the entire planning horizon. Note that, in certain cases, there was a reduction of more than 15,000 m³ in a single group, attesting to an intense use of the pipeline network. Mainly, this volume variation indicates refilling activities necessary to maintain inventory levels, specially for those lines connected to depots with low-capacity tanks. For this same reason, some pipelines did not show the typical intermittent usage, but were continuously used throughout the planning horizon (not shown in the figure). Also, notice from figure 11 that the algorithm maintained the final and initial stocks at similar levels.

In all cases, the solver found a solution with only one iteration between phases and in a reasonable amount of computer time, *e.g.*, within 10 minutes. The planning phase was

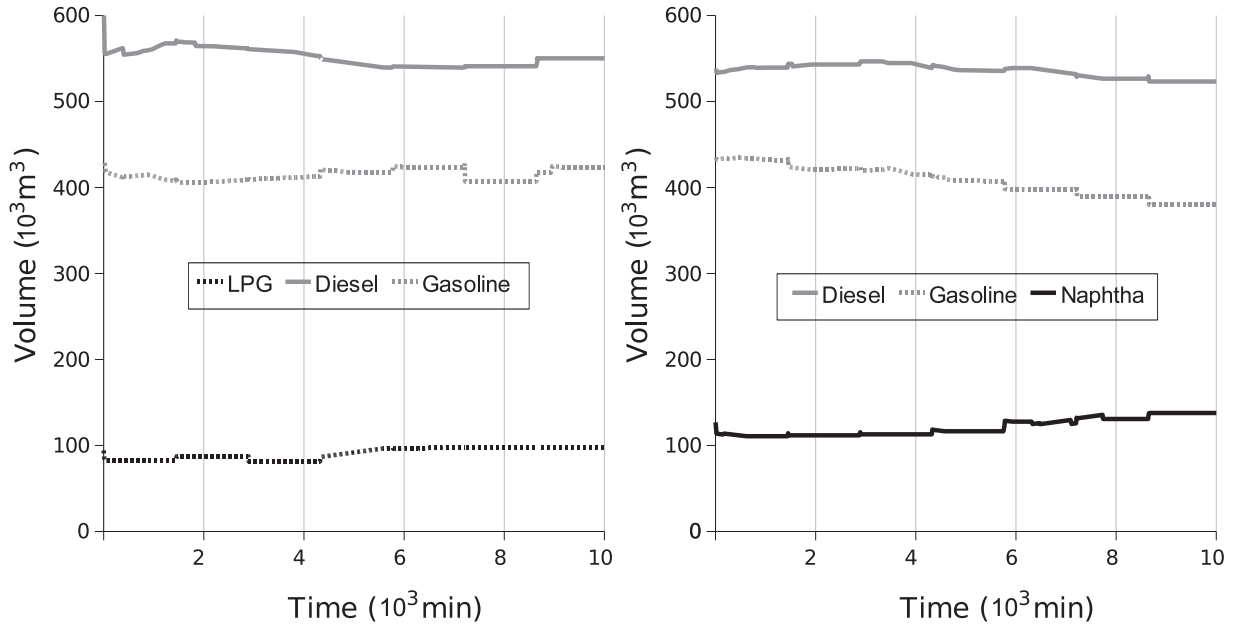


Figure 11: Tankage Evolution for two instances.

slower because it has to iterate through many combinations of products, bases, routes, and tanks for each order. Also, most variables of the scheduling phase were instantiated as a result of constraint propagation making it run faster. The search heuristic, which proved crucial in the planning phase, was also instrumental to improve other important aspects of the solution quality, a fact attested by logistic engineers. For instance, usually, the solution computed by the algorithm showed only a very small number of pipeline flow reversions, the kind of operation that engineers prefer to keep to a minimum. Also, new and interesting delivery routes were identified for some products. Some of them came as a surprise to logistic engineers, who are biased towards using of the same traditional routes they already know about from manually planning the network operation.

6 Conclusions

We proposed a novel procedure for generating feasible solutions for real instances stemming from planning and scheduling the operation of a very-large pipeline network, used to transport petroleum derivatives and ethanol. The operation of such a network is subjected to a complex set of physical and operational constraints. It makes possible the delivery of oil and biofuel to local markets, as well as the storing of the excess production from refineries. Using the CP paradigm, these constraints were adequately modelled. Problems of this size and complexity, as known by the authors, would not be solved by other

Table 3: Solver and model statistics

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

approaches reported in the literature to date, in which much of the difficult constraints and topologies are overlooked.

The algorithm has two phases. The *planning phase* generates the so called *delivery orders*, which describe how product volumes should be transmitted between depots. A delivery order contains information regarding product, volumes, origin and destination tanks, as well as timing deadlines. In this phase, a number of randomized heuristics work together in order to select appropriate depots, tanks and volumes so as to satisfy demand and production schedules. The *scheduling phase*, on the other hand, creates an ordering among the delivery orders and, further, assigns start and end times for each delivery order, while observing all deadlines and operational constraints. Additionally, it also assigns volumes and tanks for the so called *free delivery orders*. Several reasons motivated the use of CP techniques to model the activities that comprise this phase. Primarily, the scheduling problem is highly over-constrained and has several non-linear constraints, easily modelled in the CP paradigm. Besides, the main goal was to search for a feasible solution. Notably, the choice of variables for value assignment in the CP model uses a special type of restart strategy. In case no solution is found in this phase, a new set of orders can be requested from the planning phase.

The present modelling and implementation stage was reached after 2 years of problem specification, data gathering, model development, and testing. The procedure is already integrated with a proprietary flow simulation tool and the company is currently consider-

ing it for routine use on a daily basis. The tool has already proved its value, showing that it can save many valuable work hours of skilled engineers. Also, the tool allows for many different planning and scheduling scenarios to be easily set-up and quickly tested, by varying local demand needs and production schedules at refineries. In this way, a significant difference in the end-user daily routine is the ability to save many hours interacting with the simulator when searching for a feasible and adequate solution.

There are several opportunities for further research related to this problem. First, new real-world constraints are being considered to improve the adequacy of the overall model. Such could include inventory management restrictions, limitations on energy use at critical daily periods and at specific depots, and shut-down periods or partial operation intervals for tanks and pipelines. Also, one can implement more sophisticated search heuristics for both the planning and scheduling phases, making the overall approach capable of dealing with more specific real instance classes. When modelling such new constraints, we feel that the flexibility of the CP paradigm will again prove to be crucial. Finally, one can consider objective functions that would help guide the heuristics. This would provide a yardstick that could be used to gauge solution quality.

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

References

- [1] Group, A.E.: How pipelines make the oil market work - their networks, operation and regulation (2001)
- [2] Wilson, R.: Transportation in america, eighteenth edition. Washington, D.C.: Eno Transportation Foundation, Inc. (2001)
- [3] Moura, A., de Souza, C., Cire, A., Lopes, T.: Planning and scheduling the operation of a very large oil pipeline network. In: Lecture Notes in Computer Science - Proceedings of the 14th International Conference on Principles and Practice of Constraint Programming. Volume 5202. (Setembro 2008) 36–51
- [4] Moura, A., de Souza, C., Cire, A., Lopes, T.: Heuristics and constraint programming hybridizations for a real pipeline planning and scheduling problem. In: Proceedings of the 11th IEEE International Conference on Computational Science and Engineering - CSE'08. (Julho 2008) 455–462
- [5] Camponogara, E., Souza, P.S.: A-Teams for oil transportation problem through pipelines. In: Information Systems Analysis and Synthesis, Orlando, United States (1996)
- [6] Milidiú, R., dos Santos Liporace, F.: Planning of pipeline oil transportation with interface restrictions is a difficult problem. Technical Report 56, Departamento de Informática, PUC-Rio, Rio de Janeiro, RJ, Brasil (2003)
- [7] Milidiú, R.L., Pessoa, A.A., Laber, E.S.: Pipeline transportation of petroleum products with no due dates. In: LATIN '02: Proc. of the 5th Latin American Symposium on Theoretical Informatics, London, UK, Springer-Verlag (2002) 248–262
- [8] Milidiú, R.L., Pessoa, A.A., Laber, E.S.: The complexity of makespan minimization for pipeline transportation. Theoretical Computer Science **306**(1-3) (2003) 339–351
- [9] Alves, V., Filho, V.J.: Pipeline scheduling of petroleum derivatives using genetic algorithm. In: IV Congresso Brasileiro de Pesquisa e Desenvolvimento em Petróleo e Gás, Campinas, Brazil (2007)
- [10] Cafaro, D.C., Cerdá, J.: Optimal scheduling of multiproduct pipeline systems using a non-discrete MILP formulation. Computers & Chemical Engineering **28**(10) (2004) 2053–2058

- [11] Rejowski, R., Pinto, J.M.: A novel continuous time representation for the scheduling of pipeline systems with pumping yield rate constraints. *Computers & Chemical Engineering* **32** (2008) 1042–1066
- [12] Filho, E.M.S., Filho, V.J., de Lima, L.S.: Variable neighborhood search (VNS) applied to pipeline distribution problem with capacity constraints. In: IV Congresso Brasileiro de Pesquisa e Desenvolvimento em Petróleo e Gás, Campinas, Brazil (2007)
- [13] Camponogara, E.: A-Teams para um problema de transporte de derivados de petróleo. Master's thesis, Instituto de Matemática, Estatística e Ciência da Computação, Universidade Estadual de Campinas, Campinas, Brazil (1995)
- [14] de la Cruz, J., Andrés-Toro, B., Herrán-González, A., Porta, E.B., Blanco, P.F.: Multiobjective optimization of the transport in oil pipelines networks. In: Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation. Volume 1. (2003) 566–573
- [15] de la Cruz, J., Herrán-González, A., Risco-Martín, J., Andrés-Toro, B.: Hybrid heuristic and mathematical programming in oil pipelines networks: Use of immigrants. *Journal of Zhejiang University SCIENCE* **6A**(1) (2005) 9–19
- [16] Magatão, L., Arruda, L., Neves, F.: A mixed integer programming approach for scheduling commodities in a pipeline. *Computers & Chemical Engineering* **28**(1) (2004) 171–185
- [17] Magatão, L., Arruda, L., Neves, F.: Using CLP and MILP for scheduling commodities in a pipeline. *Computer-Aided Chemical Engineering* **20B** (2005) 1027–1032
- [18] Relvas, S., Barbosa-Póvoa, A.P.F.D., Matos, H.A., Fialho, J., Pinheiro, A.S.: Pipeline scheduling and distribution centre management - a real-world scenario at CLC. In: Proceedings of the 16th European Symposium on Computer Aided Process Engineering and 9th International Symposium on Process Systems Engineering, Garmisch-Partenkirchen, Germany (2006) 2135–2140
- [19] Relvas, S., Matos, H.A., Barbosa-Póvoa, A.P.F.D., Fialho, J., Pinheiro, A.S.: Pipeline scheduling and inventory management of a multiproduct distribution oil system. *Industrial & Engineering Chemistry Research* **45**(23) (2006) 7841–7855
- [20] Relvas, S., Matos, H.A., Barbosa-Póvoa, A.P.F.D., Fialho, J.: Reactive scheduling framework for a multiproduct pipeline with inventory management. *Industrial & Engineering Chemistry Research* **46**(17) (2007) 5659–5672

- [21] M. Sasikumar, M., Prakash, P.R., Patil, S.M., Ramani, S.: Pipes: A heuristic search model for pipeline schedule generation. *Knowledge-Based Systems* **10**(3) (1997) 169–175
- [22] Rejowski, R., Pinto, J.M.: Scheduling of a multiproduct pipeline system. *Computers & Chemical Engineering* **27**(8) (2003) 1229–1246
- [23] Rejowski, R., Pinto, J.M.: Efficient MILP formulations and valid cuts for multiproduct pipeline scheduling. *Computers & Chemical Engineering* **28**(8) (2004) 1511–1528
- [24] Crane, D.S., Wainwright, R.L., Schoenefeld, D.A.: Scheduling of multi-product fungible liquid pipelines using genetic algorithms. In: *Proceedings of the 1999 ACM Symposium on Applied Computing*, San Antonio, USA (1999) 280–285
- [25] Milidíu, R., dos Santos Liporace, F., de Lucena, C.J.P.: Pipesworld: Planning pipeline transportation of petroleum derivatives. In: *Proceedings of ICAPS’03 - Workshop on the Competition: Impact, Organization, Evaluation, Benchmarks*, Trento, Italy (2003)
- [26] Competition, t.I.P.: <http://zeus.ing.unibs.it/ipc-5/>
- [27] Hooker, J.N.: *Integrated Methods for Optimization* (International Series in Operations Research & Management Science). Springer-Verlag, Secaucus, USA (2006)
- [28] Cheng, B.M.W., Choi, K.M.F., Lee, J.H.M., Wu, J.C.K.: Increasing constraint propagation by redundant modeling: an experience report. *Constraints* **4**(2) (1999) 167–192
- [29] Marriot, K., Stuckey, P.: *Programming with Constraints: An Introduction*. 1^a edn. MIT Press, Cambridge, Massachusetts (1998)
- [30] ILOG: ILOG Scheduler 6.2: User’s Manual. ILOG. (2006)
- [31] Kautz, H., Horvitz, E., Ruan, Y., Gomes, C., Selman, B.: Dynamic restarts policies. In: *Proceedings of the AAAI-2002*, Edmonton, Alberta (2002)

Epílogo

A fase de planejamento aqui desenvolvida utilizou-se do conhecimento e experiência dos operadores da rede para a construção do conjunto de planos de entrega. Como visto, essa construção é incremental, com um grupo de funções avaliando as possíveis propriedades de um plano. Após a avaliação, a escolha da propriedade é aleatória sobre um subconjunto das melhores candidatas. Assim, planos são criados para atender as demandas mais urgentes através da rota que permita uma entrega mais rápida que cause o menor número de reversões. O tanque origem é aquele com maior volume disponível do produto e o de destino é aquele com maior espaço livre. Além disso, o volume do plano de entrega é maximizado de acordo com os tanques escolhidos.

A criação de cada plano tem como objetivo a satisfação de uma demanda e o procedimento termina quando todas demandas tenham sido atendidas. Neste ponto, o conjunto de planos pode ser alterado pelos operadores para satisfazer outras necessidades como, por exemplo, a transferência de quantidades entre órgãos para satisfazer demandas futuras, dentre outras.

No modelo de agendamento apresentado, os planos de entrega são vistos como *atividades*, ou seja, estruturas que permitem a aplicação de restrições globais mais específicas para problemas de agendamento [25]. Os planos geram duas atividades para cada duto por onde passam, uma atividade para a entrada no duto e a outra para a saída do duto. Cada atividade é, de forma geral, composta por um tempo de início e tempo final, os quais modelam os *intervalos* de ocorrência do plano em cada duto, ao invés de modelar os bombeamentos explicitamente.

Esta representação com intervalos de tempo permite uma formulação mais compacta do modelo para a rede de dutos, e adicionalmente, facilita a aplicação de restrições globais relativas às disjunções temporais, como as restrições *cumulative* e *disjunctive*, apresentadas no texto. Como há também um menor número de variáveis, também é possível aumentar o nível de propagação das restrições, o que reduz o tamanho da árvore de *backtracking*.

O sequenciamento dos planos de entrega foi dividido segundo dois pontos de vistas de modelagem: a visão do plano (*order viewpoint*) e a visão das operações (*operations viewpoint*). A primeira visão engloba as restrições que definem a movimentação do plano pela sua rota, definindo limitantes temporais à sua realização. Isso inclui o tratamento das vazões, a conexão entre o recebimento em um duto e o envio no próximo duto, o atendimento do prazo na base destino e a utilização de recursos nos tanques de origem e destino. Além disso, nessa visão é aplicada à restrição de disjunção temporal entre as realizações das ordens entre si num mesmo duto ou tanque. A visão das operações implementa o funcionamento dutos, forçando a necessidade de se extrair uma quantidade

igual de volume da extremidade oposta quando uma quantidade é injetada no duto. Da mesma forma, essa visão determina como a direção de movimentação em um duto pode ser alterada considerando a necessidade de planos específicos para a realização de reversões. Outro aspecto tratado são os pares incompatíveis, proibidos de seguirem juntos num mesmo duto.

O grafo de empurramento (*Pushing graph*) é uma representação redundante do problema, mas que ajuda na propagação das restrições, principalmente, aquelas referentes à precedência de atividades. O grafo é construído sobre as atividades, analisando um sequenciamento destas em toda a rede. Os vértices desses grafos são os pares de planos e dutos, já que as atividades são criadas segundo esses mesmos pares. Assim, se uma atividade de envio ocorre ao mesmo tempo que outra de recebimento no mesmo duto, então um plano de entrega ao entrar por uma extremidade empurra outro plano de entrega para fora do duto. Isso cria uma aresta no grafo de empurramento entre dois pares de planos naquele duto. Sobre essa representação pode-se criar diversas restrições relacionando as quantidades dos planos de entrega, o volume dos dutos e as precedências entre as atividades.

A representação pelo grafo de empurramento dá uma visão global sobre as relações de precedência, sendo essencial para a eficiência do modelo de programação por restrições do subproblema de sequenciamento.

As atividades com suas movimentações por intervalos de tempo, contudo, não caracterizam uma solução do **PPAORO**, conforme descrito no capítulo 2. Assim, um segundo modelo PR para *atribuição de tempos* é utilizado para determinar exatamente os tempos dos bombeamentos. Tais tempos são calculadas a partir do sequenciamento dos planos, obtido pela solução do modelo anterior. As variáveis do modelo de atribuição de tempos representam exatamente as operações de envio e recebimento que ocorrerão nos dutos, determinadas a partir de uma solução parcial do problema de agendamento. As restrições faltantes, tal como envios simultâneos e alinhamentos, são facilmente implementadas neste segundo submodelo. O artigo não apresenta a modelagem dos requisitos C5, C10 e C15 em decorrência da falta de dados na instância, mas estas podem ser implementadas utilizando-se simples restrições de disjunção temporal.

O artigo também apresenta um método de busca diferenciado de soluções, denominado *backtracking adaptativo*. Ao invés de fixar uma ordem de variáveis para atribuição de valores, o algoritmo procura identificar os dutos mais críticos a partir do número de *fails* gerados pelo resolvidor de PR durante o sequenciamento dos planos. Espera-se que a ocorrência de muitos *fails* seja devida à decisões equivocadas no passado e, portanto, os dutos problemáticos ganham prioridade na ordem de sequenciamento, quando uma nova árvore de *backtracking* é construída.

O modelo apresentado foi obtido após 2 anos de especificação, extração de dados,

modelagem e testes. O procedimento foi integrado a um simulador proprietário da empresa e as soluções foram validadas por engenheiros da PETROBRAS. É importante notar que as novas rotas encontradas estão dentro das existentes, porém não são usuais para os produtos utilizados. Como foi demonstrado, o tempo de execução para horizontes de até 10 dias manteve-se aceitável.

Existem muitas oportunidades de melhoria dos modelos de programação por restrição. No entanto, é notório que a fase de planejamento mensal recebeu atenção reduzida nesse artigo e a utilização de uma heurística, mesmo que suficiente para os propósitos de agendamento, ainda é um passo inicial. O próximo artigo apresentará uma nova modelagem que tratará a fase de planejamento usando fluxo em redes.

Capítulo 7

Um modelo para o Problema de Planejamento

Prólogo

O artigo deste capítulo foi submetido ao periódico *Computers & Chemical Engineering*, em 2010, e apresenta uma formalização do problema de planejamento junto com uma modelagem baseada em fluxo em redes.

No artigo apresentado anteriormente, a fase de planejamento recebe um tratamento heurístico. Não há, no entanto, uma formalização do problema determinando quais as restrições tratadas ou mesmo uma função objetivo. O conjunto de características tratadas do **PPAORO** restringe-se às estimativas de atendimento da demanda, ocupação dos dutos e utilização dos tanques. Não é possível, por exemplo, determinar o estado de toda a rede em um instante de tempo qualquer, o que seria necessário para verificação de restrições básicas do **PPAORO**.

Embora o tratamento apresentado seja suficiente para obtenção de instâncias de entrada para o problema de agendamento, há pouca informação sobre a qualidade dessas instâncias. Não é surpresa que um conjunto de ordens criado considerando-se menos restrições trará dificuldade para o algoritmo de agendamento. Portanto, uma modelagem que considerasse mais restrições possivelmente forneceria entradas de maior qualidade.

Os tratamentos estudados na literatura resolveram o **PPAORO** focando na obtenção de uma solução para o problema de agendamento. No entanto, nenhum desses procedimentos conseguiu tratar eficientemente uma rede com topologia complexa e um horizonte de um mês. Isso deve-se ao fato, primordialmente, que o problema de agendamento detalha diariamente, em horas ou minutos, as operações nos dutos. O problema de planejamento mensal não necessita disso, podendo sua solução indicar somente o tráfego diário. Sendo assim, muitas restrições deixam de ser aplicáveis, como por exemplo, aquelas referentes à

sequência dos bombeamentos.

O artigo a seguir apresenta uma formalização do problema de planejamento utilizando um subconjunto de restrições do **PPAORO** e uma função objetivo que minimiza o custo de transporte. A seleção das restrições foi feita conforme as principais preocupações dos engenheiros de logística quando constroem soluções manuais

A modelagem de fluxo em redes, aqui proposta, não utiliza a visão tradicional utilizada em outros modelos [9], e pode ser resolvida usando um resolvedor de modelos de programação linear. Isso torna o procedimento eficiente para teste de diversos cenários, variando a previsão de demanda e o planejamento de produção.

Nesse artigo, a fase de planejamento é entendida também como um problema isolado de planejamento tático. Neste problema é necessário dizer quais serão os bombeamentos diários partindo de cada rota para satisfazer as demandas previstas e escoar as produções planejadas. Os resultados obtidos com o uso do procedimento foram comparados com as resoluções manuais desse mesmo problema e apresentaram melhorias significativas.

Planning The Operation of a Large Real-World Oil Pipeline

Tony M. T. Lopes, Andre A. Cire, Arnaldo V. Moura, Cid C. de Souza
Institute of Computing - University of Campinas
13081-970, Campinas, SP
{arnaldo, cid}@ic.unicamp.br, {andre.cire, tony.lopes}@gmail.com

Abstract

A set of oil derivative distribution depots, including refineries and terminals, have local demands and productions of different products in a given time horizon. However, there may be not enough local stock of some product to satisfy the corresponding demand, or there may not be enough tank capacity to stock the local production. This brings the need for transportation of oil derivatives through a network of pipelines. To accomplish that, a tactical pumping plan is composed monthly, and a more detailed operational schedule, spanning a few days, must be updated daily. Both the planning and the scheduling must satisfy a large set of operation constraints. This work defines the tactical planning problem and proposes a novel network flow model to solve it. Also, a procedure is given to decompose the solution into a specific input format, as needed by another solver that computes the final, detailed, daily scheduling solution. Our model treats the oil pipeline network that is operated by the Brazilian oil company PETROBRAS. This is one of the most complex and large topologies when compared to other networks treated in the open literature. The model was tested with real-world instances and showed significant improvements over human planning.

1 Introduction

The Oil Industry faces many difficult logistic problems that must deal with unstable markets and large amount of resources. In this context, planning problems are amongst the most important and have received intense attention [1, 2, 3, 4, 5].

This paper focus the inland oil derivatives distribution problem stemming from a network of pipelines operated by the Brazilian oil company PETROBRAS. Being the 15th largest oil company in the world¹, it faces a very difficult transportation problem in which

¹See www.energyintel.com.

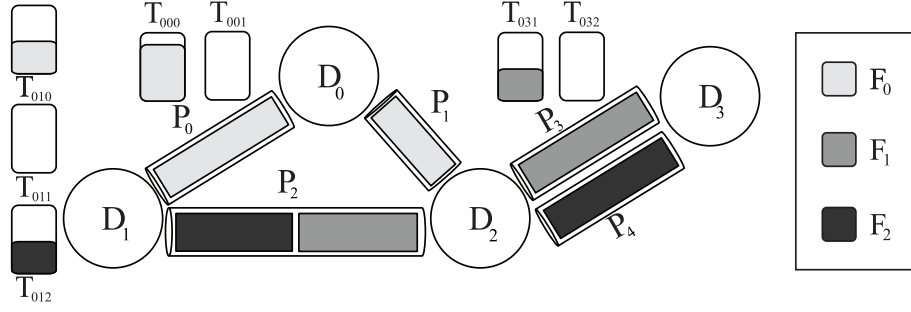


Figure 1: A Sample Pipeline Network.

ethanol and several petroleum derivatives, like gasoline, diesel, and naphtha, must be transported from refineries to depots, where consumer markets are located. Pipeline networks offer the most economical way available to transport oil derivative products inland. The scenario studied here has an extension of 7,000 kilometers, comprising 29 individual interconnecting pipelines. There are 14 distribution depots that can store up to 10 millions cubic meters of products, stocked in more than 200 tanks located at various depots. Figure 2 depicts the network topology.

The use of such a complex network must be approached at the strategic, tactical, and operational levels [6]. While the first level deals mostly with planning adjustments and extensions to the current network, the latter two manage the network operation. The difference among them resides in the amount of details each one treats, the time horizon considered, and in their operational objectives. At the tactical level, one usually aims at checking if the production plans in the refineries are enough to satisfy the forecasted demand. It must take into account as many operational decisions as possible, while still keeping the problem solvable in a feasible amount of time. It usually spans large horizons comprising from months to years. At the operational level, one treats the problem of constructing detailed daily schedules for the pumping operations at each depot, in such a way as to guarantee that all listed demands are satisfied and all refinery productions are properly stocked.

Most of the previous approaches to this problem, however, focus the operational level where detailed daily pumping schedules must be planned [7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 3]. A few other works could be considered tactical approaches [17, 18, 19]. In any case, they could not be directly applied to our problem, since they handle much simpler networks, usually with only one pipeline.

Solving the tactical problem manually can be ineffective. It is necessary to obey many refinery production schedules and the pipeline network is too complex for in depth man-

ual analysis of all possible operational constraint violations. In fact, the only constraints usually verified in manually constructed solutions are mass conservations at pipelines and tanks. On the other hand, the resulting tactical plan has a huge impact when scheduling daily plans. A bad tactical plan can result in overloading certain pipelines while underusing others. This, in turn, can lead to overdue demands. As another consequence, refinery productions of some derivatives might have to be reduced because local stocks are too high and products are not properly scheduled to be timely extracted from the corresponding tanks at refineries. To avoid these situations, engineers have to produce very conservative plans with many decisions depending solely on their past experiences. Furthermore, such manual procedures do not admit precise cost optimization considerations.

In this paper, we present a formal description of the planning problem, as we are going to call the tactical planning problem henceforth. This formalization takes into account the operational constraints that will have the greatest influence at the subsequent daily scheduling problem. A network flow model [20] is proposed to solve the planning problem. The main difficulty was to model specific pipeline characteristics, *e.g.* that they must be always completely full, while moving products through predefined pipeline routes. Transportation costs will be the objective to minimize. A decomposition algorithm is also given to extract individual pumping operations from the network flow solution. These operations could, then, be used as input to detailed daily scheduling algorithms, such as the one proposed in [3]. Computational results showed that the proposed model has an adequate execution time and produces solutions with up to 25% cost reductions when compared to manual solutions.

Section 2 describes the tactical planning problem. Section 3 applies network flow to model the tactical planning problem. The computational results can be seen in section 4. Section 5 summarizes our contributions and suggests further advances.

2 Problem Definition

The topology of a pipeline network system is given by three sets: *tanks*, *depots*, and *pipelines*. Tanks are used for product storage. During the whole planning period, a tank can store only a single type of product. Depots are geographically dispersed units where local demands for oil-derivatives occur. Each depot has its own subset of tanks and some depots may also hold refinery facilities. Pipelines interconnect the depots and are used for product transportation. Each individual pipeline connects only two depots.

An illustration of a pipeline system is presented in Figure 1, in which products (or *fluids*) F_0 , F_1 , and F_2 can circulate. The four depots D_0 , D_1 , D_2 and D_3 are connected by the pipelines P_0 , P_1 , P_2 , P_3 , and P_4 . A label T_{ijk} refers to the i -th tank in depot D_j being designated to stock product F_k . Note that two pipelines connect depots D_2 and

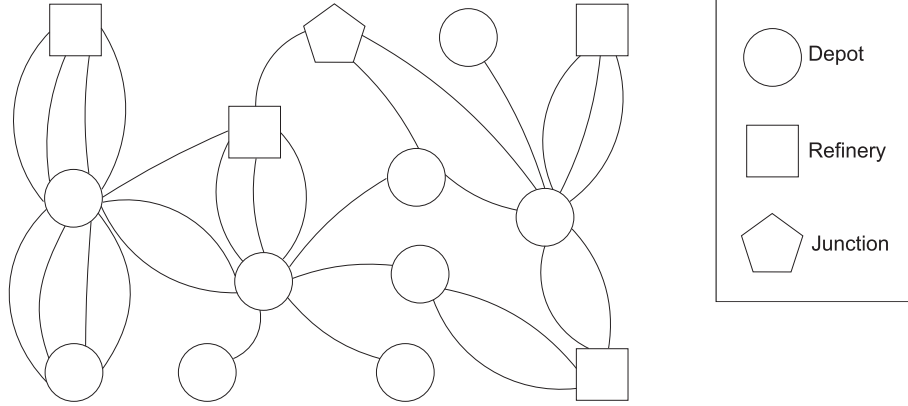


Figure 2: The inland pipeline network.

D_3 , which is a common situation in practice.

Volumes must be extracted from tanks before pumped into a pipeline. After pumped out of a pipeline, volumes can either enter a tank or move directly into another pipeline. The sequence of pipelines traversed by a volume when moving from its origin to its destination comprises its route. More precisely, we define a *route* as an alternating sequence of depots and connecting pipelines. For example, the sequence $D_0P_1D_2P_3D_3$ represents a valid route in Figure 1.

A single pipeline in a route is called a *segment*. All volumes in circulation must have a predefined route assigned to them. Also, a volume cannot be stocked at intermediate depots while moving along its route; it can only be deposited in an available tank at its destination depot.

The problem consists of planning *oil derivative transfers* between depots through valid routes in order to satisfy all depots inventory constraints, meet local demands, and accommodate all the refineries production schedules. The planning must also obey a complex set of operational restrictions over a given planning time period, or *horizon*, usually a month.

The constraints described in the next subsections were selected among many real operational constraints [3]. They represent the minimum set of constraints that are necessary to observe in order to produce a feasible tactical planning. All of these constraints must be verified by the end of each day, except for the stock levels, which are verified weekly. When producing a detailed operational pumping schedule, there are additional aspects that must be taken into consideration, and which are not considered by the tactical planner. These restrictions will not be treated here (see [3] for further discussions.) Restrictions regarding tanks, pipelines and stock levels are described in the following three

sections, respectively.

2.1 Tank Restrictions

All tanks must satisfy the following restrictions:

- (1) Tanks have a fixed and limited maximum capacity which must always be respected. Minimum capacities are all set to zero.
- (2) A tank can only store one type of product during the whole planning horizon. This constraint, required by field operators, ensures adequate product quality by avoiding possible mixtures. A depot can contain more than one tank for a given product, but it does not necessarily contain tanks for all products. It is possible that a depot contains no tanks at all, being used solely as an intermediate transmission node between two pipelines. In Figure 1, depot D_2 depicts such a situation.
- (3) There is a limit to the amount of product that can be extracted and/or injected within a time period. This is due to many operational constraints that arise when managing tanks.
- (4) The initial product stock level at each tank is given and must be respected.

Problem specifications in which constraint 3 is satisfied are said to have the *individual tank* property. An alternative view used in some scheduling models [11] relax this restriction by considering virtual tanks for each product at each depot. Such virtual tanks aggregate the capacities of all real tanks of each product in a given depot.

2.2 Pipeline Restrictions

Pipeline restrictions are as follows:

- (5) Since they are pressurized, pipelines must be completely filled with products at all times. Hence, in order to pump a certain product out of a pipeline, it is necessary to inject an equal volume at the other pipeline extremity.
- (6) At the beginning, every pipeline is filled with products, separated in batches. All such batches have already being assigned a route that must be preserved.
- (7) As there is a limited number of pumps per depot, and given that products have different densities, a maximum flow rate must be observed per pipeline, per product and per flow direction.

- (8) In order to simplify the modeling, a set of assignments of products to corresponding acceptable routes is given as input.

2.3 Inventory Constraints

At each depot, stock levels must obey the following restrictions:

- (9) The stock of a product at a certain depot at time t is obtained by summing the volumes of that product at time t from all tanks that can stock the product at the given depot. The desired maximum and minimum level of each product stock per depot must be satisfied at the end of specific time periods, usually a week. Some depots might have undefined stock levels for some products. In these cases, it is assumed that the minimum level is zero and the maximum level is the sum of capacities of all tanks that can stock the product in the depot.
- (10) The stock levels at a depot vary mainly due to *production* and *demand* operations. Productions represent volumes created at a local refinery, while demands represent volume consumptions by the local market. Both are given in advance, using market estimates and other data such as raw products availabilities and refinery capabilities. Usually, a refinery depot produces more than the local tanks can actually store within the given planning horizon. Excess must be pumped out so as to accommodate the local productions and satisfy market demands at other locations.

The consumption rate of each product may vary greatly according to monthly seasonal markets. Moreover, it is difficult to foretell exactly the local market needs for a long period. As a result, pipeline operators are required to constantly update the network schedule to accommodate new demands, guaranteeing they will be satisfied on time. This can be done by solving the problem, taking an updated input instance, every time there is a need for it.

2.4 Input Instances and Solutions

The problem data is composed by: (1) the network, described by the sets of tanks, pipelines, depots and their initial states; (2) operational parameters, such as tank capacities, pipeline flow rates and stock levels; (3) a time horizon and a set of production and demand schedules (see Section 2.3) that must be accommodated and satisfied, respectively, at each depot.

Given the problem data, a feasible *solution* is obtained by defining the amount of products that must be pumped into pipelines each day, while satisfying all restrictions.

Table 1: Initial values and constants for an input instance

Constant	What it represents
$vol(l)$	total volume of pipeline l .
$PipelineStart_l$	A set of triples (p, qty, r) each representing a quantity qty of product p that must follow route r that is initially inside pipeline l . The sum of the qty values over all triples is equal to $vol(l)$.
$stock_{d,tk}$	the initial volume inside tank tk at depot d .
$production_{p,d,t}$	the volume of product p produced at depot d in period t .
$demand_{p,d,t}$	the volume of product p demanded at depot d in period t .
$stockLevelMin_{d,p}$, $stockLevelMax_{d,p}$	the minimal and maximum volumes of product p allowed at each depot.
$maxQtyInject_{e,p,l}$, $maxQtyInject_{p,l}$, $maxQtyInject_{e,l}$, $maxQtyInject_l$	the maximum quantity that can be injected in one time period in pipeline l . Here, e represents a flow direction for product p in pipeline l : P for the normal direction and R for the reverse direction. When any of the three indices is missing, the intended meaning is to take the sum over the missing indices.
$maxTransfer_{r,p,n,u}$	the maximum volume of product p that can be injected into the n -th pipeline of route r after u time periods. If there are $n - 1$ segments in route r , then index n refers to the volume that can be delivered at the last depot in route r .
$maxDepotQtyInject_d$	the maximum quantity that can be injected into pipelines leaving depot d in one time period. It is calculated considering the number of pumps available at depot d .

Sometimes, for a given set of production and demands, there may not be a feasible solution. In these cases, it is interesting to identify which productions or demands cannot be satisfied so that the input data can be adjusted.

The objective function considers the minimization of the total transfer cost in the whole network. In reality, different pipelines might have distinct operational costs. Here, we are considering homogeneous costs as these data were not fully available. So, minimizing costs is the same as minimizing the total volume transferred over the same time horizon.

3 A Network Flow Model for the Planning Phase

In a previous work of [11], a classical network flow model for our problem has been studied. This integer programming model, however, proved inefficient in practice. [21] used a linear relaxation from this model in order to get a solution for the tactical planning problem. This relaxed solution proposed flows along each pipeline, at every time instant. With this information, a heuristic was designed to construct operational schedules with feasible pumping movements. But the detailed operational schedules so produced spanned only a few days or a few products. The model we propose overcomes these difficulties and its output can be directly used by the daily scheduler proposed in [3].

But, first, note that the classical network flow model for the problem is not yet suitable because one cannot constrain the flow to follow predefined routes. This could be troublesome, as we could obtain inadequate paths for some movements. An alternative approach is to build the network flow using the predefined routes for each product. This approach is described in the next subsections.

3.1 Network Model Definition

The network model will be built by considering pairs (r, p) , where r is a route and p is a product. Each of these pairs represents a commodity. There will be T time periods and the last one will be a special one. It will signal the end of the horizon. Each period spans one day, as demands and productions are given in a daily basis. This will also help to keep the model small, with less variables and with less constraints applying to each time period. Along with the given sets of depots, pipelines, and products an input also must define the values and constants described in Table 1.

Nodes will be created in accordance with the following rules (see also Table 1 for a summary of term definitions):

- **Tank Nodes:** For each tank tk , each depot d and each time period t , with $0 \leq t \leq T$, there will be two nodes: (1) node i (or $NodeStartTk_{d,tk,t}$) with $b_i = stock_{d,tk}$

Table 2: Node description for the Pipeline Network flow Model

Node	What it aggregates
$NodeStartTk_{d,tk,t}$	production and previous stocks at the start of period t in depot d and tank tk .
$NodeEndTk_{d,tk,t}$	total of volumes received from the network and volumes that were not distributed at period t in depot d and at tank tk .
$NodePIN_{p,d,t}$	volumes received from the network at the start of period t of a particular product p in depot d .
$NodePOut_{p,d,t}$	volumes that will be distributed to the network at period t of a particular product p in depot d .
$NodePro_{p,d,t}$	the production volume $production_{p,d,t}$ of a product p in depot d at period t .
$NodeDem_{p,d,t}$	the demanded volume $demmand_{p,d,t}$ of a product p in depot d at period t .
$NodePipeStart_{p,r}$	the volume of product p inside a pipeline l at the beginning of the horizon and that has to follow route r .
$NodePipeEnd_l$	volumes that will be stocked inside pipeline l at the end of the horizon.
$NodePipe_{r,l,p,ti,t}$	volumes of product p that traveled along route r and are currently at pipeline l at period t having departed from route's r origin at period ti .

if $t = 0$, or else $b_i = 0$ if $t > 0$, and (2) node j (or $NodeEndTk_{d,tk,t}$) with $b_j = 0$. The first node represents the tank state at the beginning of period t and the other represents it at the end of that period. All incoming and outgoing flows at tank tk will go through these nodes.

- **In and out nodes:** For each product p with tanks at depot d and each time period t , with $0 \leq t < T$, there will be a node i (or $NodePIN_{p,d,t}$) and a node j (or $NodePOut_{p,d,t}$) with $b_i = b_j = 0$. Every incoming and outgoing flow at a depot will pass through these nodes. These are intermediate nodes between tanks and the pipeline network.
- **Demand and Production Nodes:** For each product p , each depot d and each period t , with $0 \leq t < T$, there will be: (1) a node i (or $NodeDem_{p,d,t}$) with $b_i = -demand_{p,d,t}$, and (2) a node j (or $NodePro_{p,d,t}$) with $b_j = production_{p,d,t}$.
- **Pipeline Initial Stock Nodes:** For each pipeline l and each triple (p, qty, r) in $PipelineStart_l$, there will be a node i (or $NodePipeStart_{p,r}$) with $b_i = qty$.
- **Pipeline Terminal Stock Nodes:** All pipelines must remain completely filled at the end of the planning horizon. So, for each pipeline l there will be a node i (or $NodePipeEnd_l$) with $b_i = vol(l)$. In this way, the flow at the end of the horizon will sum up to the pipeline volume.
- **Pipeline Route Nodes:** Some nodes will be used to model movements of products along the pipelines. An index ti names the period where the movement starts, and an index t will represent the current period. For each pair (r, p) of a route r and a product p , and each pipeline l that is part of route r , there will be a node i (or $NodePipe_{r,l,p,ti,t}$) with $b_i = 0$, for each time period $0 \leq ti \leq t < T$. In order to handle situations when the volume was already inside the pipeline at the first period, we have nodes with $ti = 0$.

Table 2 summarizes each node function. The next step is to connect these nodes with arcs. In the following description, the traffic costs are zero and flows must be greater than zero unless stated otherwise.

- **Tank Stock Arcs:** For each node $NodeEndTk_{d,tk,t}$ there will be two arcs. The first arc, $NodeStartTk_{d,tk,t} \rightarrow NodeEndTk_{d,tk,t}$, aggregates production and previous stocks. The second one, $NodeEndTk_{d,tk,t} \rightarrow NodeStartTk_{d,tk,t+1}$, represents a stock transition between time periods. Figure 3 show these arcs as the down arrows that come out of tank nodes.

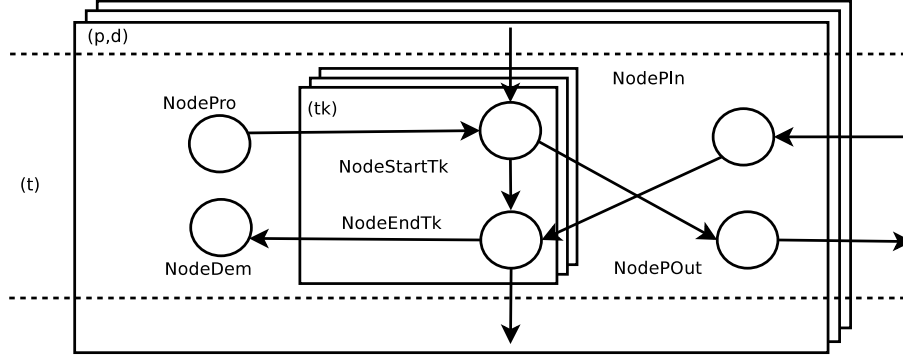


Figure 3: Nodes and arcs representing pairs of depots d and products p . Items within parenthesis index the nodes enclosed inside the rectangle. The occluded rectangles represent the multiplicity of that network with regard to the inner index.

- Tank In and Out Arcs:** For each pair of nodes $NodeStartTk_{d,tk,t}$ and $NodeEndTk_{d,tk,t}$ there will be two arcs: $NodePin_{p,d,t} \rightarrow NodeEndTk_{d,tk,t}$ and $NodeStartTk_{d,tk,t} \rightarrow NodePOut_{p,d,t}$. Here, tank tk stores product p . Volumes pass through these arcs before coming from or going to some pipeline. Figure 3 shows them as crossing arcs.
- Production and Demand Arcs:** For each pair of nodes $NodeDem_{p,d,t}$ and $NodePro_{p,d,t}$, there will be arcs $NodeEndTk_{o,tk,t} \rightarrow NodeDem_{p,d,t}$ and $NodePro_{p,d,t} \rightarrow NodeStartTk_{o,tk,t}$, for all tanks tk that can store product p at depot d . The maximum capacity of these arcs will be the same as the demand or production volumes of the respective products. Figure 3 shows these arcs as the two left horizontal arrows at the left.
- Initial Stock Arcs:** For each node $NodePipeStart_r$ and its associated content (p, qty, r) , there will be an arc $NodePipeStart_r \rightarrow NodePipe_{r,l,p,0,0}$ with minimum and maximum capacities given by qty . Figure 4 illustrates the subnetwork generated to represent a pipeline stock.
- Outgoing Terminal Arcs:** For each node $NodePipe_{r,l,p,t,t}$ where l is the first pipeline and d is the starting terminal of route r , there will be an arc $NodePOut_{p,d,t} \rightarrow NodePipe_{r,l,p,t,t}$. The maximum capacity of this arc is $maxTransfer_{r,p,0,1}$ which measures the maximum quantity of product p that can be injected into pipeline l over a whole day. These arcs are the leftmost ones shown in Figure 5.
- Intermediary Stock Arcs:** For each node $NodePipe_{r,l,p,ti,t}$ there will be an arc $NodePipe_{r,l,p,ti,t} \rightarrow NodePipe_{r,l,p,ti,t+1}$ with capacity given by the pipeline volume $vol(l)$. These arcs are the vertical ones in Figures 4 and 5.

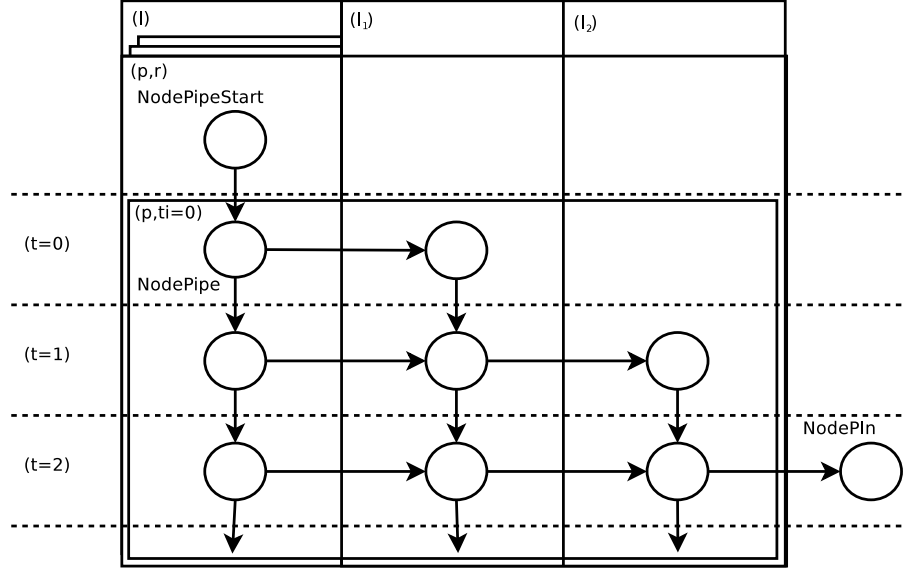


Figure 4: Nodes and arcs for the stock of a pipeline with contents (p, qty, r) . Items within parenthesis index the nodes enclosed inside the rectangle. The occluded rectangles represent the multiplicity of that network with regard to the inner index.

- Route Arcs:** For each pair of nodes $NodePipe_{r,l_1,p,ti,t}$ and $NodePipe_{r,l_2,p,ti,t}$ where l_1 and l_2 are sequential segments in route r , there will be an arc $NodePipe_{r,l_1,p,ti,t} \rightarrow NodePipe_{r,l_2,p,ti,t}$ between them. If x is the index of pipeline l_2 in route r , the maximum capacity of this arc is calculated as $maxTransfer_{r,p,x,t-ti+1}$. This is the same as the maximum quantity of product p that can be transferred out of pipeline l_1 starting at period ti and ending at period t from the beginning of route r . Figures 4 and 5 show these arcs connecting adjacent nodes that are inside rectangles representing pipelines.
- Incoming Terminal Arcs:** For each node $NodePipe_{r,l,p,t,t}$ where l is the last segment and d is the last terminal of route r , there will be an arc $NodePipe_{r,l,p,t,t} \rightarrow NodePln_{p,d,t}$. If $n - 1$ is the index of segment l in route r , the maximum capacity of this arc is calculated as $maxTransfer_{r,p,n,t-ti+1}$. This is the same as the maximum quantity that can be transferred to terminal depot d starting at period ti and ending at period t along route r . These arcs are the rightmost ones in Figure 5.
- Pipeline Terminal Stock Arcs:** For each node $NodePipeEnd_l$, with T the last period, there will be an arc $NodePipe_{r,l,p,ti,T-1} \rightarrow NodePipeEnd_l$ with maximum capacity equal to $vol(l)$.

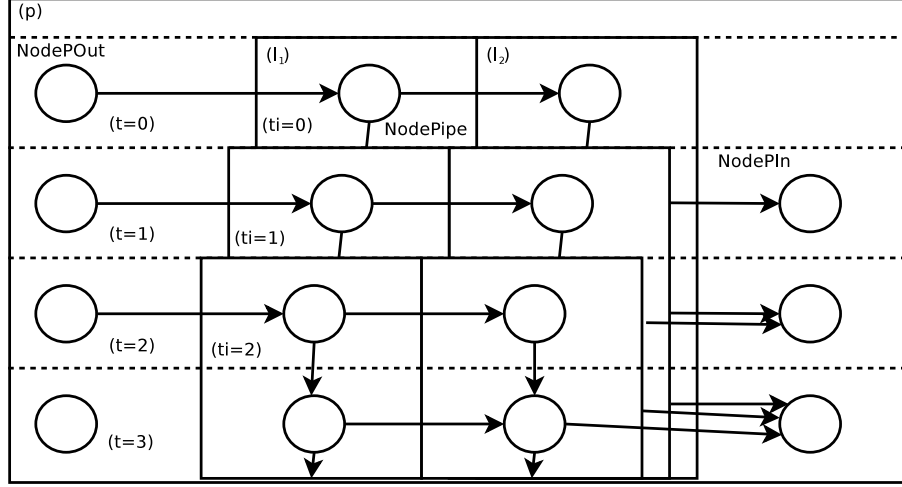


Figure 5: Nodes and arcs representing volumes traveling along routes. For a fixed route r and product p , at every period ti a new subnetwork is created so that the quantity will only be delivered after enough periods have passed. The subnetwork for period $t = 3$ was omitted to show more of the subnetwork for period $t = 2$. Items within parenthesis index the nodes enclosed inside the rectangle. The occluded rectangles represent the multiplicity of that network with regard to the inner index.

3.2 A Linear Programming Model

In order to define the constraints and the objective function of the model we need the set of nodes at period t , $Nodes_t$, the set of nodes along pipeline l at period t , $Nodes_{l,t}$, and the set of all nodes for product p at pipeline l and at period t , $Nodes_{p,l,t}$. Also, DP , P , and PL will indicate the set of all depots, products, and pipelines respectively.

The following model is not strictly a network flow model. Nevertheless, the problem structure itself retains many features of a Multi-commodity Multi-period Network Flow Model and a path decomposition algorithm will be described [20] later on.

Nodes Mass Balance

$$\sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} x_{ji} = b_i + Slack_i \quad (1)$$

$$\forall i \in Nodes_t, \quad 0 \leq t < T$$

We consider that the pipeline initial and terminal stock nodes are all in the $Nodes_0$ set. Also, this constraint is not applied to tank nodes in the last period. This is necessary in order to model what will be in stock inside tanks after the end of the horizon. The $Slack_i$

variable will only be present when one wants to know which stocks cannot be satisfied. In this case, if node i is a production node we define $Slack_i = -InsatisfiedProduction_i$, and for a demand node i we let $Slack_i = InsatisfiedDemand_i$.

Arc Bounds

$$l_{ij} \leq x_{ij} \leq u_{ij}, \quad \forall (i, j) \in A \quad (2)$$

Pipelines must always remain full

$$\sum_{j:(i,j) \in A} x_{ij} = vol(l) \quad (3)$$

$$\forall i \in Nodes_{l,t}, \quad l \in PL, \quad 0 \leq t < T$$

An amount injected at one end equals the amount delivered at the other end

$$\sum_{\substack{j:(i,j) \in A \\ Deliver(R,i,j)}} x_{ij} = \sum_{\substack{j:(j,i) \in A \\ Inject(P,j,i)}} x_{ji} \quad (4)$$

$$\forall i \in Nodes_{l,t}, \quad l \in PL, \quad 0 \leq t < T$$

$$\sum_{\substack{j:(i,j) \in A \\ Deliver(P,i,j)}} x_{ij} = \sum_{\substack{j:(j,i) \in A \\ Inject(R,j,i)}} x_{ji} \quad (5)$$

$$\forall i \in Nodes_{l,t}, \quad l \in PL, \quad 0 \leq t < T$$

The functions $Inject(e, j, i)$ and $Deliver(e, i, j)$ tell if arcs (j, i) and (i, j) inject or deliver volumes, respectively, through extremity e in the pipeline represented by node i .

Pipelines Flow Rates

$$\sum_{\substack{j:(i,j) \in A \\ Inject(R,i,j)}} x_{ij} \leq maxQtyInject_{R,p,l} \quad (6)$$

$$\forall i \in Nodes_{p,l,t}, \quad p \in P, \quad l \in PL, \quad 0 \leq t < T$$

$$\sum_{\substack{j:(i,j) \in A \\ Inject(\mathbf{P},i,j)}} x_{ij} \leq \maxQtyInject_{\mathbf{P},p,l} \quad (7)$$

$$\forall i \in Nodes_{p,l,t}, \quad p \in P, \quad l \in PL, \quad 0 \leq t < T$$

$$\sum_{\substack{j:(i,j) \in A \\ Inject(\mathbf{R},i,j)}} x_{ij} + \sum_{\substack{j:(i,j) \in A \\ Inject(\mathbf{P},i,j)}} x_{ij} \leq \maxQtyInject_{p,l} \quad (8)$$

$$\forall i \in Nodes_{p,l,t}, \quad p \in P, \quad l \in PL, \quad 0 \leq t < T$$

$$\sum_{\substack{j:(i,j) \in A \\ Inject(\mathbf{R},i,j)}} x_{ij} \leq \maxQtyInject_{\mathbf{R},l} \quad (9)$$

$$\forall i \in Nodes_{l,t}, \quad l \in PL, \quad 0 \leq t < T$$

$$\sum_{\substack{j:(i,j) \in A \\ Inject(\mathbf{P},i,j)}} x_{ij} \leq \maxQtyInject_{\mathbf{P},l} \quad (10)$$

$$\forall i \in Nodes_{l,t}, \quad l \in PL, \quad 0 \leq t < T$$

$$\sum_{\substack{j:(i,j) \in A \\ Inject(\mathbf{R},i,j)}} x_{ij} + \sum_{\substack{j:(i,j) \in A \\ Inject(\mathbf{P},i,j)}} x_{ij} \leq \maxQtyInject_l \quad (11)$$

$$\forall i \in Nodes_{l,t}, \quad l \in PL, \quad 0 \leq t < T$$

Depot Injection Limit

$$\sum_{(i,j) \in ArcsOutD_{d,t}} x_{ij} \leq \maxDepotQtyInject_d \quad (12)$$

$$\forall d \in DP, \quad 0 \leq t < T$$

The set $ArcsOutD_{d,t}$ contains all the arcs that start in $NodePOut_{p,d,t}$, for all products in depot d and at period t .

Stock Levels

$$\sum_{(i,j) \in \text{ArcsOutTankEnd}_{d,p,t}} x_{ij} \geq \text{stockLevelMin}_{d,p} \quad (13)$$

$$\forall d \in DP, \forall p \in P, \quad \forall t \in WE$$

$$\sum_{(i,j) \in \text{ArcsOutTankEnd}_{d,p,t}} x_{ij} \leq \text{stockLevelMax}_{d,p} \quad (14)$$

$$\forall d \in DP, \forall p \in P, \quad \forall t \in WE$$

The set $\text{ArcsOutTankEnd}_{d,p,t}$ contains all arcs connecting node $\text{NodeEndTk}_{d,tk,t}$ to node $\text{NodeStartTk}_{d,tk,t+1}$, for every tank tk at depot d with product p and at period t . These are the arcs that transmit stocks between periods. The set WE contains the periods where stock levels should be verified. It should be of the form $\{6, 13, 20, 27 \dots\}$, representing a weekly check.

Objective Function When engineers want to check if a proposed production plan is feasible against the forecasted demands they use the following objective function:

$$\text{minimize} \left(\sum_{i \in \text{NodesPro}} \text{InsatisfiedProduction}_i + \sum_{i \in \text{NodesDem}} \text{InsatisfiedDemmand}_i \right), \quad (15)$$

where the sets NodesPro and NodesDem contain the all production and demand nodes, respectively. The solution obtained with this objective function can also be used to correct production or demand values. This correction deals with many economical and operational issues that are not discussed here.

After finding a feasible set, the next objective function to be used is the classic one:

$$\text{minimize} \sum_{(i,j) \in A} x_{ij} \quad (16)$$

Together with the constraints, we have a complete model for the planning phase. The next step is to extract the orders using a modified decomposition algorithm.

3.3 Network Flow Decomposition

Figures 3, 4 and 5 show no bidirectional arcs and no paths going back to a previously visited node. So, no cycle can be generated within the network. This observation allows us to use a specialized version of the flow decomposition algorithm [20]. We also introduced some modifications that are specific to our problem, as described in Algorithm 3.

The main purpose of the preferences mentioned in SELECT and SEARCH is to make it easier to supply a demand by considering the longest time between the moment it is needed and the moment it is available. These heuristics do not affect the result stating that the flow can be fully decomposed into paths.

Let $\mathcal{P}'$ the set of paths and let $f(P)$ be the flow associated to a path $P \in \mathcal{P}'$. We create the necessary orders that can be input to the operational scheduling solver described in [3] by analyzing the nodes traversed by P . An order O will be represented by the tuple

$$O = (OriginTank, DestinationTank, Product, \\ Volume, Route, Deadline).$$

A path P is built following the inverse order with respect to the network timeline. This way it will always start at a demand node, a tank node or a pipeline stock node at the last period. The deadline is either the demand node period or the last period when the node corresponds to tank or pipeline stocks. A path P should always end at a production node, tank initial node or pipeline initial stock node. Sometimes the path will not contain any pipeline nodes. Such is the case when a demand is satisfied by a production inside the same depot. In these cases, the route will be null. Conversely, the path P can contain nodes present in more than one route. In this case, one must create as many orders as there are pipeline nodes in P , because each order has exactly one route associated to it.

The origin and destination tanks should be the ones connecting to pipeline nodes. If there is no route, the two tank variables will have the same value. Finally, the volume is given by the flow $f(P)$ associated with the path.

A decomposition of the network can generate a huge number of orders with small volumes. It is interesting, then, to aggregate similar orders. If two orders have the same origin tank, destination tank, product, route, and close deadlines, they can be merged into one order with their volumes summed up. Two deadlines are close if they happen within a range of days. The horizon can be divided in as many ranges as necessary. In our case, it was divided in ten ranges of three days each, generating a set of orders with adequate size.

Algoritmo 3 Pipeline Network Flow Decomposition Algorithm

Entrada: $G = (N, A)$ with arc flow x **Saída:** $\mathcal{P}$ with $f(P), \forall P \in \mathcal{P}$

NOTATION:

 y - flow working copy $A(y) = \{(i, j) \in A | y_{ij} > 0\}$ (Arcs with positive flow in y) $N(y) = \{i | (i, j) \in A(y) \text{ or } (j, i) \in A(y)\}$ (Nodes incident to arcs in $A(y)$) $G(y) = (N(y), A(y))$ $\mathcal{S} = \{i \in N(y) | b_i > 0\}$ (supply nodes) $\mathcal{D} = \{i \in N(y) | b_i < 0\}$ (demand nodes) s and t are the start and end nodes of path P . $\Delta(P) = \min\{b(s), -b(t), \min\{y_{ij} | (i, j) \in P\}\}$ (Capacity of path P)

```

1: procedimento PIPELINEFLOWDECOMPOSITION
2:    $y = x, \mathcal{P} = \emptyset$ 
3:   enquanto  $A(y) \neq \emptyset$  faça
4:      $s = \text{SELECT}(y)$ 
5:      $\text{SEARCH}(s, y)$ 
6:     se Path  $P$  found então
7:        $\mathcal{P} = \mathcal{P} \cup \{P\}$ 
8:        $f(P) = f(P) + \Delta(P)$ 
9:        $y_{ij} = y_{ij} - \Delta(P) \forall (i, j) \in P$ 
10:       $b(s) = b(s) - \Delta(P)$ 
11:       $b(t) = b(t) - \Delta(P)$ 
12:     fim se
13:     Update  $A(y), N(y), \mathcal{S}, \mathcal{D}$ 
14:   fim enquanto
15: fim procedimento
16: função  $\text{SELECT}(y)$ 
17:   Order nodes in  $\mathcal{S}$  first by the most pressing demands;
   secondly, by those tank nodes at the last period, and
   lastly, by pipeline terminal stock nodes.
18:   retornar the first node in  $\mathcal{S}$  by the given ordering.
19: fim função
20: função  $\text{SEARCH}(s, y)$ 
21:   Do a DFS starting with node  $s$  until a path  $P$  is found in  $G(y)$  ending at node
    $t \in \mathcal{D}$ .
22:   Inside the DFS, before choosing the next node, order the candidate nodes
   firstly by being of the same tank, if applicable,
   then by those nodes being in the same time period.
23:   retornar The path  $P$  found.
24: fim função

```

Instance	Rows	Columns	Non-zeros	Model Time (s)	Decomp. Time (s)	Orders
1	23801	51948	266778	22.1	3.3	700
2	22935	49788	256470	19.5	3.2	685
3	22346	48607	248310	18.4	3.5	700
4	22976	50494	257631	17.9	4.1	731
5	23237	50711	260774	20.3	4.5	759
6	24532	54414	281496	23.0	4.0	841
7	23079	50448	258469	20.3	3.4	826
8	24355	53797	278574	28.0	4.2	897
9	24462	53152	273019	22.9	3.7	776
10	25948	57326	295073	26.8	3.9	796
11	25536	56357	290854	26.5	4.0	818
12	21554	46487	236411	17.6	3.9	727
Mean	23730	51960	266988	21.9	3.8	771

Table 3: Execution Results

Instance	1	2	3	4	5	6	7	8	9	10	11	12	Mean
Proportion	0.72	0.69	0.80	0.71	0.81	0.82	0.83	0.76	0.72	0.67	0.69	0.73	0.75 ± 0.05

Table 4: Manual planning vs Real planning - Total Network Flow Proportion

4 Computational Results

Solutions were obtained on a *Intel Pentium Core 2 Duo* 2.1 GHz CPU platform, with 3MB L2 Cache and 4GB RAM. The program was coded in C++ and compiled using *GCC-4.2* without optimization. The network flow model used the CPLEX 10 solver with presolving columns and row elimination. Among the linear programming solving techniques available, the barrier method was selected as it gave the best execution times.

All instances refer to the same network topology, with 19 pipelines, as depicted in Figure 2. The number of tanks vary from 0 to 20 at each depot, with a total of 192 tanks. Also, there are 11 products that circulate through the network.

The model was tested against 12 real instances, spanning a whole year of planning, one instance per month. Some characteristics of these instances cannot be disclosed, as they could expose classified data.

From Table 3 it can be seen that the whole execution takes around 20 seconds. This makes the procedure suitable for testing “what-if” scenarios. The number of orders per

Product	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	Mean
Proportion	0.91	0.65	1.05	0.73	1.05	0.79	0.92	0.83	0.58	0.15	0.56	0.75 ± 0.26

Table 5: Manual planning vs Real planning - Total Network Flow Proportion by Product

Pipeline Proportion	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}	
	0.27	0.85	0.65	0.95	0.91	0.85	1.05	1.00	0.34	0.82	
Pipeline Proportion	D_{11}	D_{12}	D_{13}	D_{14}	D_{15}	D_{16}	D_{17}	D_{18}	D_{19}		Mean
	0.58	0.64	0.70	0.66	1.04	1.14	0.78	0.18	0.91		0.75 ± 0.27

Table 6: Manual planning vs Real planning - Total Network Flow Proportion by Pipeline

month never exceed 900, and so, roughly, it is necessary to schedule 30 orders per day in order to reach the planning goal.

PETROBRAS also provided data on manually generated plans for each of the instances. In Table 4, the total volume scheduled to be transferred by a real planning is compared against the one proposed by the model. The mean proportion is 0.75, meaning that the model transfers 25% less units of volume each month on the average. This represents a quite significant economy when multiplied by the very large volumes that are moved each month. It must be noted that the model does transfer what is necessary in order to satisfy every demand and keep the stock levels in safe conditions. Thus, these savings in product shipments were not obtained at the expense relaxing some problem constraints.

One reason for the 25% reduction could be related to the choice of products used to push others along pipelines, when really necessary. Sometimes, the products used to push others out of a pipeline are not directly useful for the planning goals. This incurs in wasted pumping moves. Examining Table 5 one can see that the differences in product choices are high. As an example, product P_{10} has a proportion of only 0.15.

Another important aspect of this issue is the choice of routes. A refinery has many outgoing routes and, so, poor choices of routes along which products will be set to flow along may also lead to wastes. Table 6 shows that pipelines D_1 , D_9 and D_{18} are rarely used. Hence, there will be less need to push products out of them.

The objective was to minimize the total volume transported across the network. This means that a product will be moved only to satisfy a constraint. Engineers have the same objective but, having solely on their past experiences to rely upon, and having very limited time to workout a solution, they will accept any solution that keeps wastes below an “acceptable” level. The threshold where the solution is acceptable might be related to the 25% reduction in transported volumes.

5 Conclusions and Future Work

This paper described the oil pipeline planning problem faced by PETROBRAS, a Brazilian oil company. The operation of its pipeline network involves many constraints related to tanks, volumes, pipeline utilization, stock levels as well as demand and production sched-

ules. There is also a large set of complex operational constraints that must be satisfied by feasible schedules [3]. The objective was to obtain a minimum cost transportation plan for all oil derivatives and ethanol, given that the specified demands and productions are adequate to meet the constraints. Although the real network operation requires a daily pumping schedule, here the problem is solved at the tactical level, where the horizon spans one or more months. See [3] for a scheduler that takes as input the tactical plan constructed by our algorithm and outputs a detailed daily schedule.

In its fullest, the problem can only be solved for a span of a few days. In order to make it more tractable, only the most important characteristics were included in our formal definition of the problem. Engineers validated these characteristics as the ones having the greatest impact over the network daily operation. An important characteristic requires that all pipelines must always remain completely full, and tanks should keep the local stock within their capacities. Transfers had to observe permitted pipeline flow rates, which can vary by pipeline, by product and by flow direction. Productions and demands were satisfied on a daily basis. Stock levels were verified weekly. Another important constraint was that volumes should follow along predefined routes that depend on product type.

A model to solve this problem was proposed using network flow ideas. Nodes were defined so as to model the state of each depot. Pipelines were modeled by classifying the nodes that represent each route that a product can travel. The model could also be used to test if the proposed demand and production schedules were feasible or, else, to give a minimum cost transportation plan for feasible movements.

Test results showed that the model can adequately deal with the PETROBRAS network topology and with monthly scenarios, within small computational times. Also, the solutions generated were compared to the ones proposed by the company expert engineers. It was found that the model almost always gave a 25% cost reduction. Tentative interpretations for this gain were listed.

Although the model already included most of constraints the engineers consider when managing the tactical planning problem, more constraints could be considered. One of them could be to model what happens when two products that cannot make contact are to be injected in a pipeline. In this case, a third product must be used to separate them. Also, since sometimes pipeline and tank maintenance periods are known beforehand, such constraints could be incorporated into the model. The algorithms presented here were developed together with the work described in [3]. Finally, the tactical plan it outputs could be used as input to the daily scheduler discussed in that same work.

Acknowledgments This research was supported by grants 05/57343-0 and 05/57344-7 from FAPESP and grants 301732/07-8, 478470/06-1, 472504/07-0, and 473726/07-6,

305781/2005-7 from CNPq. The authors are also grateful to Fernando Marcellino and the team of engineers from PETROBRAS-TI/SP.

References

- [1] Dempster, M.A.H., Pedron, N.H., Medova, E.A., Scott, J.E., Sembos, A.: Planning logistics operations in the oil industry. *The Journal of the Operational Research Society* **51**(11) (2000) 1271 – 1288
- [2] Milidíu, R., dos Santos Liporace, F., de Lucena, C.J.P.: Pipesworld: Planning pipeline transportation of petroleum derivatives. In: *Proceedings of ICAPS'03 - Workshop on the Competition: Impact, Organization, Evaluation, Benchmarks*, Trento, Italy (2003)
- [3] Lopes, T.M., Ciré, A.A., Souza, C.C., Moura, A.V.: Special issue on the 14th international conference on principles and practice of constraint programming. *Constraints* **15**(2) (2010)
- [4] Herran, A., de la Cruz, J., de Andres, B.: A mathematical model for planning transportation of multiple petroleum products in a multi-pipeline system. *Computers & Chemical Engineering* **34**(3) (2010) 401 – 413
- [5] Moura, A.V., Pereira, R.A., de Souza, C.C.: Scheduling activities at oil wells with resource displacement. *International Transactions in Operational Research* **15**(6) (2008) 659–683
- [6] Maravelias, C.T., Sung, C.: Integration of production planning and scheduling: Overview, challenges and opportunities. *Computers & Chemical Engineering* **33**(12) (2009) 1919 – 1930 *FOCAPO 2008 - Selected Papers from the Fifth International Conference on Foundations of Computer-Aided Process Operations*.
- [7] Alves, V., Filho, V.J.: Pipeline scheduling of petroleum derivatives using genetic algorithm. In: *IV Congresso Brasileiro de Pesquisa e Desenvolvimento em Petróleo e Gás*, Campinas, Brazil (2007)
- [8] Cafaro, D.C., Cerdá, J.: Optimal scheduling of multiproduct pipeline systems using a non-discrete MILP formulation. *Computers & Chemical Engineering* **28**(10) (2004) 2053–2058
- [9] Rejowski, R., Pinto, J.M.: A novel continuous time representation for the scheduling of pipeline systems with pumping yield rate constraints. *Computers & Chemical Engineering* **32** (2008) 1042–1066
- [10] Filho, E.M.S., Filho, V.J., de Lima, L.S.: Variable neighborhood search (VNS) applied to pipeline distribution problem with capacity constraints. In: *IV Congresso Brasileiro de Pesquisa e Desenvolvimento em Petróleo e Gás*, Campinas, Brazil (2007)

- [11] Camponogara, E., Souza, P.S.: A-Teams for oil transportation problem through pipelines. In: Information Systems Analysis and Synthesis, Orlando, United States (1996)
- [12] Camponogara, E.: A-Teams para um problema de transporte de derivados de petróleo. Master's thesis, Instituto de Matemática, Estatística e Ciência da Computação, Universidade Estadual de Campinas, Campinas, Brazil (1995)
- [13] de la Cruz, J., Andrés-Toro, B., Herrán-González, A., Porta, E.B., Blanco, P.F.: Multiobjective optimization of the transport in oil pipelines networks. In: Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation. Volume 1. (2003) 566–573
- [14] de la Cruz, J., Herrán-González, A., Risco-Martín, J., Andrés-Toro, B.: Hybrid heuristic and mathematical programming in oil pipelines networks: Use of immigrants. Journal of Zhejiang University SCIENCE **6A**(1) (2005) 9–19
- [15] Magatão, L., Arruda, L., Neves, F.: A mixed integer programming approach for scheduling commodities in a pipeline. Computers & Chemical Engineering **28**(1) (2004) 171–185
- [16] Magatão, L., Arruda, L., Neves, F.: Using CLP and MILP for scheduling commodities in a pipeline. Computer-Aided Chemical Engineering **20B** (2005) 1027–1032
- [17] Relvas, S., Barbosa-Póvoa, A.P.F.D., Matos, H.A., Fialho, J., Pinheiro, A.S.: Pipeline scheduling and distribution centre management - a real-world scenario at CLC. In: Proceedings of the 16th European Symposium on Computer Aided Process Engineering and 9th International Symposium on Process Systems Engineering, Garmisch-Partenkirchen, Germany (2006) 2135–2140
- [18] Relvas, S., Matos, H.A., Barbosa-Póvoa, A.P.F.D., Fialho, J., Pinheiro, A.S.: Pipeline scheduling and inventory management of a multiproduct distribution oil system. Industrial & Engineering Chemistry Research **45**(23) (2006) 7841–7855
- [19] Relvas, S., Matos, H.A., Barbosa-Póvoa, A.P.F.D., Fialho, J.: Reactive scheduling framework for a multiproduct pipeline with inventory management. Industrial & Engineering Chemistry Research **46**(17) (2007) 5659–5672
- [20] Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network flows : theory, algorithms, and applications / Ravindra K. Ahuja, Thomas L. Magnanti, James B. Orlin. Prentice Hall, Englewood Cliffs, N.J. : (1993)

- [21] Braconi, V.M.: Heurísticas multifluxo para roteamento de produtos em redes dutoviárias. Master's thesis, Departamento de Informática, PUC-Rio, Rio de Janeiro, RJ, Brasil (2002)

Epílogo

A definição formal do problema de planejamento é a mesma do **PPAORO**, removendo-se um subconjunto de restrições. Nos tanques, C3, C4 e C5 foram desconsideradas pois a simultaneidade de operações ou a sequencia delas não é importante quando não se detalha um dia. As restrições C7, C8, C9 e C10 foram consideradas pelos operadores no cálculo das quantidades máximas que um órgão pode enviar de cada produto e durante um período. No dutos, C13, mesmo não sendo considerada, não foi violada em nenhuma solução devido a agregação de planos de entrega. A restrição C15 foi desconsiderada pela falta de dados, mas também poderia entrar nos calculos de quantidade de envio máxima por período. Já as restrições de incompatibilidade para o sequenciamento nos dutos C14 não foram consideradas, pois não são importantes e grande parte das incompatibilidades são pré-resolvidas na criação do conjunto de produtos e seleção das rotas permitidas.

O modelo de fluxo em redes para o problema de planejamento é construído sobre os tanques e, principalmente, sobre os pares de produtos e rotas pré-definidos. Essa escolha permite uma modelagem compacta, onde não serão criados movimentos desnecessários ou improváveis. A rede criada separa os nós em camadas, representando a evolução do tempo. Em cada período, são criados os nós que representam a movimentação nos tanques, o atendimento de demandas, o escoamento das produções e as entradas e saídas de uma base. O tráfego de volumes pela rede possui uma representação especial, pois para implementar uma simulação do transporte de um volume por uma rota é necessário saber em qual período ele foi enviado. Isso faz com que a cada novo período, seja necessária uma rede a mais, do que no período anterior, para representar o tráfego por rotas.

A estrutura de rede criada é livre de ciclos, o que permite a simplificação do algoritmo de decomposição de fluxo em redes. Esse algoritmo extrai a partir do fluxo obtido, os planos de entrega que servirão de entrada para a fase de agendamento. Um pós-processamento sobre esse conjunto de planos foi realizado para reduzir o seu tamanho. A agregação de planos é útil para facilitar o agendamento e atender requisitos de tamanho mínimo de bateladas.

Uma outra motivação do artigo é a utilização do modelo para o planejamento tático da movimentação nos oleodutos. A escolha de um modelo de fluxo em redes, e que fosse possível implementar através de uma programação linear, foi feita principalmente para manter o tempo de execução controlável. Os testes demonstraram que para a rede apresentada pela PETROBRAS é possível obter soluções em poucos minutos para horizontes de um mês. Isso permite a execução de cenários condicionais para, por exemplo, verificar se é possível atender as demandas com uma nova proposta de produção das refinarias.

Os resultados, quando comparados a soluções manuais do planejamento tático, mostraram custos 25% menores para o atendimento dos mesmos níveis de estoque. Isso demonstra que

a ferramenta desenvolvida pode ser útil, mesmo sem a execução da fase de agendamento.

Trabalhos futuros incluem a integração dessa nova modelagem para o planejamento com o modelo de programação por restrições do agendamento. Como foi visto no capítulo 6, o conjunto de planos gerado pela abordagem heurística tem em torno de 900 planos de entrega para 10 dias de horizonte, enquanto na abordagem de fluxo em redes o mesmo número de planos é suficiente para 30 dias. Além disso, ao verificar a validade do estado da rede diariamente, entende-se que o conjunto de novos planos, gerados em novas ativações do algoritmo, será mais facilmente sequenciado e escalonado.

Capítulo 8

Conclusões

Esta dissertação trata o problema de distribuição de derivados de petróleo e álcoois em redes de duto. Também denominado **PPAORO**, o problema é atualmente enfrentado pela PETROBRAS.

O objetivo é definir movimentos de transporte de produtos entre órgãos, de tal forma que as campanhas de produção e os mercados locais, representados por valores estimados de demandas, sejam totalmente satisfeitos. Para tanto, um amplo conjunto de restrições operacionais sobre as movimentações em dutos e tanques deve ser respeitado, envolvendo seqüenciamentos válidos de produtos, capacidade de tanques e limites de envio simultâneo.

Até então, modelos para o **PPAORO** pressupunham fortes relaxações de seus principais requisitos, ou lidavam apenas com topologias de rede restritas, sendo assim, tratáveis eficientemente com técnicas de otimização clássicas. Por outro lado, tais relaxações deram origem a soluções pouco aplicáveis para os cenários reais, pois descartavam restrições fortes para os operadores de duto, tais como interface de produtos e não-simultaneidade de operações em tanques.

O trabalho aqui desenvolvido propõe formulações capazes de representar a maior parte das restrições fundamentais do **PPAORO**, com potencial para serem utilizadas em instâncias de tamanho real. Para tal, a modelagem é dividida em duas fases, *planejamento* e *agendamento*.

Uma formulação em *Programação por Restrições* para a fase de agendamento do **PPAORO** foi desenvolvida. Tal técnica foi utilizada devido à sua flexibilidade em modelar problemas de agendamento e por conta dos fortes mecanismos de propagação existentes, os quais permitem encontrar soluções factíveis rapidamente.

Os primeiros resultados utilizaram uma versão heurística da fase de planejamento, resultando na publicação de um artigo científico no periódico internacional *Constraints* [27], um dos mais importantes da área de programação por restrições.

Em seguida, como uma evolução do tratamento da fase de planejamento, um artigo foi

submetido ao periódico internacional *Computers & Chemical Engineering* apresentando uma modelagem usando fluxo em redes. O artigo formaliza o problema de planejamento e propõe um modelo linear. Tal modelo foi capaz de gerar as entradas para a fase de agendamento, bem como soluções para o problema tático de planejamento como descrito no artigo.

Como trabalhos futuros, um primeiro passo poderia ser a integração dessa nova abordagem do planejamento com a fase de agendamento. Após isso, poder-se-ia acrescentar novas restrições a ambos os modelos, bem como realizar testes em diferentes topologias da própria PETROBRAS.

Referências Bibliográficas

- [1] Ravindra K. Ahuja, Thomas L. Magnanti, e James B. Orlin. *Network flows : theory, algorithms, and applications / Ravindra K. Ahuja, Thomas L. Magnanti, James B. Orlin*. Prentice Hall, Englewood Cliffs, N.J. :, 1993.
- [2] Vanessa Alves e Virgílio J.M.F. Filho. Pipeline scheduling of petroleum derivatives using genetic algorithm. *IV Congresso Brasileiro de Pesquisa e Desenvolvimento em Petróleo e Gás*, Campinas, Brazil, 2007.
- [3] Vanessa Rennó Frota Moraes Alves. Programação de transferência de derivados de petróleo em rede dutoviária usando algoritmo genético. Dissertação de Mestrado, COPPE - Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ, Brasil, 2007.
- [4] R. Barták. Constraint programming: In pursuit of the holy grail. *Week of Doctoral Students (WDS99), Part IV, MatFyzPress, Prague*, páginas 555–564, June de 1999.
- [5] M.S. Bazaraa, J.J. Jarvis, e H.D. Sherali. *Linear Programming and Network Flows*. John Wiley and Sons, 1990.
- [6] Viviane Monteiro Braconi. Heurísticas multifluxo para roteamento de produtos em redes dutoviárias. Dissertação de Mestrado, Departamento de Informática, PUC-Rio, Rio de Janeiro, RJ, Brasil, 2002.
- [7] Diego C. Cafaro e Jaime Cerdá. Optimal scheduling of multiproduct pipeline systems using a non-discrete MILP formulation. *Computers & Chemical Engineering*, 28(10):2053–2058, 2004.
- [8] E. Camponogara e P. S. Souza. A-Teams for oil transportation problem through pipelines. *Information Systems Analysis and Synthesis*, Orlando, United States, 1996.
- [9] Eduardo Camponogara. A-Teams para um problema de transporte de derivados de petróleo. Dissertação de Mestrado, Instituto de Matemática, Estatística e Ciência da Computação, Universidade Estadual de Campinas, Campinas, Brazil, 1995.

- [10] André Augusto Ciré. Modelos computacionais para o escalonamento de tarefas em redes de dutos. Dissertação de Mestrado, Instituto de Computação, Universidade Estadual de Campinas, Campinas, SP, Brasil, 2008.
- [11] 5th International Planning Competition. <http://zeus.ing.unibs.it/ipc-5/>.
- [12] Thomas H. Cormen, Charles E. Leiserson, e Ronald L. Rivest. *Introduction to Algorithms*. MIT Press, 2 edição, 2001.
- [13] D. Scott Crane, Roger L. Wainwright, e Dale A. Schoenefeld. Scheduling of multi-product fungible liquid pipelines using genetic algorithms. *Proceedings of the 1999 ACM Symposium on Applied Computing*, páginas 280–285, San Antonio, USA, 1999.
- [14] G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1963.
- [15] J.M. de la Cruz, B. Andrés-Toro, A. Herrán-González, E. Besada Porta, e P. Fernandez Blanco. Multiobjective optimization of the transport in oil pipelines networks. *Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation*, volume 1, páginas 566–573, 2003.
- [16] J.M. de la Cruz, A. Herrán-González, J.L. Risco-Martín, e B. Andrés-Toro. Hybrid heuristic and mathematical programming in oil pipelines networks: Use of immigrants. *Journal of Zhejiang University SCIENCE*, 6A(1):9–19, 2005.
- [17] Érito Marques de Souza Filho, Vanessa Rennó Frota Moraes, e Virgílio José Martins Ferreira Filho. Utilização de técnicas de pesquisa operacional em problemas de distribuição dutoviária: Uma revisão. *XXXVIII Simpósio Brasileiro de Pesquisa Operacional*, páginas 1873–1880, Goiania, Brazil, 2006.
- [18] R. Dechter e D. Frost. Backtracking algorithms for constraint satisfaction problems; a survey. Relatório técnico, University of California at Irvine, 1999.
- [19] Frederico dos Santos Liporace. *Planejadores para transporte em polidutos*. Tese de Doutorado, Departamento de Informática, PUC-Rio, Rio de Janeiro, RJ, Brasil, 2005.
- [20] Erito M. Souza Filho, Virgílio J.M.F. Filho, e Leonardo S. de Lima. Variable neighborhood search (VNS) applied to pipeline distribution problem with capacity constraints. *IV Congresso Brasileiro de Pesquisa e Desenvolvimento em Petróleo e Gás*, Campinas, Brazil, 2007.

- [21] I. Gent e B. Smith. Symmetry breaking during search in constraint programming. *Proceedings of ECAI'2000*, páginas 599–603, 2000.
- [22] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.
- [23] A. Herran, J.M. de la Cruz, e B. de Andres. A mathematical model for planning transportation of multiple petroleum products in a multi-pipeline system. *Computers & Chemical Engineering*, 34(3):401 – 413, 2010.
- [24] John N. Hooker. *Integrated Methods for Optimization (International Series in Operations Research & Management Science)*. Springer-Verlag, Secaucus, USA, 2006.
- [25] ILOG. *ILOG Scheduler 6.2: User's Manual*. ILOG, 2006.
- [26] R. Rejowski Jr. e José M. Pinto. An MILP formulation for the scheduling of multi-product pipeline systems. *Brazilian Journal of Chemical Engineering*, 19(4):467–474, 2002.
- [27] Tony M. Lopes, Andre A. Ciré, Cid C. Souza, e Arnaldo V. Moura. Special issue on the 14th international conference on principles and practice of constraint programming. *Constraints*, 15(2), 2010.
- [28] M. M. Sasikumar, P. R. Prakash, S. M. Patil, e S. Ramani. Pipes: A heuristic search model for pipeline schedule generation. *Knowledge-Based Systems*, 10(3):169–175, 1997.
- [29] L. Magatão, L.V.R. Arruda, e F. Neves. A mixed integer programming approach for scheduling commodities in a pipeline. *Computers & Chemical Engineering*, 28(1):171–185, 2004.
- [30] L. Magatão, L.V.R. Arruda, e F. Neves. Using CLP and MILP for scheduling commodities in a pipeline. *Computer-Aided Chemical Engineering*, 20B:1027–1032, 2005.
- [31] K. Marriot e P.J. Stuckey. *Programming with Constraints: An Introduction*. MIT Press, Cambridge, Massachusetts, 1ª edição, 1998.
- [32] Rodrigo Más e José M. Pinto. A mixed-integer optimization strategy for oil supply in distribution complexes. *Optimization and Engineering*, 4(1):23–64, 2003.
- [33] R.L. Milidíu e Frederico dos Santos Liporace. Planning of pipeline oil transportation with interface restrictions is a difficult problem. Relatório Técnico 56, Departamento de Informática, PUC-Rio, Rio de Janeiro, RJ, Brasil, 2003.

- [34] R.L. Milidíu, Frederico dos Santos Liporace, e Carlos José P. de Lucena. Pipesworld: Planning pipeline transportation of petroleum derivatives. *Proceedings of ICAPS'03 - Workshop on the Competition: Impact, Organization, Evaluation, Benchmarks*, Trento, Italy, 2003.
- [35] R.L. Milidíu, A.A. Pessoa, V. Braconi, E.S. Laber, e Rey P.A. Um algoritmo GRASP para o problema de transporte de derivados de petróleo em oleodutos. *Proceedings of the XXXIII Brazilian Symposium on Operations Research*, páginas 237–246, 2001.
- [36] S.A. MirHassani e M. Ghorbanalizadeh. The multi-product pipeline scheduling system. *Computers & Mathematics with Applications*, 56(4):891 – 897, 2008.
- [37] A.V. Moura, C.C. de Souza, A.A. Cire, e T.M.T. Lopes. Heuristics and constraint programming hybridizations for a real pipeline planning and scheduling problem. *Proceedings of the 11th IEEE International Conference on Computational Science and Engineering - CSE'08*, páginas 455–462, Julho de 2008.
- [38] A.V. Moura, C.C. de Souza, A.A. Cire, e T.M.T. Lopes. Planning and scheduling the operation of a very large oil pipeline network. *Lecture Notes in Computer Science - Proceedings of the 14th International Conference on Principles and Practice of Constraint Programming*, volume 5202, páginas 36–51, Setembro de 2008.
- [39] G. Ottonsson. *Integration of Constraint Programming and Integer Programming for Combinatorial Optimization*. Tese de Doutorado, Uppsala University, Information Technology, Computing Sceince Department, 200. Uppsala, 2000.
- [40] C.H. Papadimitriou e K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, 1998.
- [41] Gilles Pesant e Michel Gendreau. A view of local search in constraint programming. *Principles and Practice of Constraint Programming*, páginas 353–366, 1996. Disponível em citeseer.ist.psu.edu/pesant96view.html.
- [42] PETROBRAS. Relatório anual de atividades, 2004.
- [43] PETROBRAS. RIMA - Relatório de Impacto Ambiental, Plano Diretor de Dutos de São Paulo PDD/SP, Setembro, 2007.
- [44] R. Rejowski e José M. Pinto. Efficient MILP formulations and valid cuts for multi-product pipeline scheduling. *Computers & Chemical Engineering*, 28(8):1511–1528, 2004.

- [45] R. Rejowski e José M. Pinto. A novel continuous time representation for the scheduling of pipeline systems with pumping yield rate constraints. *Computers & Chemical Engineering*, 32:1042–1066, 2008.
- [46] Susana Relvas, Ana Paula F. D. Barbosa-Póvoa, Henrique A. Matos, João Fialho, e António S. Pinheiro. Pipeline scheduling and distribution centre management - a real-world scenario at CLC. *Proceedings of the 16th European Symposium on Computer Aided Process Engineering and 9th International Symposium on Process Systems Engineering*, páginas 2135–2140, Garmisch-Partenkirchen, Germany, 2006.
- [47] Susana Relvas, Henrique A. Matos, Ana Paula F. D. Barbosa-Póvoa, e João Fialho. Reactive scheduling framework for a multiproduct pipeline with inventory management. *Industrial & Engineering Chemistry Research*, 46(17):5659–5672, 2007.
- [48] Susana Relvas, Henrique A. Matos, Ana Paula F. D. Barbosa-Póvoa, João Fialho, e António S. Pinheiro. Pipeline scheduling and inventory management of a multiproduct distribution oil system. *Industrial & Engineering Chemistry Research*, 45(23):7841–7855, 2006.
- [49] M.G.C. Resende e C.C. Ribeiro. Greedy randomized adaptive search procedures. F. Glover e G. Kochenberger, editors, *Handbook of Metaheuristics*, páginas 219–249. Kluwer Academic Publishers, 2002.
- [50] Álvaro García Sánchez e Miguel Ortega Mier. Programación de oleoductos: presentación del problema y revisión de enfoques. *Anales del IX Congreso de Ingeniería de Organización (CIO 2005)*, 2005.
- [51] R.A. Wilson. Transportation in america, eighteenth edition. Washington, D.C.: Eno Transportation Foundation, Inc., 2001.
- [52] L. A. Wolsey. *Integer Programming*. Wiley-Interscience, 1998.