UNIVERSIDADE FEDERAL FLUMINENSE

INSTITUTO DE CIÊNCIA E TECNOLOGIA

MESTRADO EM ENGENHARIA DE PRODUÇÃO E SISTEMAS COMPUTACIONAIS

ROBERT DA SILVA BRESSAN

SEQUENCIAMENTO DE TAREFAS PARA DESCOMISSIONAMENTO DE DUTOS FLEXÍVEIS SUBMARINOS

RIO DAS OSTRAS, RJ

2020

ROBERT DA SILVA BRESSAN

SEQUENCIAMENTO DE TAREFAS PARA DESCOMISSIONAMENTO DE DUTOS FLEXÍVEIS SUBMARINOS

Dissertação apresentada ao Instituto de Ciência e Tecnologia da Universidade Federal Fluminense, como requisito parcial da obtenção do título de Mestre em Engenharia de Produção e Sistemas Computacionais.
Campo de Confluência:
Pesquisa Operacional

Orientador:
Prof. Dr. Danilo Artigas da Rocha

Rio das Ostras, RJ

2020

Ficha catalográfica automática - SDC/BRO
Gerada com informações fornecidas pelo autor

B843s Bressan, Robert da Silva
Sequenciamento de tarefas para descomissionamento de dutos flexíveis submarinos / Robert da Silva Bressan ; Danilo Artigas da Rocha, orientador. Niterói, 2020.
77 f. : il.

Dissertação (mestrado profissional)-Universidade Federal Fluminense, Rio das Ostras, 2020.

DOI: http://dx.doi.org/10.22409/PPG-MESC.2020.m.06088616938

1. Descomissionamento. 2. Duto. 3. Otimização. 4. Teoria dos Grafos. 5. Produção intelectual. I. Rocha, Danilo Artigas da, orientador. II. Universidade Federal Fluminense. Instituto de Ciência e Tecnologia. III. Título.

CDD -

Bibliotecário responsável: Sandra Lopes Coelho - CRB7/3389

ROBERT DA SILVA BRESSAN

SEQUENCIAMENTO DE TAREFAS PARA DESCOMISSIONAMENTO DE DUTOS FLEXÍVEIS SUBMARINOS

Dissertação apresentada ao Instituto de Ciência e Tecnologia da Universidade Federal Fluminense, como requisito parcial da obtenção do título de Mestre em Engenharia de Produção e Sistemas Computacionais.
Campo de Confluência:
Pesquisa Operacional

Aprovada em 28 de agosto de 2020.

BANCA EXAMINADORA

Prof. Dr. Danilo Artigas da Rocha- UFF
Orientador

Prof. Dr. Dalessandro Soares Vianna - UFF

Prof. Dr. Edwin Benito Mitacc Meza - UFF

Prof. Dr. Fabiano de Souza Oliveira - UERJ

Rio das Ostras, RJ

2020

Dedicado às pessoas que enxergam o valor da destruição, para que haja espaço para algo novo.

Agradecimentos

Primeiro, gostaria de agradecer a quem mais dividiu as emoções da jornada comigo, minha esposa Analú. Ela sempre esteve lá, tomando conta de tudo para que eu pudesse avançar no trabalho. Um beijo, meu amor.

Não posso tirar o mérito de meu filho, Aaron. Sabe-se lá quantas vezes ele me perguntou "pai, o que tu tá fazendo?". Ficou demonstrado que Richard Feynman estava certo quando disse que você só aprendeu algo se consegue ensinar uma criança. Acho que meu filho sabe mais de teoria dos grafos que eu mesmo.

Agradeço também aos meus pais e meu irmão — Nereu, Cleusa e Albert, por sempre proverem incentivo e conforto.

Quero registrar aqui também o empenho do meu orientador, Danilo Artigas. Encarou o trabalho como se fosse dele. Eu não esperava tanta dedicação de um professor como ele o fez. Muito grato por tudo.

Aos demais professores e colegas da UFF, o agradecimento pelos conhecimentos compartilhados, muitas vezes utilizados fora do contexto da academia.

A todos que de alguma forma estiveram comigo neste processo, seja para discutir ciência, frustrações ou alegrias.

Um agradecimento também à CAPES, por auxiliar na qualidade do curso de pós graduação.

"Todo ato de criação é, antes de tudo, um ato de destruição."
(Pablo Picasso)

Resumo

O envelhecimento das instalações de petróleo tornou o projeto de descomissionamento relevante. Entre os elementos a descomissionar estão dutos flexíveis submarinos cuja retirada está sujeita a restrições de ordem, causados por cruzamentos. Propõe-se criar um algoritmo computacional para conceber uma ordem ótima de intervenção baseada em teoria dos grafos. Um caso real foi estudado e, a partir dele, criou-se o modelo matemático base, com grande correlação com o problema do conjunto mínimo de arcos de retorno. Este modelo utiliza o algoritmo A* para nortear a pesquisa da solução. Descobriu-se que o modelo criado tem complexidade computacional NP-difícil e exige, para o caso real, soluções aproximadas, em que foi vislumbrado o uso de beam search. Diversas heurísticas foram testadas para um subconjunto dos dados reais a fim de compreender quais são mais atrativas para limitar a largura de feixe e acelerar a pesquisa. Ficou demonstrado que as heurísticas baseadas em contagem de dutos cruzados e o algoritmo GreedyFAS produziram os melhores resultados na pesquisa. Por fim, percebe-se que há grande margem para melhoria através das estimativas do A*, e a expansão do modelo para inclusão de outros aspectos operacionais.

Palavras-chave: Duto flexível submarino; descomissionamento; sequenciamento de tarefas; teoria dos grafos.

Abstract

The aging of the oil installations made the decommissioning project relevant. Among the elements to be decommissioned are flexible subsea pipelines whose removal is subject to order restrictions, caused by crossings. It is proposed to create a computational algorithm to design an optimal order of intervention based on graph theory. A real case was studied, and, from it, the mathematical base model was created, with great correlation with the minimum feedback arc set problem. This model uses the A * algorithm to guide the search for the solution. It was discovered that the model created has a NP-hard computational complexity and requires, for the real case, approximate solutions, in which the use of beam search was envisioned. Several heuristics have been tested for a subset of the actual data in order to understand which are more attractive to limit the beam width and speed up the search. It was demonstrated that the heuristics based on counting of crossed ducts and the GreedyFAS algorithm produced the best results in the research. Finally, it becomes clear there are great opportunities for improvement through the A * estimates, and the expansion of the model to include other operational aspects.

Keywords: Subsea flexible pipes; decomissioning; task scheduling; graph theory.

Lista de ilustrações

Lista de algoritmos

Lista de tabelas

Sumário

1 Introdução

O petróleo ainda é uma das matrizes energéticas mais importantes do planeta, e não obstante, boa parte dos materiais utilizados na indústria ainda depende dele. Apesar de ocorrer um movimento para utilização de recursos renováveis, através de matrizes eólicas e solares, o petróleo ainda tem papel fundamental na sociedade atual.

A produção de petróleo exige equipamentos bastante complexos, e, como qualquer maquinário, está sujeita a uma vida útil. O petróleo, como recurso finito, uma vez extraído não há mais como realizar seu retorno ao estado original. Assim, a produção associada a uma jazida é limitada também pelo volume disponível na locação.

A consequência imediata destas características é que ao final da extração em uma jazida, ocorre o abandono das instalações de produção daquela região, eventualmente reaproveitando alguns equipamentos. Este trabalho aspira a maneira ótima de realizar este descomissionamento.

1.1 A produção de petróleo no mar

A produção marítima do petróleo inicia estendendo a produção terrestre, criando locais secos para colocação das árvores de natal, equipamento básico de operação e segurança do poço, em plataformas fixas.

Entretanto, o aumento da profundidade das instalações torna impraticável economicamente a construção de jaquetas mais elevadas, impossibilitando o uso delas para o desenvolvimento da produção. A primeira saída é tornar a jaqueta flutuante, mas ainda relativamente estática, presa por tendões. Nesse modelo de unidade de produção, a árvore de natal ainda permanece na unidade. O uso de árvores de natal secas também tem o inconveniente de limitar o alcance de produção de uma unidade.

Com o advento da tecnologia de árvore de natal molhada (ANM), começa a ser possível afastar a produção de sua unidade, uma vez que a cabeça do poço não é obrigatoriamente mais sob a unidade, conseguindo grande afastamento entre esta e a unidade. Também foi possível com isso permitir uma maior movimentação das unidades de produção, resultando no desenvolvimento de unidades de produção em navios.

Em contraponto, o escoamento de petróleo da cabeça de poço até a unidade de produção exige uma tubulação entre as mesmas, dando origem aos dutos submarinos de produção de petróleo. Além disso, pela separação entre a unidade e sua ANM, uma tubulação de controle, conhecida como umbilical, é necessária para movimentar e operar estes equipamentos.

Outros aspectos da transição da produção terrestre para a produção marítima, com a sua cronologia, podem ser vistas em (MASON, 2006).

1.1.1 Dutos submarinos de produção de petróleo

As primeiras tubulações para produção de petróleo foram as mais simplistas possíveis. A utilização direta de tubulação de superfície no fundo do mar exige poucas adaptações em sua operação em relação às suas contra-partes terrestres. Destaca-se as mudanças mais radicais no método de inspeção e no lançamento destes dutos. Esta primeira classe de dutos é conhecida como dutos rígidos.

O duto rígido, apesar de sua simplicidade, possui diversos percalços ao longo de sua vida útil: exige curvaturas pequenas nas suas rotas; sua construção é feita na locação, tornando sua instalação lenta; reaver o duto para reaproveitamento é, em geral, impraticável; sofre com vibrações, tanto da unidade quanto os causados pelas correntes marítimas, fenômeno conhecido como Vibração Induzida por Vórtice (VIV); e o planejamento da sua instalação exige cuidados na análise do solo, invalidando todo o projeto em uma alteração de posição no projeto do poço. Surge na década de 1970 a tecnologia de duto flexível submarino, visando atacar estas deficiências encontradas na instalação do duto rígido.

1.1.2 Dutos flexíveis submarinos

Os dutos flexíveis submarinos são tubulações compostas por diversas camadas poliméricas e metálicas, em que suas camadas poliméricas são responsáveis pela estanqueidade, enquanto as metálicas elevam a resistência estrutural.

Pelas suas características construtivas, este tipo de duto permite curvaturas com raios menores que 3 m, conseguindo suprir com isso duas das deficiências do projeto de duto rígido: como o duto pode ser enrolado em um raio relativamente pequeno, trechos prontos da ordem de quilômetros podem ser previamente construídos; outro ponto é que o duto consegue se acomodar ao solo, enfraquecendo as vibrações causadas pela corrente marinha e ajustes de rota, com pouca ou nenhuma análise, favorecendo o reaproveitamento desses dutos em diversos projetos.

Em contrapartida, as estruturas flexíveis costumam ser mais pesadas, fazendo com que as cargas transmitidas às unidades sejam maiores. Ela também não tem sua integridade medida, pois não há instrumento de inspeção capaz de realizar diagnóstico em multicamadas. Por fim, a sua montagem exige terminações bastante complexas, os conectores, que são responsáveis estruturalmente pela transmissão dos esforços.

Estas características do duto flexível ainda permitem facilmente realizar remanejamento deles entre poços. Quando um poço deixa de ser produtivo, pode-se simplesmente

desmontar a sua cabeça, tomar suas linhas, recolhendo-as parcialmente, e remontar na rota do poço substituto.

Mais informações acerca de dutos flexíveis podem ser encontradas no trabalho de Bai e Bai (2014, cap. 28).

1.2 Operações com dutos flexíveis

1.2.1 Embarcações

Para a instalação e retirada de dutos flexíveis submarinos, são utilizadas embarcações especiais conhecidas como *pipeline layer support vessel* (PLSV). O navio PLSV, representado na Figura 1, corriqueiramente possui as seguintes facilidades:

- Tensionador: uma esteira de lançamento que comprime o lado externo do duto, e o movimenta controladamente, seja no sentido de recolher ou lançar o duto.
- Mesa: ponto para apoio dos conectores, em que os acessórios são montados
- *Remotely Operated Vehicle* (ROV): robô submarino controlado a partir do navio, que transmite imagens subaquáticas e realiza pequenas intervenções com ferramentas.
- Guinchos: transportam as extremidades dos dutos e equipamentos entre o fundo marinho e navio.
- Carrosséis: estocam dutos, formando pilhas enroladas, para lançamento ou retirada.

Figura 1 – *Pipeline layer support vessel*

Fonte: Sapura (2020).

Outras facilidades são presentes a depender da aplicação, mas que para a aplicação desta tese não são relevantes. Existem navios mais simples que são especializados nas tarefas com ROV, conhecidos como RSV (*ROV support vessel*) e nas tarefas com guincho,

os AHTS (*anchor handling, tug and support*). Estes navios podem auxiliar em algumas tarefas do PLSV, a um custo significativamente mais baixo.

Os ROV são capazes de realizar diversas operações com dutos flexíveis, dentre elas:

- cortar um duto no seu corpo tubular. Este tipo de corte danifica irreversivelmente o duto, de tal sorte que pela ausência de um conector na sua extremidade, a operação de recolhimento é extremamente prejudicada.
- cortar na emenda de tramos. Este tipo de corte causa poucos danos no tubo, permitindo a instalação de ferramentas convencionais de dutos na sua extremidade. É executado serrando os estojos no espaço entre os flanges, ou aplicando ferramentas de destorqueamento.
- instalação de flutuadores. A instalação de flutuadores suspende parcialmente o duto, permitindo o acesso a pequenos trechos.

1.2.2 Descomissionamento de sistemas submarinos

"Assim como a única certeza da vida humana é a morte, a única certeza da vida de um campo é a obrigação do operador em descomissioná-lo" (ARS; RIOS, 2017).

O fim da vida do campo, associado a não-economicidade do campo ou a aspectos contratuais, como o distrato do afretamento de uma unidade, exige diversas ações para que o dano ambiental e o rastro da sua produção sejam minimizados. O projeto de abandono de um sistema de produção decorre de obrigações tanto legais quanto de imagem das companhias, visto que todo o recurso dispensado no descomissionamento não incorrerá em receita futura para as mesmas (GASPAR; BILDA, 2017).

Entre as ações de preparação, está o fechamento definitivo dos poços, limpeza das tubulações, tratando os rejeitos oleosos, a desconexão da tubulação entre a unidade de produção e o poço, até a liberação da unidade, com sua desancoragem e destinação (AHIAGA-DAGBUI et al., 2017).

O descomissionamento não necessariamente preconiza a retirada de todas as estruturas, podendo o abandono da instalação ocorrer com descarte *in situ*, retirada total, ou retirada parcial. O descarte *in situ* é típico de tubulação rígida, em que apenas a retirada das proteções é requisitada, e o duto é deixado para naturalmente se decompor.

O abandono diametralmente oposto é o recolhimento total das estruturas. Nesta categoria estão a maioria das estruturas de produção, como as linhas de ancoragem, os dutos flexíveis submarinos, e a unidade de produção, como visto em (ROUSE et al., 2018).

Para plataformas fixas, uma solução intermediária pode ser adotada: recolhe-se as estruturas nas porções mais rasas, e a parte restante é convertida em recife artificial

(SULAIMAN; ROS; AZIZAN, 2018). Não é objeto deste trabalho decidir qual é a melhor alternativa de abandono, mas o fato dos dutos flexíveis terem preferência pelo recolhimento total como modelo de abandono é primordial para este trabalho. O outro ponto vicinal é a existência de múltiplas alternativas de execução para o mesmo destino de estruturas.

1.3 Objetivos

1.3.1 Objetivos gerais

O objetivo desta pesquisa é propor um método computacional para elencar a sequência de tarefas a executar em um projeto de recolhimento de dutos flexíveis submarinos.

1.3.2 Objetivos específicos

Para atender o objetivo geral da pesquisa, foram estabelecidos objetivos específicos:

1. Desenvolver e implantar um método baseado em grafos para criar um espaço de pesquisa de soluções
2. Desenvolver uma notação compacta para modelagem específica do problema de cruzamentos.

1.3.3 Delimitação do problema

O problema abordado considerará apenas a etapa de recolhimento dos dutos flexíveis, dado um estado inicial conhecido. As etapas preliminares, de arriamento do duto da unidade de produção, e desconexão dos poços não serão abordados.

1.4 Justificativa e relevância da Pesquisa

Até a década de 1990, não existia uma preocupação com as consequências da atividade humana sobre o meio ambiente. E, já nessa época, a Bacia de Campos já possuía cerca de 20 unidades de produção (NUNES; FAVARO, 2019). Na implantação destes projetos, não houve preocupação com o destino do campo após sua produção, pois nem existia lei ambiental, que passou a existir com a Constituição de 1988 ((BRASIL, 1988), (BRASIL, 1967)). Assim, há diversos sistemas que nunca foram pensados para serem retirados.

Com a expansão dos sistemas de produção, a quantidade de unidades em produção em 2020 chega a 158. Muitos destes não tem um projeto claro de abandono, visto que

não há ainda um regulamento claro sobre o assunto (GASPAR; BILDA, 2017). O Brasil atualmente possui 65 unidades de produção com mais de 20 anos de operação (MACEDO, 2018), e nessas unidades, estão instalados mais de 10.000 km de dutos flexíveis submarinos na costa brasileira (OLIVEIRA et al., 2015). Não obstante, existe um passivo ambiental gigantesco a ser recolhido, fruto do tratamento inadequado dados aos sistemas de produção já abandonados. A demanda por projetos de descomissionamento de flexíveis pode ser vista em (PRADO, 2015).

Os pareceres do Instituto Brasileiro do Meio Ambiente e Recursos Renováveis (IBAMA) indicam a necessidade de remover todos os os dutos flexíveis submarinos (PETROBRAS, 2015), inclusive condicionando a colocação de novos sistemas, os chamados projetos de revitalização. Com a Resolução ANP 817 (BRASIL, 2020), a obrigatoriedade do recolhimento tornou-se mais clara.

Assim, é imperativo e tempestivo planejar o recolhimento de toda essa massa de dutos, de maneira eficiente e econômica. O auxílio computacional na tomada de decisões é justificado pelo grande volume de dutos a tratar. Atualmente o processo manual executado para o campo da Tainha (nome ficto) dispendeu cerca de quatro meses de análise.

Considerando que definir as ações é o primeiro passo de um projeto de descomissionamento, a criação do *software* de sequenciamento de tarefas pode subsidiar outras ações, como a escolha das ferramentas necessárias para execução, o levantamento das necessidades dos navios, e assim por diante.

Ademais, pela semelhança entre os projetos de investimento e os de descomissionamento em termos operacionais, a mesma ferramenta computacional pode ser aplicada com poucas adaptações para os projetos de instalação, ainda que o direcionador dos projetos de investimento seja o reservatório.

Logo, o cenário de crescente demanda por projetos de descomissionamento, e as possibilidades de expansão dessa ferramenta para cobrir outros aspectos do projeto justificam a pesquisa e a implantação da ferramenta de apoio descrita.

1.5 Estrutura do Trabalho

O primeiro capítulo dedica-se a contextualizar o leitor um pouco sobre instalações submarinas de petróleo e gás, identificando alguns elementos e detalhando os dutos flexíveis submarinos. O problema também foi enunciado em linhas gerais, e a seguir teve sua relevância justificada.

Segue-se no Capítulo 2 para a estratégia inicial de desenvolvimento da solução, primeiramente enunciando o problema, encontrando semelhanças com outros problemas em pesquisa operacional de forma a sustentar a estratégia de solução. A seguir, no Capítulo

3, a metodologia utilizada é apresentada.

O Capítulo 4 dedica-se à principal ferramenta da solução, teoria dos grafos. São apresentados os conceitos relevantes para o trabalho neste campo, bem como os algoritmos pertinentes a pesquisa.

Seguindo para o Capítulo 5, constrói-se a solução, criando o modelo baseado em teoria dos grafos para o problema proposto. Algumas definições formais, simplificações de solução são apresentadas, colocando os primeiros testes.

O trabalho sucede no Capítulo 6 com a aplicação do modelo, detalhando implantação, testes, propondo heurísticas e apresentando novos resultados no campo de Tainha. Por fim, uma conclusão no Capítulo 7 registra os desafios encontrados e proposições de pesquisa futura.

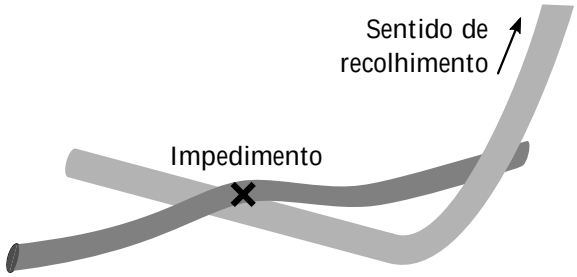
2 Problema proposto e estratégia de solução

2.1 Restrições ao descomissionamento de dutos flexíveis submarinos

O projeto de descomissionamento de flexíveis possui uma sistemática semelhante a qualquer outra movimentação de duto: as mesmas operações que permitem a instalação do duto são as mesmas que se colocam em seu recolhimento.

A restrição mais importante no projeto de recolhimento é o cruzamento dos dutos no leito marinho. Eventualmente, em função dos remanejamentos ocorridos ao longo da produção do campo, feixes de dutos podem acabar sobre outros. Um duto que esteja por cima impede, imediatamente, o recolhimento do duto que estiver por baixo, como demonstrado na Figura 2

Figura 2 – Cruzamento impedindo recolhimento



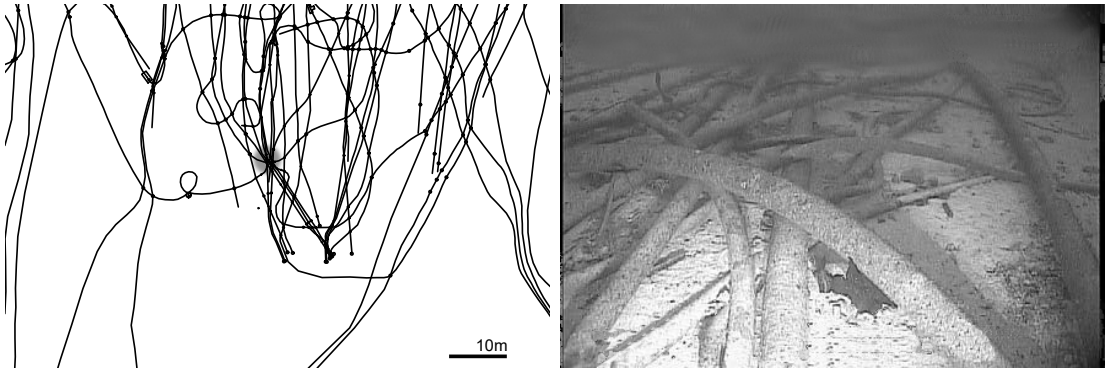
Fonte: Bressan e Artigas (2019).

Em campos mais antigos, este problema torna-se ainda maior, com um grande número de cruzamentos no leito submarino, consequência de uma quantidade maior de projetos de manutenção. A Figura 3 mostra um arranjo real, congestionado.

Esta restrição ao recolhimento pode ser removida de duas maneiras: movimentando o duto que estiver por cima, em uma operação conhecida como remanejamento, ou efetuando um corte no duto. As duas abordagens estão descritas na Figura 4.

A próxima restrição diz respeito à extremidade de um duto. Um duto pode ser movimentado a partir da sua extremidade, se ela for possível de ser içada. Uma extremidade pode ser içada facilmente se nela houver um conector. A tarefa pode ser restringida caso, em virtude de um corte, a extremidade não apresentar um conector. Neste caso, ou há uma penalidade forte, para montagem de ferramentas especiais, ou há a completa impossibilidade de içamento da estrutura por aquela extremidade.

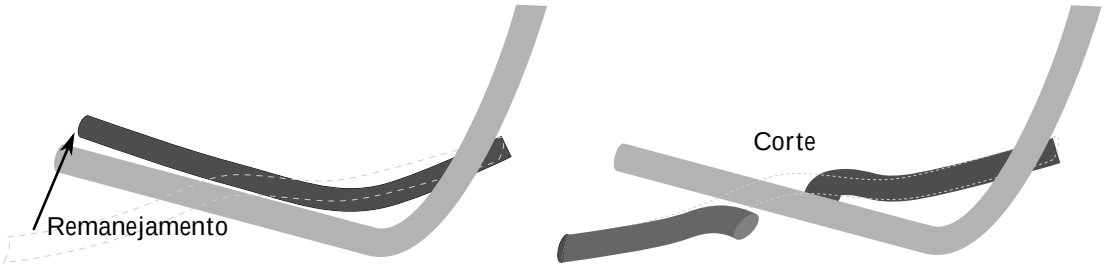
Figura 3 – Arranjo congestionado



(a) Vista em planta (b) Imagem de ROV destacado

Fonte: Bressan e Artigas (2019).

Figura 4 – Soluções para contornar cruzamento de dutos.



(a) Remanejamento liberando duto (b) Corte liberando duto

Fonte: Bressan e Artigas (2019).

Por fim, há restrições de ordem ambiental. Duas são mais proeminentes: a presença de corais no entorno do duto a ser recolhido, e a presença de agentes contaminantes no interior do duto. Os corais são organismos estruturantes para a vida marinha, de elevada sensibilidade. Em função dessas características, é desaconselhado realizar qualquer movimentação na tubulação que estiver cercada de corais. Neste trabalho, estas restrições de ordem ambiental são incorporadas apenas na meta, não afetando o método de solução.

Os dutos flexíveis, mesmo após lavagem, pela forma da carcaça intertravada (estrutura metálica mais interna), possuem uma quantidade significativa de óleo acumulado. Decorre daí outra restrição a cortes de dutos de produção de óleo e umbilicais com fluidos hidráulicos e funções químicas. Ainda podem ocorrer em dutos de produção materiais radioativos de ocorrência natural (NORM - *naturally ocurring of radioactive materials*). Dutos com NORM podem ser movimentados no fundo sem restrições, mas não podem ser embarcados, logo também não podem ser recolhidos nas condições normais de ocorrência (VALEUR, 2011). Assim, a movimentação desses dutos só ocorre até a chegada da extremidade no PLSV. Estas restrições não serão tratadas neste trabalho.

2.2 Definição do problema

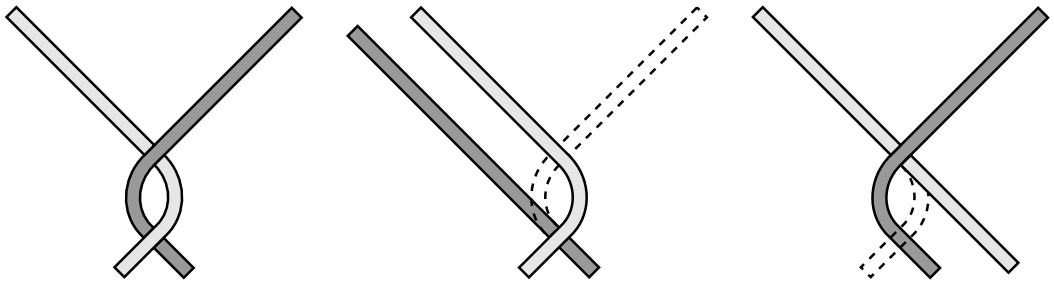
Um sistema submarino é o conjunto de todos os equipamentos e tubulações constantes de um bloco de produção, incluindo os dutos flexíveis. O arranjo submarino é a representação gráfica dessa disposição, descrevendo em um espaço georreferenciado a disposição e a interação entre os diversos componentes.

Ao longo da vida do campo, diversas modificações são realizadas nesse sistema, alterando o seu leiaute, e portanto, os meios de sua desativação. São exemplos de mudanças relevantes:

- Inclusão/Exclusão de poços;
- Mudança de função de poços;
- Manutenção dos componentes;
- Agregação de métodos de recuperação secundária.

O projeto de desativação possui diversas alternativas de execução, cada uma com um custo associado. Por exemplo, toma-se o laço da Figura 5a. Não é possível remover uma linha que esteja sob outra, então o laço deve ser desfeito. Este laço pode ser desfeito de duas maneiras distintas, seguindo os caminhos descritos na Figura 5b ou na Figura 5c.

Figura 5 – Um laço e duas formas de remover o laço.



(a) Dutos em laço (b) Alternativa longa (c) Alternativa curta

Fonte: Autor (2020)

A existência de uma variedade de meios de executar leva à busca por uma alternativa mais atrativa para que o estado desejado seja atingido.

O estado final desejado, via de regra, é a remoção de todos os elementos. Entretanto, nada impede que uma retirada parcial dos elementos seja a meta para o projeto de descomissionamento. O importante é que tenha-se claro quais elementos são obrigatórios para remoção, quais são opcionais, e quais tem sua retirada proibida.

Com isso, pode-se enunciar o problema deste trabalho:

> *Seja um arranjo conhecido de dutos flexíveis submarinos, em que cada elemento está atribuída uma destinação — retirar, deixar ou indiferença. Elencar a sequência de atividades necessárias mais atrativa para que a destinação de cada elemento seja respeitada.*

2.3 Problemas correlatos

O problema de recolhimento de sistemas submarinos ainda não é explorado na literatura, e portanto, não há referências diretas no assunto. Uma análise da base OnePetro, principal base de dados científicos da indústria do petróleo, sobre o descomissionamento retorna cerca de 300 publicações com as palavras-chaves "*decom*" ou "*decommissioning*". Entretanto, praticamente a totalidade das publicações discorre sobre impacto ambiental, regulamentos e demanda, alguns deles citados no Capítulo 1. Não há publicações na área de pesquisa operacional, colocando a oportunidade de desenvolvimento de ferramentas e estudos.

Dada a ausência de literatura sobre o assunto, o problema de otimização do recolhimento deverá ser construído com apoio de problemas correlatos. A principal característica do problema é uma relação de precedência das operações, que por limitação operacional, deve seguir a ordem dos cruzamentos.

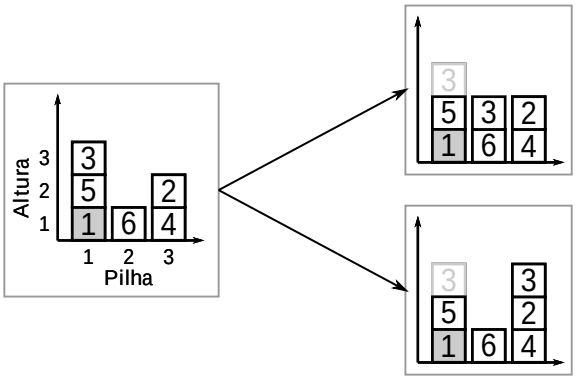
Essa restrição pode ser entendida da mesma forma que o processo de triagem de contêineres em um porto, como visto em (TANAKA; TAKII, 2016). Na triagem de contêineres, estes estão empilhados em diversas pilhas, e devem ser reorganizados para uma ordem de prioridade para o seu carregamento. Em portos, apenas um contêiner pode ser movimentado de cada vez, e para obter um contêiner, ele precisa estar necessariamente estar no topo de sua pilha.

Caso seja necessário um contêiner que esteja sob uma pilha, deve-se remover todos aqueles que estiverem sobre o objeto de interesse. É possível visualizar esse problema na Figura 6. Deseja-se obter o contêiner de número 1, na pilha 1. Para acessá-lo, requere-se movimentar os contêineres 3 e 5. Para o movimento do contêiner 3, pode-se escolher mover para a pilha 2 ou para a pilha 3.

A principal diferença entre o problema da triagem e o problema tratado é que um duto pode impedir a movimentação de diversos dutos, enquanto o contêiner necessariamente está empilhado sobre apenas um outro. Além disso, um par de dutos pode ter um cruzamento em que um está sobre outro, e em outro cruzamento, o contrário ocorre. Em contrapartida, um contêiner nunca está sobre e sob outro contêiner simultaneamente.

Nas publicações de Bacci, Mattia e Ventura (2019) e Wu e Ting (2010), o problema da triagem de contêineres é abordado construindo um grafo. Ele coloca como vértice do

Figura 6 – Movimentação de contêineres



Fonte: Adaptado de Tanaka e Takii (2016)

grafo uma fotografia da disposição dos contêineres, representados por diversas pilhas. A aresta do grafo construído é uma ação, que pode ser "remover" ou "mover" a depender das possibilidades, unindo dois vértices. Um custo de execução é atribuído a cada aresta, de acordo com a mudança executada. Para a pesquisa da sequência de triagem, o método *beam search* foi aplicado.

Uma técnica semelhante foi utilizada por Bennell, Cabo e Martínez-Sykora (2018) no problema de empacotamento de materiais irregulares. Na solução indicada, os estados são os arranjos de folhas, e as arestas representam a adição de um pacote em uma determinada posição. Com isso, conseguiu-se discretizar o problema do empacotamento. A relação de precedência do problema do empacotamento é a ordem dos pacotes.

Khemani et al. (2019) aplica o mesmo conceito para encontrar solução ótima do cubo de Rubik e determinar quais posições do cubo exigem as soluções mais longas. A demonstração almejada nessa publicação era demonstrar o diâmetro do grafo das posições do cubo, em que as arestas representam movimentos do cubo (girar uma face). Para isso, para ganho de tempo computacional, uma heurística de solução parcial foi gerada.

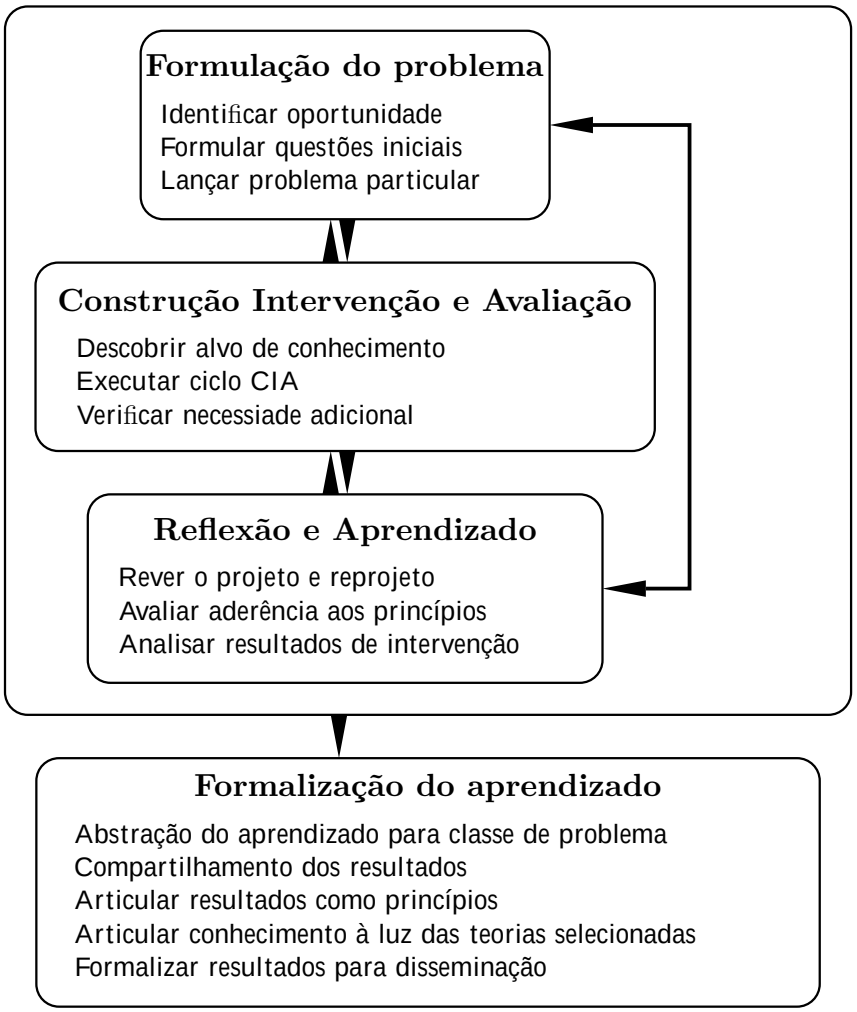
Ainda nos problemas correlatos, merece atenção o problema de máximo palíndromo em uma sequência de caracteres, que encontra aplicação no estudo de sequenciamento genético. Este problema foi abordado por Djukanovic, Raidl e Blum (2019), utilizando grafos para estruturar as sequências simétricas.

3 Metodologia

A fim de atingir o objetivo da pesquisa deve-se levar em conta as condições envolvidas no momento da pesquisa. O estado de indefinição das premissas inerente ao problema, visto que há uma forte discussão na legislação, já abordada na introdução, fragiliza métodos em que é criada uma solução geral e depois é feita uma particularização, como o clássico *Design Science Research* apresentado por Peffers et al. (2007).

Uma abordagem mais direta para uma instância do problema torna-se mais atrativa nesse cenário. (SEIN et al., 2011) apresenta uma metodologia batizada de *Action Design Research*, cujas etapas são representadas na Figura 7.

Figura 7 – Action Design Research



Fonte: Adaptado de Sein et al. (2011)

A metodologia *Action Design Research* ocupa-se primariamente de resolver um problema de campo, para em seguida usar o aprendizado para generalizar o problema.

Por esta característica de focar inicialmente na instância restrita, a solução pode ser discutida com as partes interessadas, de forma que apenas a resposta que satisfaça a todos é generalizada. Esta qualidade do método assemelha-se bastante ao desenvolvimento de *software* por metodologia ágil, descrito por (BECK et al., 2001).

Em resumo, o *Action Design Research* preconiza:

1. Criação interativa com a parte interessada;
2. Solução imediata de uma instância do problema;
3. Criação de um artefato utilizando uma teoria-base;
4. Generalização da solução obtida para a instância.

O caso particular que será estudado é o descomissionamento de um campo da bacia de Campos, que será denominado pelo nome fictício "Tainha". O posicionamento dos dutos e os custos foram cedidos pela operadora deste campo. Os dados de dutos e cruzamentos existentes no campo foram escalonados por um fator constante, de forma que a solução seja representativa do campo estudado e não seja quebrado o sigilo empresarial.

Entende-se que as definições e propriedades encontradas para o campo de Tainha serão válidas para qualquer outro campo com dutos flexíveis submarinos, pelo fato deste campo possuir a maior complexidade encontrada entre os campos da operadora, e vislumbrar todas as hipóteses de restrição até agora percebidas.

Esta é uma pesquisa essencialmente aplicada, uma vez que estuda-se a configuração particular de um campo de produção. Esta é uma pesquisa explicativa visto que o método de solução foi encontrada após experimentação. Por fim pode-se classificar esta pesquisa em experimental já que verifica-se os efeitos de cada tentativa sobre o conjunto de dados do campo escolhido.

4 Teoria dos grafos

4.1 Definições em teoria dos grafos

O grafo é uma estrutura matemática bastante flexível que representa conexões entre elementos. Uma definição formal para um grafo é encontrada em Feofiloff, Kohayakawa e Wakabayashi (2011):

Seja um conjunto qualquer V, denota-se $V^{(2)}$ o conjunto de todos os pares não ordenados de elementos de V. Se V tem n elementos, então $V^{(2)}$ possui $\binom{n}{2} := \frac{n(n-1)}{2}$ elementos. Os elementos de $V^{(2)}$ serão identificados com os subconjuntos de V que têm cardinalidade 2. Assim, cada elemento de $V^{(2)}$ terá a forma $\{v, w\}$, sendo v e w dois elementos distintos de V.

Um grafo é um par (V, E) em que V é um conjunto arbitrário e E é um subconjunto de $V^{(2)}$. Os elementos de V são chamados vértices e os de A são chamados arestas. Neste texto, restringir-se-á a grafos em que o conjunto de vértices é finito, e via de regra, não-vazio.

Quando $V^{(2)}$ é o conjunto dos pares ordenados de elementos de V, ou seja, (u, v) é distinto de (v, u), a estrutura é conhecida como *digrafo*, ou *grafo direcionado*. Ainda, caso seja possível mais de um arco entre dois vértices, cada aresta agora da forma $((u, v), r)$ denotando que o arco (u, v) tem multiplicidade r, ou seja, $E \quad V^2 \times n$, com $n \quad \mathbb{N}$, a estrutura matemática é conhecida como multigrafo.

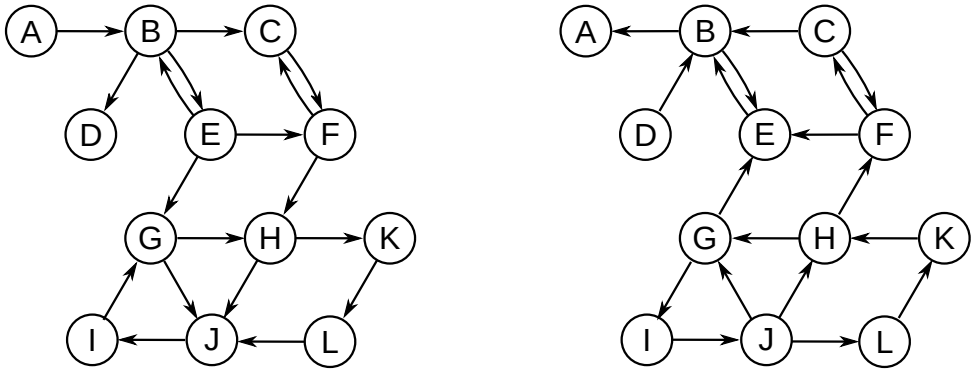
É costumeiro representar geometricamente um grafo como um conjunto de pontos, representando os vértices, com linhas ligando estes pontos para as suas arestas (MELO, 2014). Quando o grafo é direcionado, uma seta é utilizada para indicar o sentido da relação.

Neste trabalho, apenas os grafos e multigrafos *direcionados* serão utilizados, de tal forma que as definições apresentadas aqui considerarão apenas este contexto. As definições apresentadas aqui são retiradas de Bondy e Murty (2010).

Dois vértices u e v são adjacentes se a aresta $(u, v) \quad E$. Esta aresta é *divergente* em u e *convergente* em v. A quantidade de arestas divergentes em um vértice é define o grau de saída ($d^+(v)$) deste vértice, enquanto a contagem de arestas convergentes define o grau de entrada ($d^-(v)$) do mesmo.

Um vértice que possua apenas arestas divergentes, ou seja, grau de entrada nulo, é dito *fonte*, enquanto um vértice que possua apenas arestas convergentes é conhecido como *sumidouro*. Um vértice sem qualquer aresta é dito *isolado*. Estes vértices possuem grande relevância para a solução do problema, pelas suas propriedades.

Figura 8 – Grafo Transposto



(a) Um grafo direcionado (b) O grafo transposto

Fonte: Dasgupta (2008).

Para as definições de subgrafo, denotar-se-á por $V(G)$ o conjunto de vértices do grafo G e $E(G)$ o conjunto de arestas de G. Um grafo H é um *subgrafo próprio* de G ($H \quad G$) se $V(H) \quad V(G)$, $E(H) \quad E(G)$. Um subgrafo é chamado *induzido* se o conjunto $E(H)$ é o maior subconjunto de $E(G)$, ou seja, aproveitando todas as arestas incidentes em $V(H)$. Também vale citar a definição de grafo *transposto* (G^R), que é o grafo direcionado com as arestas com o sentido invertido, como na Figura 8.

Um *passeio* é uma sequência finita $W = v_0 e_1 v_1 e_2 v_2 e_2 \ldots e_k v_k$ em que e_i é uma aresta divergente de v_{i-1} e convergente em v_i. Um *caminho* é um passeio em que os vértices são todos distintos. Por fim, se apenas o primeiro e o último vértice são idênticos, e os demais são distintos, este passeio é denominado *ciclo*.

Um grafo que não possua ciclos é conhecido como *DAG - directed acyclic graph* - digrafo acíclico. Os digrafos acíclicos possuem uma ordenação parcial de seus elementos, em que um vértice divergente é colocado antes do vértice convergente em uma aresta. Esta ordenação é conhecida como *ordem topológica* e ela existe se e somente se o grafo é acíclico.

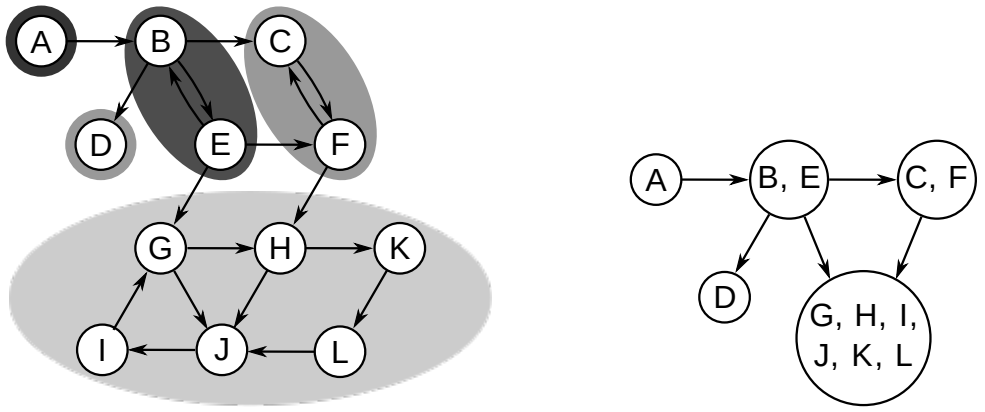
Este tipo de grafo é muito relevante para sequenciamento de tarefas. Modelando atividades como vértices e relações de precedência como arestas direcionadas, a ordem topológica é uma ordem válida de execução das tarefas que não quebra as relações de precedência.

Um vértice u está conectado a v se existe um caminho de u para v. Se além disso existir um caminho de v para u, formando um ciclo, estes dois vértices são *fortemente conectados*. Um *conjunto fortemente conexo* é um conjunto de vértices em que todos eles estão fortemente conectados. Este conjunto é torna-se um *componente fortemente conexo* se contiver o máximo de vértices possível.

Um grafo direcionado qualquer pode ser decomposto nos seus componentes fortemente conexos, sendo que ao aglutinar estes componentes — colocar o conjunto todo como

um único vértice — as arestas restantes formam um digrafo acíclico. Um exemplo de como ocorre essa aglutinação é dada na Figura 9. As partições induzidas pelos componentes fortemente conexos serão utilizadas para simplificar a análise do problema.

Figura 9 – Separação em componentes conexos



(a) Grafo inicial (b) Grafo induzido pelos componentes

Um grafo direcionado com os grupos fortemente conexos destacados. Ao aglutinar os grupos para um único vértice, o grafo resultante é acíclico. Fonte: Dasgupta (2008).

Ainda apresentando as definições, tem-se o *grafo ponderado*. Um grafo ponderado tem um valor (peso) associado a cada aresta, que pode ser interpretado como um custo, uma distância ou qualquer outra grandeza associada ao problemaa. Em um grafo ponderado, um *camínho mínimo* entre dois vértices é o caminho em que a soma dos pesos das arestas que o compõe é a menor entre todos os caminhos possíveis entre vértices deste grafo. A busca por este caminho mínimo é a chave da solução do problema proposto.

4.2 Algoritmos principais em teoria dos grafos

Serão apresentados aqui alguns dos principais algoritmos utilizados em teoria dos grafos e que possuam relevância para a solução do problema abordado neste trabalho. São eles: Busca em profundidade, Ordenação Topológica (Tarjan e Kahn), Componentes Fortemente Conexos (Kosaraju), Camínho mínimo (Dijkstra e A*), Conjunto mínimo de arcos de retorno (Eades et al).

4.2.1 Busca em profundidade

A busca em profundidade é um algoritmo de pesquisa em grafos que lembra um método de encontrar a saída de um labirinto usando um giz e um rolo de fio. Um meio de encontrar a solução é, ao encontrar uma interseção (vértice), escolher um caminho (aresta) e marcá-lo com um giz. Caso se encontre um local sem saída, ou um lugar previamente

marcado, usa-se o rolo de fio para retornar ao último ponto de decisão, e escolhe-se outra rota (DASGUPTA, 2008).

A marcação em giz pode ser interpretada como uma variável booleana — riscado ou não — e o rolo de fio pode ser entendido computacionalmente como uma pilha. O algoritmo finaliza, no pior caso, ao visitar todos os corredores, que são transcritos como arestas do grafo. Uma maneira mais formal é apresentada no Algoritmo 1.

Algoritmo 1 Busca em Profundidade

Entrada Grafo $G(V, E)$; $v \quad V$

```
1:  procedimento Explorar(G, v)
2:      visitado[v] ← Verdadeiro
3:      Previsita(v)
4:      para todo (v, u)  E faça
5:          se ¬visitado(u) então
6:              Explorar(G, u)
7:      Posvisita(v)
8:  procedimento DFS(G)
9:      para todo v  V faça
10:         visitado(v) ← Falso
11:     para todo v  V faça
12:         se ¬visitado(v) então
13:             Explorar(G, v)
```

Ao executar uma busca em profundidade, ainda no exemplo do labirinto, cada aresta pode ser classificada em árvore, retorno ou cruzada, de acordo com a ordem visitada:

Árvore Foi encontrada uma nova interseção no labirinto, nunca visitada.

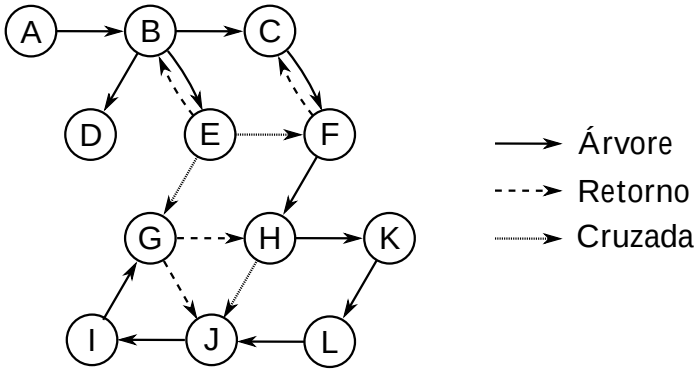
Retorno Foi encontrada a linha, indicando que há um caminho em círculo.

Cruzada Foi encontrada um local já visitado pela marcação, mas sem encontrar a linha, indicando dois caminhos que levam ao mesmo lugar.

Por exemplo, o grafo da Figura 8a, utilizando a ordem lexicográfica como referência, classificará as arestas conforme a Figura 10.

As funções Previsita e Posvisita do Algoritmo 1 são opcionais, e podem assumir diversas funções de acordo com a necessidade do problema. Um uso bastante comum para essas duas funções é colocar um contador, que associa ao vértice em questão à ordem de execução. Com isso, são criados dois vetores: $pre[v]$ e $pos[v]$, armazenando a ordem em que o vértice v entrou e saiu da pilha da recursão. Considerando o grafo da Figura 8a, a execução do algoritmo atribuirá os valores da Tabela 1 para $pre[v]$ e $pos[v]$ ao executar a busca em profundidade.

Figura 10 – Classificação de arestas em uma busca em profundidade



Fonte: Autor (2020).

Tabela 1 – Ordem de execução da busca em profundidade

v	A	B	C	D	E	F	G	H	I	J	K	L
$pre[v]$	1	2	3	11	12	4	10	5	9	8	6	7
$pos[v]$	12	11	8	9	10	7	1	6	2	3	5	4

Fonte: Autor (2020).

A classificação das arestas pode ser feita utilizando os valores de $pre[v]$ e $pos[v]$. Ao visitar uma aresta (u, v), os três tipos de aresta satisfazem as seguintes desigualdades:

Árvore $pre[u] < pre[v] < pos[v] < pos[u]$

Retorno $pre[v] < pre[u] < pos[u] < pos[v]$

Cruzada $pre[v] < pos[v] < pre[u] < pos[u]$

Enuncia-se aqui uma propriedade importante acerca das arestas de retorno que será explorada ao longo deste trabalho:

Propriedade 1. *Uma busca em profundidade retorna arestas de retorno se e somente se existe um ciclo no grafo.*

O algoritmo de busca em profundidade pode então ser utilizado para detectar a existência de um ciclo, verificando a desigualdade acima. Além disso, caso não haja arestas de retorno, ou seja, o grafo é acíclico, colocando os vértices em ordem decrescente de $pos[v]$ obtem-se uma ordem topológica válida.

Esta constatação foi documentada inicialmente por Tarjan (1972). Neste trabalho, no entanto, para ordem topológica, foi preferido o algoritmo de Kahn (1962), descrito na Subseção 4.2.2, por ser a base de outro algoritmo utilizado na solução.

4.2.2 Algoritmo de Kahn

O Algoritmo de Kahn (1962) para ordenação topológica baseia-se em identificar vértices fonte sucessivamente, e que, em um digrafo acíclico, sempre existirá ao menos um. Ao encontrar este vértice, o subgrafo induzido pela remoção dele é investigado, até que este se torne vazio. A retirada do vértice é executada apenas atualizando os graus de entrada dos vizinhos que divergem dele, resultando no Algoritmo 2.

Algoritmo 2 Ordem Topológica - Kahn

Entrada Grafo $G(V, E)$

```
1:  procedimento Kahn(G)
2:      Inicie uma fila Q ←
3:      Inicie um conjunto S ←
4:      para todo v  V faça
5:          se d⁻[v] = 0 então                    d⁻[v] Grau de entrada
6:              S ← S  {v}
7:      enquanto S =  faça
8:          Escolha u  S
9:          Q ← Q  {u}
10:         S ← S − {u}
11:         para todo v  N(u) faça                N(u) é a vizinhança de u
12:             d⁻(v) ← d⁻(v) − 1
13:             se d⁻(v) = 0 então
14:                 S ← S  {v}
15:     se |V| = |Q| então
16:         retorne Q
17:     senão
18:         Falha, há ciclo no grafo.
```

O algoritmo de Kahn também pode ser executado no sentido inverso, calculando os graus de saída, pesquisando vértices sumidouros e atualizando os graus dos vizinhos conforme as arestas convergentes dos vértices sumidouros. A ordem obtida utilizando os sumidouros estará invertida. Esta abordagem invertida é utilizada no algoritmo GreedyFAS, discutido na Subseção 4.2.7.

4.2.3 Algoritmo de Kosaraju-Sharir

O algoritmo de Kosaraju-Sharir (SHARIR, 1981) resolve o problema de encontrar os componentes fortemente conexos baseado em uma propriedade desses componentes:

Propriedade 2. *Os componentes fortemente conexos de G e do grafo transposto G^R são idênticos.*

Esta abordagem, apesar de que não seja computacionalmente a mais eficiente, tem uma das implantações mais simples. Outra propriedade relevante para este algoritmo é:

Propriedade 3. *Os componentes fontes do grafo reguzido de G são componentes sumidouros daquele de G^R e vice-versa.*

Ao executar a função `Explorar` da busca em profundidade em um vértice de um componente sumidouro, ele alcançará apenas os vértices do seu próprio conjunto fortemente conexo. O algoritmo de busca em profundidade calcula facilmente os componentes fontes, utilizando $pos[v]$, bastando ordenar do maior para o menor (DASGUPTA, 2008).

Ao transpor o grafo, estes componentes fontes em G se tornam sumidouros em G^R. Então, ao executar uma nova busca em profundidade em G^R respeitando a ordem decrescente de $pos[v]$ obtida ao executar a busca em G, conseguir-se-á isolar os componentes conexos.

Como um exemplo, obtem-se a seguinte ordem de $pos[v]$ para o grafo da Figura 8a: $G, I, J, L, K, H, F, C, D, E, B, A$. Ao executar a segunda busca em profundidade no grafo transposto (Figura 8b) invertendo esta ordem, a execução encontrará os vértices $\{A\}, \{B, E\}, \{D\}, \{C, F\}, \{H, G, I, J, L, K\}$, em que cada conjunto é uma execução da função `Explorar`. A visualização dos componentes está na Figura 9.

4.2.4 Algoritmo de Dijkstra

O algoritmo de Dijkstra (1959) encontra o caminho mínimo entre um nó e todos os demais vértices do grafo, dado que os pesos atribuídos às arestas sejam não-negativas.

Dados um grafo ponderado G com pesos não-negativos e um vértice s. A ideia do algoritmo de Dijkstra consiste em separar os vértices do grafo em dois conjuntos: vértices conhecidos e desconhecidos. Inicialmente, apenas o vértice s é conhecido. A cada iteração o algoritmo torna conhecido o vértice v mais próximo de s. Em seguida, as distâncias entre s e os vizinhos de v são atualizadas utilizando algum caminho formado por vértices conhecidos.

A fim de se conhecer o vértice mais próximo de v, mantém-se uma fila de prioridades Q e a melhor estimativa da distância $d[v]$ de v até s. O valor de $d[s]$ deve ser zero, pois s é o vértice inicial, enquanto os demais recebem um valor indefinido. O vértice m de maior prioridade — menor valor acumulado em d— é retirado da fila e passado para os vértices conhecidos. Então é feita a pergunta: o caminho com peso $w[m, u]$ passando pelos vizinhos de m leva a algum vizinho u dele a ser mais favorável? Se sim, ou seja, $d[m] + w[m, u] < d[u]$, atualiza-se o valor de $d[w]$, e consequentemente, reorganiza-se a fila Q.

Um resultado bastante conhecido é: no momento que o vértice v é removido de Q, $d[v]$ contém exatamente o valor do caminho mínimo entre s e v. Neste sentido, se possuímos um único vértice t como meta, pode-se parar a execução assim que t sair da fila

de prioridade.

Na versão apresentada no Algoritmo 3, além do parâmetro de meta, para que reconstrua-se o caminho executado ao atingi-la, armazenamos também o predecessor $p[w]$ dele no caminho ao atualizar $d[w]$. Verificando sucessivamente o predecessor até que $p[w] = s$, obtém-se o caminho trilhado.

Algoritmo 3 Caminho mínimo – Dijkstra

Entrada Grafo Ponderado $G(V, E)$
$s \quad V$, Início
$t \quad V$, Meta
$w(u, v) \quad \mathbb{R}_+$, Pesos de aresta

```
1:  procedimento Reconstruir(u)
2:      se p[u] = s então
3:          Reconstruir(p[u])
4:      Escreva p[u]
5:  função Dijkstra(G, s, t)
6:      Q ←                                   Q é um conjunto de vértices
7:      para todo v  V faça
8:          d[v] ← ∞                          Vetor de distância
9:          p[v] ← nulo                       Vetor de precedentes
10:         Adicione v a Q
11:     d[s] ← 0
12:     enquanto Q =  faça
13:         u ← o vértice de menor d[u] tal que u  Q
14:         se u = t então
15:             Reconstruir(u)
16:             retorne d[u]
17:         Q ← Q − {u}
18:         para todo v  N(u) tal que v  Q faça
19:             se d[u] + w[u, v] < d[v] então
20:                 d[v] ← d[u] + w(u, v)
21:                 p[v] ← u
22:     se d[t] = ∞ então
23:         erro Não há um caminho de s até t
```

4.2.5 Algoritmo A*

O algoritmo de Dijkstra sofre com uma deficiência quando o objetivo é encontrar o caminho entre dois nós, porque visita diversos nós que não são promissores para a solução. Uma solução para priorizar rotas promissoras é colocar uma estimativa do quanto falta para atingir o destino, e priorizar a visita dos nós levando em conta tanto o valor já percorrido quanto o estimado para o restante do trajeto, ou seja, pelo valor esperado do percurso todo.

Foi esta a ideia de Hart, Nilsson e Raphael (1968) para expandir do algoritmo de Dijkstra, conhecida como A* (*A-star*). A eficiência do algoritmo é maior na medida em que esta estimativa para a solução é mais próxima da solução efetiva do problema.

O algoritmo A* tem exatamente a mesma estrutura do algoritmo de Dijkstra, com duas diferenças. A primeira é que cada nó deve ganhar uma estimativa de qual é a distância entre ele e o destino, calculada por uma função aqui chamada `Estimador`, e armazenada em $h[v]$, conforme a Linha 7 do Algoritmo 4. A outra diferença está na Linha 11, que deve ser reescrita para colocar os vértices na ordem do valor esperado, e não apenas do valor acumulado. O Algoritmo 4 mostra estas duas diferenças.

Algoritmo 4 Caminho mínimo – A*

Entrada Grafo Ponderado $G(V, E)$
$s \quad V$, Início
$t \quad V$, Meta
$w(u, v) \quad \mathbb{R}_+$, Pesos de aresta
`Estimador`: $V \rightarrow \mathbb{R}_+$, Estimador da distância para solução

```
1:  procedimento Reconstruir(u)
    ...                                              Sem alterações aqui

2:  função AStar(G, s, t, Estimador)
3:      Crie um conjunto de vértices Q
4:      para todo v ∈ V faça
5:          d[v] ← ∞
6:          p[v] ← nulo
7:          h[v] ← Estimador(v)                      Atribuição da estimativa
8:          Adicione v a Q
9:      d[s] ← 0
10:     enquanto Q ≠ ∅ faça
11:         u ← o vértice de menor d[u] + h[u]       Diferença para Dijkstra aqui
12:         se u = t então
13:             Reconstruir(u)
14:             retorne d[u]
15:         Q ← Q − {u}
16:         para todo v ∈ N(u) tal que v ∈ Q faça
17:             se d[u] + w(u, v) < d[v] então
18:                 d[v] ← d[u] + w(u, v)
19:                 p[v] ← u
20:     se d[t] = ∞ então
21:         erro Não há um caminho de s até t
```

A execução do algoritmo A* exige que seja dada uma estimativa para a solução do problema. Se a heurística aplicada para a estimativa da finalização for nula, o algoritmo torna-se o de Dijkstra. Esta estimativa, para que seja garantida a otimalidade sem revisitar

um nó, precisa atender dois critérios (HART; NILSSON; RAPHAEL, 1972):

- Admissibilidade: a estimativa deve ser um limite inferior da solução. Isto é, a estimativa dada deve ser otimista em relação à solução.
- Consistência: a estimativa de um nó deve ser sempre inferior que a estimativa de um vizinho somado ao custo de atingir o vizinho. Ou seja, ela respeita a desigualdade triangular.

É conveniente colocar os vértices sob demanda na fila de prioridade Q e armazenar apenas os valores de distância $d[v]$ já estimados quando em uma execução num grafo com um número grande de vértices, ou quando não se conhece previamente todos os vértices. A fila de prioridade Q conterá apenas os vértices já atingidos pela busca, mas não fechados enquanto o vetor de distância torna-se um dicionário. Esta abordagem de alocação dinâmica dos vértices na fila será explorada a seguir.

4.2.6 Busca em feixe (*beam search*)

Eventualmente, um grafo pode se tornar tão grande que não é viável realizar a pesquisa do algoritmo A* por completo. Uma forma de obter uma solução mais rapidamente, sacrificando o critério de otimalidade, é reduzir a quantidade de vizinhos avaliados em cada vértice, descartando arestas que, no ponto de vista do problema, não parecem conduzir à solução ótima.

Esta abordagem de retirar os vizinhos menos promissores é conhecida como *beam search*. Os critérios para escolha de um vizinho e descarte de outros podem ser gulosos ou probabilísticos, como o utilizado por Blum e Blesa (2009). Aqui será descrita a variante gulosa.

Para cada nó visitado, define-se um número β máximo de vizinhos, chamado *largura de feixe*. Ao visitar um vértice, todos os vizinhos dele ganham, além do custo do caminho, uma função de priorização de aresta, chamada de `Favorabilidade` no Algoritmo 5.

Os β melhores candidatos são colocados na fila de prioridade, enquanto os demais são descartados. Esta alteração é realizada na Linha 16 do Algoritmo 4, resultando nas linhas 15 a 17 do Algoritmo 5.

Convém lembrar que no *beam search*, como não são explorados todos os nós, o algoritmo pode não encontrar um caminho entre s e t, mesmo que ele exista, pois o caminho disponível pode estar fora dos β vizinhos mais favoráveis. Assim, a mensagem de erro da Linha 28 deve ser reescrita levando isto em consideração.

Algoritmo 5 Caminho mínimo – *beam search* em A*

Entrada Grafo Ponderado $G(V, E)$
$s \quad V$, Início
$t \quad V$, Meta
$w(u, v) \quad \mathbb{R}_+$, Pesos de aresta
Estimador: $V \rightarrow \mathbb{R}_+$, Função estimadora da distância para solução
Favorabilidade: $E \rightarrow \mathbb{R}$, Função priorizadora de arestas
$\beta \quad \mathbb{N}$, Largura de feixe

```
1: procedimento Reconstruir(u)
   ...

2: função Beam Search(G, s, t, Estimador, β, Favorabilidade)
3:     Crie uma fila de prioridade Q
4:     Crie um conjunto S
5:     Q ← {s}
6:     S ←
7:     d[s] ← 0; h[s] ← Estimador(s)
8:     enquanto Q =  faça
9:         u ← o vértice de menor d[u] + h[u]
10:        se u = t então
11:            Reconstruir(u)
12:            retorne d[u]
13:        S ← S  {u}
14:        Q ← Q − {u}
15:        para todo v  N(u) em que v / S faça
16:            f[v] ← Favorabilidade(u, v)
                                  f[v] armazenará as favorabilidades temporariamente
17:        F ← β melhores v, de acordo com f[v].        Filtro do beam search
18:        para todo v  F faça
19:            se v / Q então
20:                Q ← Q  {v}
21:                h[v] ← Estimador(v)
22:                d[v] ← d[u] + w(u, v)
23:                p[v] ← u
24:            senão se d[u] + w(u, v) < d[v] então
25:                d[v] ← d[u] + w(u, v)
26:                p[v] ← u
27:    se v / S então
28:        Erro: Não foi encontrado um caminho de s até t.
```

4.2.7 Conjunto mínimo de arcos de retorno

O problema do conjunto mínimo de arcos de retorno é um problema combinatório amplamente estudado em grafos, cuja definição pode ser colocada como:

Definição 1. *Seja G um grafo direcionado de arestas E. O menor dos subconjuntos das*

arestas $F \subseteq E$ tal que $G - F$ é acíclico é o conjunto mínimo de arcos de retorno.

O problema é equivalente a ordenar os vértices de tal forma que o menor número de arestas aponte no sentido contrário à ordem elencada. Se o grafo for acíclico, essa ordem será a ordem topológica.

Karp (1972) demonstrou que o problema de obtenção deste conjunto é NP-difícil, não sendo conhecido algoritmo polinomial que retorne a solução exata deste problema.

Sucessivamente, diversos trabalhos propõem heurísticas para agilizar a obtenção de um bom conjunto de arcos de retorno. Simpson, Srinivasan e Thomo (2016) compara onze estratégias em diversos tamanhos de grafos em tempo de execução e tamanho do conjunto obtido. Os tamanhos dos grafos variam de 10^4 a 10^9 vértices, e de 7×10^4 a 4×10^{10} arestas. Este trabalho propõe-se a estudar grafos menores que os analisados por Simpson, e optou-se pelo algoritmo *GreedyFAS* (EADES; LIN; SMYTH, 1993)

Este algoritmo retira sucessivamente da análise os vértices fontes e sumidouros, seguindo a mesma lógica do algoritmo de Kahn, pois estes vértices seguramente não participam de ciclos. Quando não houver mais qualquer vértice fonte ou sumidouro, arbitra-se um vértice como sendo fonte (ou como sumidouro, em algumas variantes), e todas as arestas convergentes neste vértice são consideradas de retorno.

O critério de escolha de vértice deve ser aquele que, ao mesmo tempo que possua poucas arestas convergentes, possua uma quantidade razoável de arestas divergentes. No GreedyFAS, é escolhido o que detiver a maior diferença entre os graus de saída e de entrada do vértice.

Uma vez retirado esse vértice arbitrado, volta-se a pesquisar por fontes e sumidouros, repetindo o processo toda vez que é encontrado um ciclo. Com isso, o algoritmo termina assim que observar todos os vértices.

Algoritmo 6 Conjunto mínimo de retorno – GreedyFAS

Entrada Grafo $G = (V, E)$

```
1:  procedimento GreedyFAS(G)
2:      Inicie uma lista vazia L
3:      repita
4:          para v ∈ V faça
5:              Calcule o grau de entrada d⁻(v) de v
6:              se d⁻(v) = 0 então
7:                  G ← G − {v}
8:                  Finalize o para
9:              Calcule o grau de saída d⁺(v) de v
10:             se d⁺(v) = 0 então
11:                 G ← G − {v}
12:                 Finalize o para
13:             v ← o vértice com maior valor d⁺(v) − d⁻(v)
14:             para todo u ∈ N(v) faça
15:                 Adicione (u, v) em L
16:             G ← G − {v}
17:     até G = ∅
18:     Retorne L
```

5 Modelagem do problema

Em Bressan e Artigas (2019) é feita a modelagem inicial deste trabalho. Esta dissertação refina o modelo do artigo e desenvolve a estratégia de solução, elencando propriedades e métodos de solução.

5.1 Modelagem do arranjo submarino

O arranjo submarino do problema possui uma série de informações a armazenar, mas que são irrelevantes para o processo de descomissionamento. Apenas as questões que restringem o encaminhamento da solução devem ser considerados. A informação mais trivial para a execução do recolhimento de qualquer duto é o seu comprimento, mesmo que não haja qualquer restrição a sua execução.

A primeira restrição que deve ser elencada é a possibilidade técnica do recolhimento. Para que um recolhimento seja iniciado, é necessário que haja uma extremidade amigável para o içamento. Em um duto flexível essa função é exercida pelo conector. Na ausência de uma interface amigável, ocorre um dispêndio maior de recursos para criá-la, com ferramentas dedicadas e mais lentas. Com isso, o duto precisa minimamente das informações de comprimento e o estado de cada uma das duas extremidades.

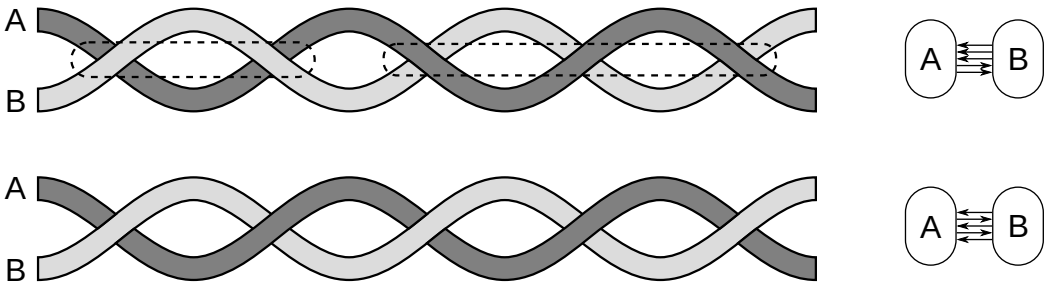
A restrição mais forte para dutos flexíveis é a existência de cruzamentos. Um cruzamento caracteriza uma relação de precedência entre dois dutos. O duto que estiver por cima necessariamente deve ser movimentado antes do duto que estiver por baixo, inferindo uma ordenação. Uma entidade matemática que descreve bem relações de precedência é um grafo direcionado.

Dois dutos podem cruzar múltiplas vezes, levando essa relação de precedência a envolver os mesmos dutos diversas vezes, e, inclusive, em sentidos opostos. Neste caso um grafo simples não é suficiente para modelar essa relação de precedência, exigindo um multigrafo direcionado.

Outro aspecto importante da restrição de cruzamento é relacionado à ordem em que estes eventos ocorrem em um duto. Na Figura 11, a quantidade de cruzamentos dos dois arranjos é idêntica, mas o agrupamento dos cruzamentos pela ordem torna o primeiro arranjo muito mais simples de resolver, bastando apenas uma movimentação. Então, a ordem desses eventos deve ser armazenada. O multigrafo proposto deve possuir mais um atributo relacionado à posição do evento nos dois dutos envolvidos.

Um critério simples para ordenar estes eventos é colocar a cota (distância da extremidade) da ocorrência. Além disso, no contexto de movimentação de dutos, a distância

Figura 11 – Entrelaçamentos distintos com o mesmo grafo.



O arranjo de cima possui os cruzamentos com o mesmo duto por cima juntos, agrupados, enquanto o arranjo de baixo possui os cruzamentos alternando o duto por cima. Com isso, o arranjo de cima pode ser resolvido por um único remanejamento, removendo todo o grupo de cruzamentos indicados pelas linhas pontilhadas. No arranjo de baixo só é possível remover um cruzamento de cada vez. Fonte: Autor (2020)

surge naturalmente quando tem-se os custos de modificação do leiaute submarino. A questão da distância pode ser visualizada na voltando à Figura 5. No laço indicado na Figura 5a, pode-se retirar o cruzamento movendo tanto da forma da Figura 5b quanto da forma da Figura 5c. A movimentação na Figura 5b é muito mais longa que na Figura 5c, e, portanto, tende a possuir um custo maior.

Com isso, o multigrafo precisa carregar informações sobre os dois dutos envolvidos, a ordem do cruzamento (quem está por cima) e as cotas nos dutos envolvidos. Um arranjo submarino pode ser definido formalmente por:

Definição 2. *Defini-se um duto por* $P = (l_p, b_p, e_p)$ *com* $l_p \quad \mathbb{R}_+$, $b_p \quad \{0,1\}$ *e* $e_p \quad \{0,1\}$.

Definição 3. *Um arranjo submarino é definido por um conjunto de dutos* V *e suas relações de cruzamentos* $A = (P, Q, k_P, k_Q) \quad V^2 \times \mathbb{R}^2$, *com* $P \quad V$, $Q \quad V$, $0 < k_P < l_p$ *e* $0 < k_Q < l_q$.

Em que:

P é o duto que está por cima no cruzamento;

Q é o duto que está por baixo no cruzamento;

l_p é o comprimento do duto P;

l_q é o comprimento do duto Q;

b_p indica existência de uma interface amigável na extremidade de cota[1] zero;

e_p indica existência de uma interface amigável na extremidade oposta à cota zero;

[1] É rotineiro com dutos flexíveis estipular uma extremidade para ser a referência de comprimento. Esse comprimento pode ser usado para localizar um evento no duto, como um defeito ou um dano. A distância entre um ponto do duto e essa extremidade de referência é chamada cota.

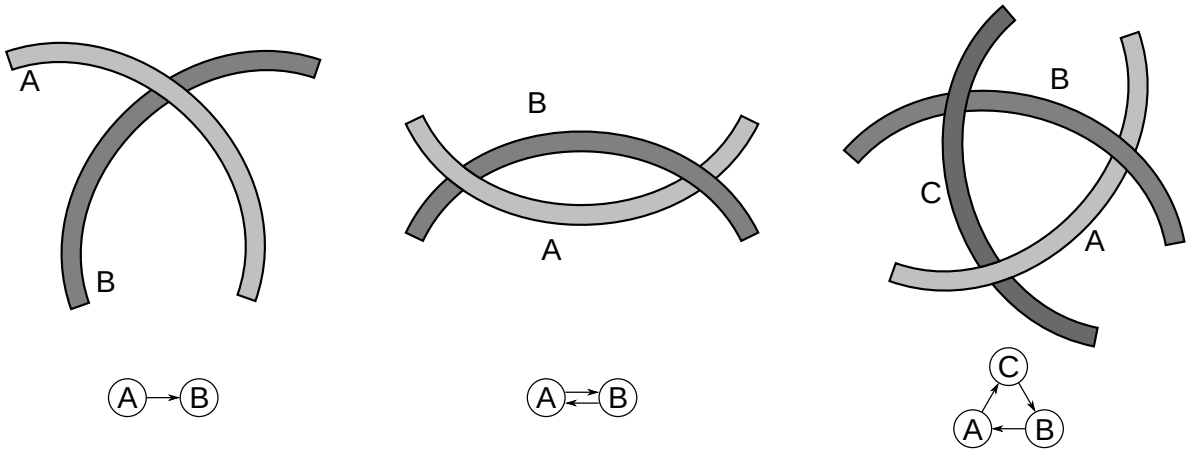
k_P é a cota em que ocorre o cruzamento no duto P;

k_Q é a cota em que ocorre o cruzamento no duto Q.

As desigualdades $0 < k_P < l_p$ e $0 < k_Q < l_q$ forçam um cruzamento esteja dentro do comprimento do duto. É importante ressaltar que P e Q podem estar representando um mesmo duto, caso que ocorre quando ocorre um *loop* no duto. Outro fato importante é que a referência para o cruzamento deve ser sempre a mesma extremidade, e não a mais próxima.

Exemplos de como são representados os cruzamentos podem ser encontrados na Figura 12.

Figura 12 – Representação de arranjo submarinos em grafos.



Fonte: Autor (2020)

5.2 Movimentos entre arranjos

Uma vez conseguido representar um arranjo submarino, agora passar-se-á para a forma de representar o avanço da solução. O recolhimento dos dutos flexíveis submarinos tem poucos movimentos efetivos. Os movimentos possíveis para um duto qualquer são:

Recolhimento – Retirar um duto integralmente, o objetivo do problema. Esta ação é possível quando o duto não apresenta nenhuma restrição de recolhimento.

Remanejamento – Recolher um duto parcialmente, e relançar em uma rota distinta. Essa ação é possível a partir de extremidades, e o tamanho do recolhimento é ditado pela presença de cruzamentos sobre o tramo.

Desconexão – Separar flanges[2] em uma conexão presente no duto, sem içar o duto. Essa ação pode auxiliar na descoberta de trechos parcialmente recolhíveis, sem colocar penalidades de recolhimento. Este movimento não será considerado neste trabalho, pela dificuldade em associar as conexões aos tramos corretos na base utilizada.

Corte – Cortar o corpo tubular de um duto. O corte de um duto retira o cruzamento naquele ponto, deixando as extremidades com maior dificuldade recolhimento, transformando um duto em dois.

A existência de mais do que o movimento do objetivo indica a presença de restrições a essa ação, desencadeando as demais. A principal restrição ao problema é o cruzamento entre dutos no fundo do mar, que impede o recolhimento.

A cada movimento, um novo arranjo é gerado, pois em um recolhimento retira-se um elemento do sistema e nos demais movimentos as relações de interdependências são alteradas.

É importante ressaltar que é possível que movimentos distintos gerem o mesmo arranjo. Um caso simples em que isso ocorre é quando são executados dois cortes seguidos. A ordem em que os cortes ocorrem não altera o arranjo final.

Então, cada arranjo gerado nas etapas de solução possui pelo menos um ancestral, e cada arranjo ancestral possui o movimento associado que levou o arranjo ancestral a ele.

Com essas características, pode-se criar um grafo, direcionado, relacionando os arranjos. Como cada movimento possui um custo associado, esse grafo torna-se ponderado. Neste novo grafo, cada arranjo é um vértice, e o movimento uma aresta.

A partir dessa modelagem, existe um estado inicial, com todos os dutos no arranjo submarino. Por outro lado, um conjunto vazio de dutos a recolher representa a meta. A solução do problema é, portanto, o caminho mínimo desse grafo de arranjos e movimentos.

Assim, a modelagem do problema passa a ser um digrafo em que cada vértice é um multigrafo. Com o exposto até aqui, falta apenas definir as consequências de cada movimento no multigrafo.

5.3 Movimentos entre estados

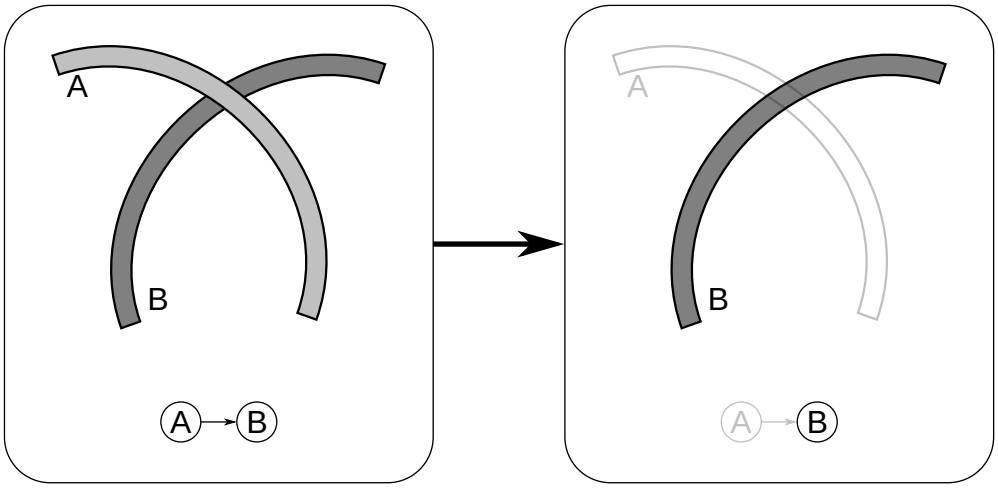
Já foram detalhados os movimentos possíveis na resolução do problema, mas ainda não foram apresentados como os movimentos alteram os multigrafos de arranjo. Estes movimentos também seguem o estudo de Bressan e Artigas (2019).

O movimento mais simples é o de recolhimento. Em um recolhimento, o multigrafo simplesmente tem os vértices relativos aos dutos recolhidos eliminados do grafo, resultando

[2] Componente que permite unir dois tubos e separá-los sem danos irreversíveis.

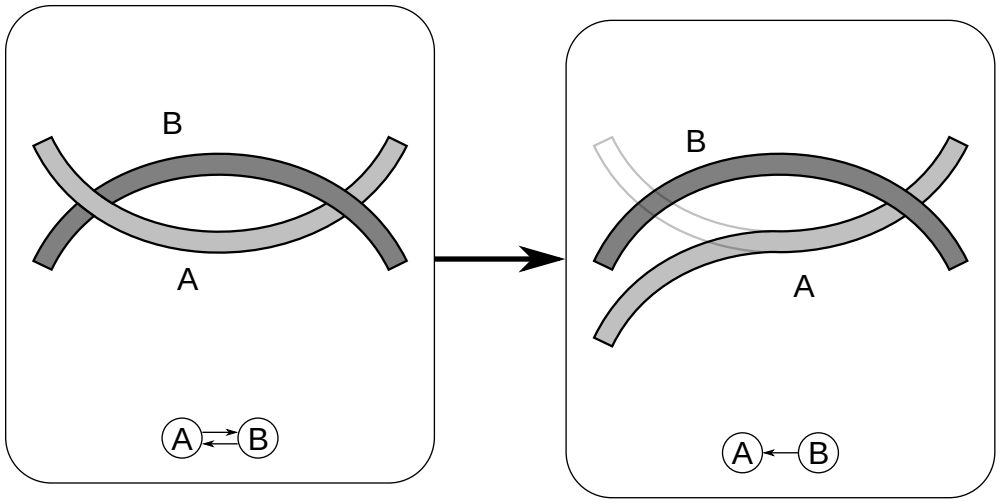
no grafo induzido pela remoção dos vértices. A Figura 13 dá uma visualização de uma remoção em um grafo, no caso, a remoção do duto A.

Figura 13 – Visualização do recolhimento no grafo



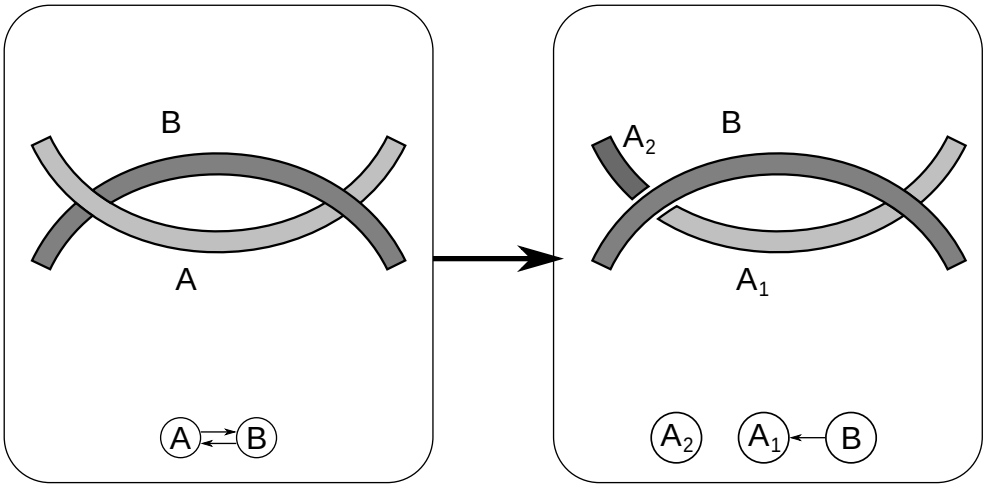
O remanejamento não altera os vértices do multigrafo, se limitando a remover arestas do mesmo. Os critérios para remoção das arestas não ficam explícitos apenas pela topologia, mas sim, pelas informações carregadas em cada aresta. A Figura 14 expõe a alteração no multigrafo ocorrida nesse movimento.

Figura 14 – Visualização do remanejamento no grafo



O corte causa alterações de maior magnitude no multigrafo. Ao executar um corte, passam a existir dois dutos, cada um sem pelo menos um dos conectores. Assim, um vértice do grafo é transformado em dois. O corte sempre ocorre em um cruzamento, logo, o cruzamento cortado deixa de existir como aresta. As arestas restantes são divididas entre os dois vértices novos, a depender de sua posição em relação ao cruzamento cortado. A Figura 15 permite uma compreensão melhor do efeito.

Figura 15 – Visualização do corte no grafo



5.4 Custos envolvidos

Existem diversos fatores que influenciam nos custos envolvidos, desde o preço de referência do petróleo até a localização dos elementos a movimentar. Em um descomissionamento, o projeto é executado em um curto espaço de tempo, no mesmo campo — ou seja, nas mesmas condições ambientais. Assim, é bastante factível considerar apenas aspectos operacionais.

Um recolhimento possui duas componentes de tempo:

- Resgatar a extremidade do fundo e posicionar a linha no tensionador
- Trazer o duto do tensionador até a área de estocagem do navio.

A primeira componente é essencialmente característica do sistema de içamento do navio, e depende muito mais da profundidade do resgate e a característica da extremidade que dos atributos da linha. Nesse sentido pode-se colocar um componente fixo, relativo ao içamento.

Para executar um içamento, caso a extremidade não possua um apoio adequado — um duto cortado — este recolhimento requisitará maior esforço e ferramentas para sua execução. Então, há um componente de penalidade caso um duto não possua nenhuma das duas extremidades adequadas ao recolhimento.

A passagem do duto no tensionador tem uma velocidade aproximadamente constante, função da característica do conjunto de lançamento. Assim, este componente é ditado fortemente pelo comprimento da tubulação a recolher.

Para um remanejamento, as mesmas considerações são válidas. Algumas diferenças são essenciais:

- O duto parcialmente recolhido fará o caminho inverso para ser novamente depositado. Assim, o custo relativo ao comprimento é maior.
- O remanejamento depende de qual extremidade será içada, logo a penalidade de içamento pode ser aplicada mesmo que a outra extremidade esteja em condições de içamento.

O corte tem um aspecto diferenciado do ponto de vista de custos. Primeiramente, a embarcação para executar um corte pode ser muito mais simples: ela apenas precisa de um ROV. Outro ponto importante na definição dos custos é a indiferença da posição em que ocorre o corte para o tempo de execução. Grande parte do tempo gasto para o corte é na preparação — descida de ROV, aproximação, erosão do solo — tornando o cisalhamento efetivamente uma parcela pequena no custo total. Assim, é razoável pensar que o corte tem custo constante.

Outra característica do corte se refere não mais ao corte em si, mas nas consequências dele: ele é o principal fator para executar a penalidade no recolhimento. O corte também torna o recolhimento posterior mais caro por aumentar um içamento.

Assim, são propostos como custos as seguintes funções lineares com o comprimento:

$$\mathit{Recolhimento} = I + M_r + R_c \cdot L \tag{5.1}$$

$$\mathit{Remanejamento} = I + M_r + R_c \cdot L + R_a \cdot L + D + M_d \tag{5.2}$$

$$\mathit{Corte} = C \tag{5.3}$$

Em que:

D Custo de abandono da extremidade

I Custo de içamento da extremidade

L Comprimento movimentado

M_r Penalidade de recolhimento

M_d Penalidade de abandono

R_c Custo por comprimento recolhido

R_a Custo por comprimento abandonado

C Custo por corte realizado

É possível realizar uma comparação entre os dois métodos de remoção de cruzamento: corte e remanejamento. Dado que um corte produz a necessidade de um novo recolhimento

e possivelmente uma penalidade, e que um remanejamento pode remover n cruzamentos em uma única rodada, um recolhimento é mais atrativo quando:

$$I + M_r + R_c \cdot L + R_a \cdot L + D + M_d < \frac{\square}{n}(C + I + M_r) \tag{5.4}$$

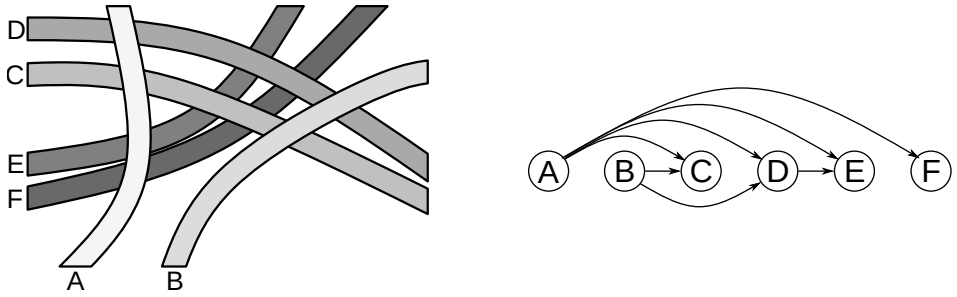
5.5 Consequências da modelagem

Uma vez que o duto é representado por um vértice em um arranjo de dutos, e que um cruzamento de dutos é representado em um grafo por uma aresta divergente do duto que está por cima e convergente no duto que está por baixo, pode-se colocar algumas consequências desta modelagem.

A primeira consequência é que um duto pode ser recolhido se e somente se o vértice que o representa for fonte. Se existisse alguma aresta incidente neste vértice, o duto estaria impedido de ser recolhido pelo cruzamento representado por esta aresta incidente. Portanto, a ação de recolher um duto só se aplica aos vértices fontes do grafo de arranjo.

Indo mais adiante, pode-se estender a consequência anterior para um agrupamento de dutos. Um grupo de dutos pode ser recolhido sem realizar remanejamentos ou cortes caso o seu grafo representativo seja acíclico, como na Figura 16.

Figura 16 – Recolhimento em ordem topológica



Se em um arranjo há um duto livre, ou seja, um vértice fonte, então pode-se retirá-lo sem quaisquer restrições. Com a retirada deste duto, três hipóteses podem ocorrer:

1. O subgrafo gerado pela retirada possui um vértice fonte, ou seja, um duto se tornou livre com a retirada. Repete-se o procedimento até que esta hipótese não seja mais atendida.

2. Haja dutos no leito marinho, e em todos há pelo menos um cruzamento em que este esteja por baixo. Nesse caso, o grafo deste estado não possui qualquer vértice fonte, caracterizando um ciclo, e os dutos não são recolhíveis.

3. Este era o último duto, e o grafo agora é vazio. Com isso, todos os dutos foram retiradas.

O procedimento acima equivale a executar o algoritmo de Kahn de ordenação topológica. Com isso, fica demonstrada a equivalência entre o recolhimento de um grupo de dutos sem remanejamento ou corte e a inexistência de um ciclo no grafo representativo do arranjo.

Outra inferência que pode-se fazer da modelagem trata dos vértices sumidouros. Estes vértices não possuem remanejamento efetivo no problema. Isso é explicado pois se o vértice é sumidouro, todo cruzamento que o duto possui ele está por baixo. Assim, qualquer movimentação de extremidade é interrompida pela existência do cruzamento representado pela aresta.

Então, na pior das hipóteses, a movimentação toma um trecho sem cruzamento, e coloca sobre um duto existente, aumentando o número de cruzamentos. Na melhor hipótese, o duto é recolocado em uma rota que não gera cruzamento, por exemplo, sua rota antiga. Portanto, como não ocorre redução do grafo, esta movimentação não contribui com o avanço da solução.

Ainda em remanejamento, a operação é possível quando o duto possui cruzamentos por cima em locais mais próximos da extremidade do que o seu cruzamento por baixo mais próximo da mesma extremidade. Em termos matemáticos, pode-se dizer que um remanejamento de P é possível a partir da primeira extremidade quando:

$$(P, Q, k_{PQ}, k_{QP}) \quad E \text{ tal que } \quad (R, P, k_{RP}, k_{PR}) \quad E, \; k_{PQ} < k_{PR}. \tag{5.5}$$

Em que k_{XY} é a cota do cruzamento do duto X quando cruza-se com Y. Da mesma forma, na segunda extremidade há remajenamento quando:

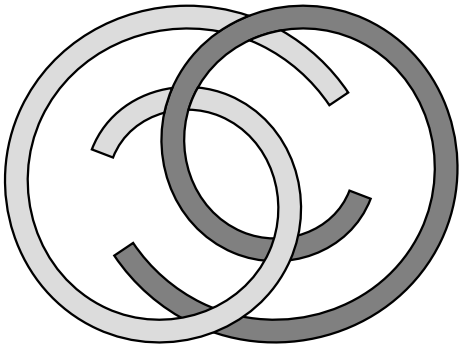
$$(P, Q, k_{PQ}, k_{QP}) \quad E \text{ tal que } \quad (R, P, k_{RP}, k_{PR}) \quad E, \; k_{PQ} > k_{PR}. \tag{5.6}$$

É possível que haja entrelaçamento de dutos em que não exista movimento efetivo para remover cruzamentos. É uma configuração incomum, mas detectada na prática por uma mudança na forma de lançamento dos dutos: antigamente era comum o lançamento de múltiplos dutos concomitantemente, e em determinados projetos de manutenção, o laço era completado. A Figura 17 mostra um caso em que um remanejamento é impossível.

O conjunto de arestas de retorno delimita o escopo de cortes que resolvem a retirada completa de um arranjo. O número de cortes necessários para permitir o recolhimento total de um conjunto de dutos sem remanejamento é menor ou igual ao número de arcos pertencentes ao conjunto mínimo de arcos de retorno do grafo.

Como abordado anteriormente, um conjunto de dutos é recolhível se o grafo de cruzamentos não possuir ciclo. Ao executar um corte, a aresta relativa ao cruzamento

Figura 17 – Configuração impossível de remanejamento



O entrelaçamento acima faz com que o primeiro cruzamento a partir de qualquer ponta nos dois dutos esteja por baixo, impedindo a execução do recolhimento. Neste caso, a solução ocorre com pelo menos um corte. Fonte: Autor (2020)

é removida. Assim, retirando os cruzamentos do conjunto de arcos de retorno, ou seja, executando os cortes conforme este conjunto, é garantido que o grafo de cruzamentos não contenha mais ciclos.

Então, um conjunto com o mínimo de arcos de retorno é um limitante superior do número mínimo de cortes. Entretanto, não pode-se afirmar que o número é o mínimo visto que no corte, há substituição de um vértice por dois, distribuindo as arestas do duto cortado entre os dois vértices novos, de acordo com as suas posições.

Nessa distribuição, os dois vértices podem tornarem-se fontes e o grafo possui recolhimento. Um exemplo em que o conjunto mínimo de retorno não é o mínimo de cortes necessários está na Figura 18.

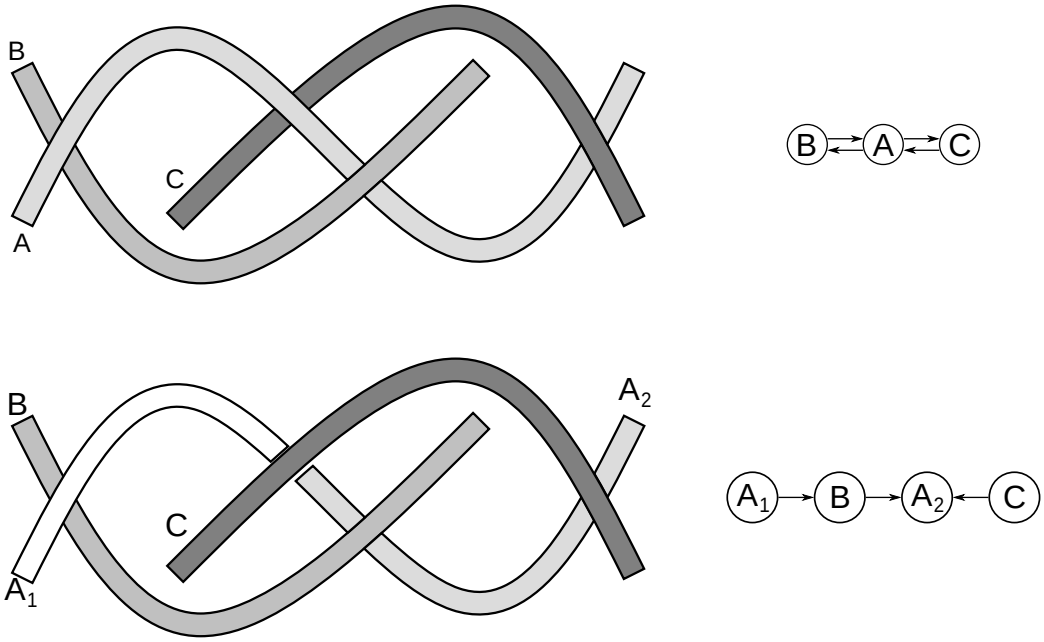
Aprofundando o conceito de ordem topológica, pode-se simplificar instâncias do problema utilizando dos componentes fortemente conexos.

Os componentes fortemente conexos formam uma partição do grafo em que cada uma destas partições não contém nenhum vértice fonte, nem sumidouro. Com isso, é garantido que em um componente fortemente conexo não seja possível realizar nenhum recolhimento. Então o conjunto solução desse componente fortemente conexo passa necessariamente por remanejamento ou corte.

Por outro lado, essa partição de componentes fortemente conexos cria um grafo induzido das suas partições. Esse grafo induzido é necessariamente acíclico. Então, considerando a obtenção da solução de cada componente isoladamente, basta seguir a ordem topológica das soluções obtidas para cada uma das partições para obter a solução do problema como um todo.

Os vértices deste grafo induzido são traduzidos em recolhimentos dos tramos do grupo de dutos que está sobre outro grupo, liberando o avanço da solução até sua finalização, dando tratamento a todo o problema. Portanto, a solução do problema de um estado é

Figura 18 – Configuração com dois arcos de retorno e um corte é solução

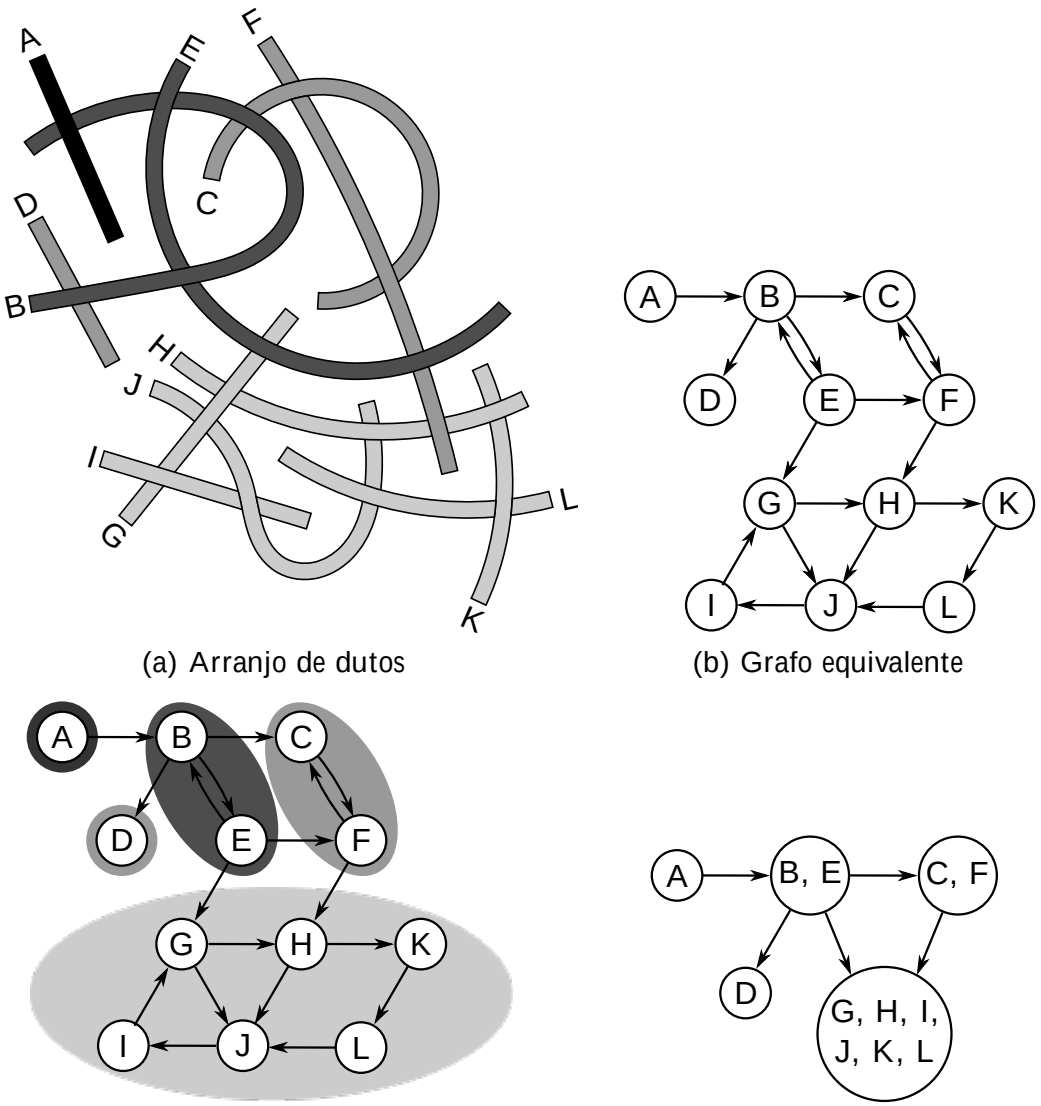


Esta é uma configuração em que o grafo possui duas arestas de retorno (uma entre A e B e outra entre A e C). O corte do cruzamento entre A e B, gerando os dutos A_1 e A_2, é suficiente para ocorrer um recolhimento na ordem C, A_1, A_2 e B. Fonte: Autor (2020)

uma combinação das soluções dos seus componentes fortemente conexos, executadas na ordem topológica destes componentes.

Um exemplo deste agrupamento é feito na Figura 19a, cujo grafo equivalente é aquele da Figura 19b. Considere que a meta é o recolhimento total. Primeiramente, faz-se a separação dos componentes conexos como na Figura 19c.

Figura 19 – Componentes fortemente conexos



(a) Arranjo de dutos

(b) Grafo equivalente

(c) Componentes fortemente conexos

(d) Ordem das soluções parciais

Uma configuração de dutos, e sua conversão em diversos problemas menores, utilizando seus componentes fortemente conexos. Fonte: Autor (2020)

Soluciona-se individualmente cada componente fortemente conexo. Para os componentes que possuem apenas um duto, basta recolhê-lo, enquanto os demais grupos necessitarão de tratamento adicional. Finalmente, após resolver cada grupo, determina-se a solução total colocando sucessivamente a solução dos subproblemas na ordem topológica desses componentes, como em 19d. Um exemplo de solução válida para este problema é:

1. Solução de $\{A\}$:
    1. Recolher A
2. Solução de $\{B, E\}$:

2. Remanejar E
3. Recolher B
4. Recolher E

3. Solução de $\{D\}$:

4. Recolher D

4. Solução de $\{C, F\}$:

5. Remanejar C
6. Recolher F
7. Recolher C

5. Solução de $\{G, H, I, J, K, L\}$:

8. Remanejar I
9. Recolher G
10. Recolher H
11. Recolher K
12. Recolher L
13. Recolher J
14. Recolher I

5.6 Existência de solução

Já foi descrito como é uma fotografia do fundo marinho, e como cada uma das ações — recolher, remanejar e cortar — altera o arranjo submarino, e como valorar cada uma das ações. Agora resta pesquisar um caminho de solução.

Dado um arranjo qualquer, há um número finito de estados que são atingíveis pela solução. Se há um duto fonte no arranjo, então é possível realizar o recolhimento deste duto. Caso contrário, pode-se ir por dois caminhos: remanejar uma extremidade, em que há no máximo o dobro do número de dutos, ou cortar um cruzamento, em que o número de cortes possíveis é no máximo o número de cruzamentos.

Apesar de que eventualmente exista uma configuração sem remanejamentos possíveis, é sempre possível executar um corte. Assim, sempre existe pelo menos um vizinho, a menos que o arranjo seja o arranjo vazio.

A garantia da existência de um vizinho faz com que seja sempre possível utilizar o Algoritmo 7 para a solução do problema.

Algoritmo 7 Descomissionamento de dutos (ingênuo)

Entrada $G = (P, X)$ arranjo submarino, e T a meta para cada duto de P.

```
1: procedimento Descomissionar(G, T)
2:     enquanto T não é atendido faça
3:         Calcule os graus de entrada do arranjo atual
4:         se Existe algum vértice fonte s então
5:             Recolha(s)
6:         senão
7:             s ← o duto com menor grau de entrada
8:             x ← um cruzamento c em que o está sobre s
9:             Cortar(x, o)                    Cortar o duto o no cruzamento x
```

O algoritmo acima termina pois o arranjo troca um cruzamento por um duto em um corte, reduzindo o número de arestas e aumentando o número de vértices. Então, eventualmente, algum vértice se tornará fonte ou isolado. Uma vez que isso ocorra, o vértice é removido do grafo, reduzindo com certeza o número de vértices, e possivelmente o de arestas, no caso de o vértice não ser isolado. Assim, é garantido que há uma solução para o problema.

5.7 Busca pela solução ótima

O Algoritmo 7 garante a existência de um caminho entre um arranjo inicial dado e a meta, através de operações de corte e recolhimento, mas dada a possibilidade de ocorrer remanejamento, ou outros cortes serem mais efetivos para a solução esta solução provavelmente não é ótima.

A solução do problema então é encontrar um caminho mínimo em um grafo que inicia de um arranjo informado, fonte, e tem seus vizinhos ponderados pelas operações de recolhimento, remanejamento e corte. Esta pesquisa é possível com o algoritmo A*, discriminado anteriormente, e este será aplicada nesta dissertação. Falta apenas criar uma estimativa para a solução — a função `Estimador` no Algoritmo 4.

Para o problema de recolhimento, a única certeza para a solução é que os dutos serão recolhidos. Assim, a primeira heurística para solução é considerar que os demais tramos estão sem qualquer cruzamento sobre eles.

Esta estimativa é otimista, no sentido que na melhor hipótese, o movimento realmente livrou todos os dutos e eles foram recolhidos, satisfazendo o critério de admissibilidade.

Esta proposta também é consistente, uma vez que qualquer estado, caso seja remanejado, terá o custo de remanejamento e o próximo estado terá a mesma estimativa de solução. Caso seja um corte, a estimativa aumenta pelo içamento adicional e possivelmente

pelas penalidades aplicadas.

Uma estimativa mais precisa deverá levar em conta os cruzamentos. Há uma dificuldade para estabelecer um critério consistente por dois aspectos:

- A função de corte é correlacionada ao conjunto de arcos de retorno, mas além de o problema do conjunto de arcos de retorno ser NP-*difícil*, não é conhecido um algoritmo que possa aproximar em tempo polinomial uma fração arbitrária dos arcos.
- A função de remanejamento não opera em uma quantidade pré-determinada de arcos, e a depender da quantidade de arcos removidos, ela pode ser mais vantajosa para a solução que os diversos cortes.

O custo de um recolhimento não depende de quando ele é executado. Assim, ao se descobrir um duto fonte, apenas um vizinho deve ser explorado: recolhê-lo. O recolhimento pode revelar novos vértices fonte, prosseguindo sucessivamente como no algoritmo de Kahn, até que não haja mais vértices fonte. Não havendo vértice fonte, ocorre a pesquisa pelas demais operações. Esta abordagem está demonstrado no Algoritmo 8.

Algoritmo 8 Descoberta de vizinhos

Entrada $G = (P, X)$ arranjo submarino.

```
1: procedimento Descobrir Vizinhos(G)
2:     R ← Kahn(G)
3:     se R =  então
4:         Recolher(R)
               Retorna o peso da aresta e o arranjo resultante de recolher os dutos de R
5:     senão
6:         M ←Remanejamentos(G)
       Retorna os pesos e os arranjos resultantes de cada remanejamento possível em G
7:         C ←Cortes(G)
                 Retorna os pesos e os arranjos resultantes de cada corte possível em G
8:         Retorne M   C
```

Com isso, pode-se completar o algoritmo de pesquisa da solução ótima para o recolhimento de dutos, combinando o A* (Algoritmo 4) e os movimentos do problema como no Algoritmo 9.

Algoritmo 9 Descomissionamento de dutos (A*)

Entrada Arranjo inicial s, meta t.

```
1:  função Solução(s, t)
2:      Inicie uma fila de prioridade Q = {s}
3:      d[s] ← 0
4:      h[s] ← Estimador(s)
5:      repita
6:          Retire de Q o arranjo u de menor d[u] + h[u]
7:          se u atende t então
8:              R ← Reconstruir(u)
9:              Retorne R
10:         senão
11:             N ← Descobrir Vizinhos(u)
12:             para todo v ∈ N faça
13:                 se v ∉ Q então
14:                     Q ← Q ∪ {v}
15:                     h[v] ← Estimador(v)
16:                     d[v] ← d[u] + w(u, v)
17:                     p[v] ← u
18:                 senão se d[u] + w(u, v) < d[v] então
19:                     d[v] ← d[u] + w(u, v)
20:                     p[v] ← u
21:     até Q = ∅
22:     Retorne erro.                          A meta não é atingível
```

5.8 Cardinalidade

Uma estimativa grosseira da quantidade de arranjos gerados a partir de um problema é verificar o número de cruzamentos. Cada cruzamento, ao longo da solução, pode estar em três estados possíveis: existente, removido ou cortado. Um duto que já tenha sido removido em um arranjo é equivalente, em termos de solução, ao mesmo arranjo com o duto destacado, isto é, sem qualquer cruzamento.

Assim, pode-se estimar o número máximo de arranjos possível pelo número de cruzamentos simplesmente por 3^{nc}, em que nc é o número total de cruzamentos no estado inicial. Por exemplo, Tainha, com seus 4764 cruzamentos teria $3^{4764} = 10^{2273}$ arranjos distintos (algo em torno de 2^{7551}).

Esta abordagem desconsidera que diversos cruzamentos nunca serão avaliados. Em um arranjo, toda vez que um cruzamento não faz parte de um grupo fortemente conexo, ele simplesmente se converterá em um recolhimento simples. Então os três estados se aplicam àqueles cruzamentos integrantes de um grupo fortemente conexo.

Esta lógica se sustenta no caso extremo: se todo componente fortemente conexo possui apenas um vértice, então ele necessariamente possui uma ordem topológica. E, se

ele possui esta propriedade, há apenas um caminho a seguir: recolher todos os tramos nesta ordem topológica. Evidentemente, um arranjo pode ter mais de uma ordem topológica, mas não é decisivo para sua execução qual ordem será seguida, porque a velocidade de deslocamento do navio entre os movimentos é cerca de 50 vezes maior que a velocidade dentro de um movimento.

Voltando ao exemplo de Tainha, os grupos fortemente conexos possuem no total 4339 cruzamentos. Então a estimativa de arranjos possíveis passa para $1.7 \times 10^{2070} \quad 2^{6878}$. É uma redução bastante considerável.

Separando a solução para cada um dos grupos fortemente conexos, o grupo fortemente conexo de Tainha com o maior número de cruzamentos contém 3722 arestas, o tamanho do espaço de solução máximo que será efetivamente resolvido é de $7 \times 10^{1775} \quad 2^{5900}$.

5.9 Complexidade

O problema apresentado possui um caso extremo relevante: um em que apenas cortes são executados. Neste caso, em que são cortados os cruzamentos até que um grafo acíclico direcionado seja obtido, é bastante relacionado, mas não idêntico, ao problema do conjunto mínimo de arcos de retorno de um grafo. Como já demonstrado na seção anterior, a solução ótima do problema de cortes pode ser menor que o conjunto mínimo de arcos de retorno.

A ordem em que ocorre os cortes pouco importa para a obtenção da solução, visto que os cortes, se executados nos mesmos dutos e nas mesmas cotas produzem o mesmo arranjo submarino final. Também é fato que um corte adicional em um arranjo que já possua uma ordem topológica não altera a condição topológica, pois o corte, como já evidenciado em seção anterior, reduz uma aresta e aumenta o grafo em um vértice. Este excesso apenas torna o recolhimento mais oneroso, por exigir um içamento a mais, ou por tornar um duto cortado em duas extremidades, causando dificuldades em recolhimento.

Outra peculiaridade relevante da operação de corte, é que ela pode ser executada tanto no vértice de origem quanto no vértice de destino, ou o duto de cima quanto o duto de baixo. Então, todo corte precisa ser avaliado duas vezes, com resultados distintos a depender se o tramo a ser cortado é o superior ou o inferior.

Fixando uma convenção em que sempre o duto de cima será cortado, o conjunto mínimo de arcos de retorno do arranjo representa uma solução, possivelmente não-ótima, do problema proposto. Dado o conjunto mínimo de arcos de retorno, pode-se testar se algum corte pode ser suprimido, mantendo a característica de conjunto acíclico, e avançando a remoção destes cortes até que não seja possível retirar cortes sem incluir um arco de retorno.

Dessa forma, o problema proposto está intrinsecamente relacionado ao problema do conjunto mínimo de arcos de retorno. Acredita-se que ambos os problemas tem dificuldade semelhante e que seja possível fazer uma redução polinomial de conjunto mínimo de arcos de retorno para o problema proposto, o que permitiria classificar o problema como NP-difícil. Obter tal redução é um dos problemas em aberto desta dissertação.

6 Aplicação ao problema

Um modelo teoricamente viável para a solução do problema de recolhimento de dutos já foi apresentado. Passar-se-á agora para a implantação do código, para verificar a viabilidade e velocidade de execução do código.

6.1 Implantação

Implantar-se-á aqui o Algoritmo 9, aplicando o mesmo no campo de Tainha, mostrando algumas etapas da codificação e os recursos utilizados.

6.1.1 Arranjos canônicos

Para garantir que o algoritmo construído julgue corretamente os problemas, uma série de testes básicos devem ser executados. Os casos elencados foram desenhados para que estes possam ser verificados manualmente, e assim, validar a solução.

São propostos os seguintes casos de validação, colocando como solução o recolhimento total:

0. Arranjo vazio.
1. Dutos soltos: um conjunto de dutos sem qualquer cruzamento.
2. Dutos empilhados: um conjunto com uma pilha simples de dutos, sem laços.
3. Par torcido: um único par de dutos em que há dois cruzamentos, em sentidos opostos. Duas variantes serão consideradas:
   a) Um dos cruzamentos é muito próximo à extremidade do tramo que está por cima;
   b) Os cruzamentos são próximos entre si, mas distantes de qualquer extremidade;
4. Círculo: um conjunto de dutos, cada um com um cruzamento por cima e um por baixo, de tal forma que o grafo de cruzamentos forme um círculo.

O caso zero testa se o algoritmo reconhece um arranjo vazio, afinal, ele é o resultado esperado para o campo de Tainha.

O primeiro caso é absolutamente trivial. A solução é apenas uma série de recolhimentos, em qualquer ordem. Não é considerado aqui o deslocamento entre os recolhimentos, então a ordem deles não interfere no valor da solução.

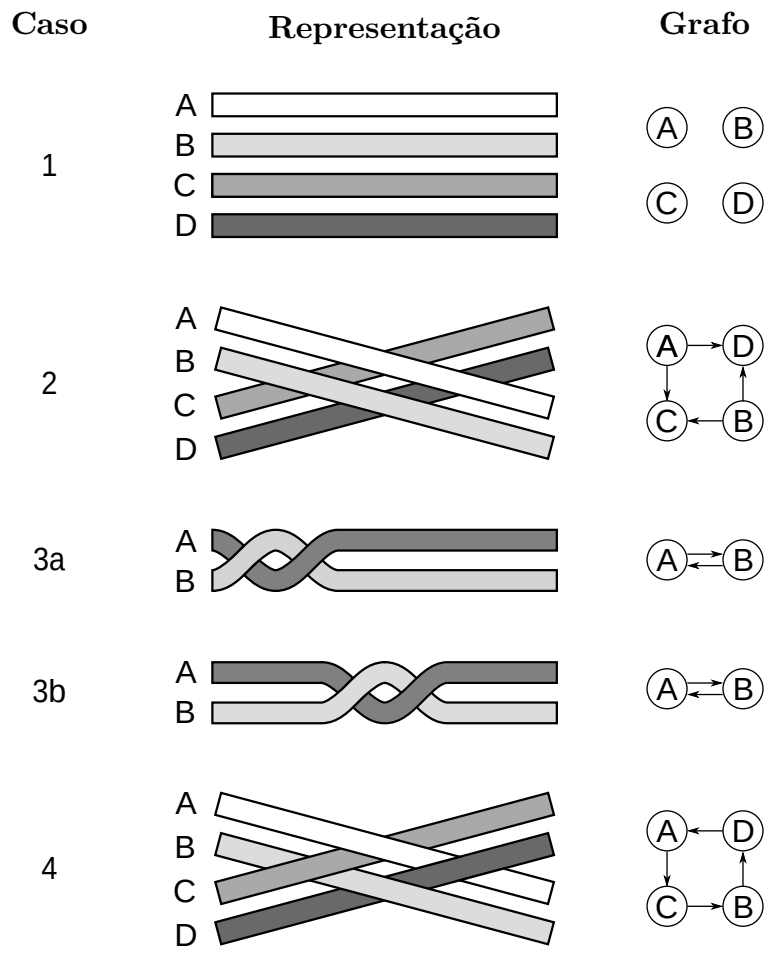
O segundo caso de validação é bastante simples. Uma série de dutos empilhados forma necessariamente uma ordem topológica em seu grafo de cruzamentos. Portanto, o algoritmo deve indicar uma série de recolhimentos respeitando a ordem topológica do grafo inicial. Novamente, caso haja uma escolha de dois dutos possíveis, as duas ordenações devem retornar o mesmo valor.

O terceiro caso de validação é um pouco mais elaborado e visa testar o custo de remanejamento contra o custo de corte do cruzamento. Quando o cruzamento é absolutamente próximo da extremidade, o custo de remanejamento deste duto é menor que a extinção do cruzamento através do corte, colocando a solução como *remanejar o duto B, recolher A e recolher B*.

Por outro lado, quando não há uma extremidade próxima, o custo de remanejamento torna-se maior e supera o custo do corte. A solução para este caso deve envolver corte: *cortar o duto B, gerando os dutos B_1 e B_2, recolher na ordem A, B_1, B_2*.

O último caso de validação testa se o grupo é fortemente conexo. O círculo impedirá inicialmente o recolhimento de qualquer duto. Quando uma extremidade qualquer é remanejada, o grafo de cruzamentos passa a ter uma ordenação total, ou seja, com uma única ordem topológica. Exemplos dos casos estão na Figura 20.

Figura 20 – Casos de validação



Além dos casos de validação acima, outros casos para teste dos algoritmos envolvidos foram utilizados. Merece destaque a implantação do GreedyFAS (Algoritmo 6), que possui potencial para implantação de heurística. Com isso, refina-se o código até que estes casos de validação trouxessem a sequência operacional correta. Validados estes testes, passa-se para o teste com o campo real.

6.1.2 Representação Compacta

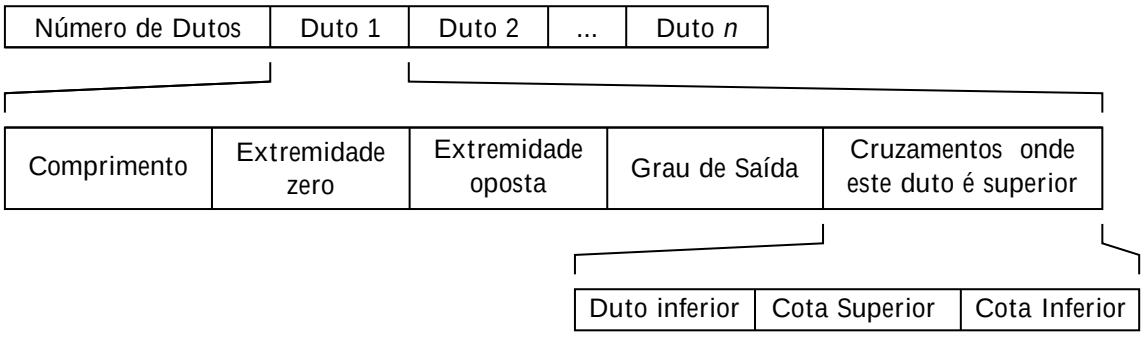
O processo de armazenamento dos arranjos, dada a quantidade de arranjos possivelmente gerados, uma representação compacta para armazenamento destas configurações torna-se necessária. Duas abordagens são mais comuns em grafos: matriz de adjacência ou listas de adjacência. A depender dos algoritmos que processarão o grafo, uma ou outra forma são mais adequadas.

Considerando que o grafo de cruzamentos é bastante esparso, que entre dois dutos podem existir diversos cruzamentos, a abordagem mais compacta para armazenamento de um arranjo é baseada em uma lista de adjacência. Para simplificar a representação, os dutos não serão identificados diretamente, e sim apenas por um índice.

A representação colocará inicialmente o número de dutos. Após esse número, começa-se a representar cada duto, começando com o comprimento, e passando para o estado das extremidades. Logo após, o grau de saída do duto é colocado.

Os cruzamentos podem ser armazenados junto ao duto que estiver por cima, tornando este duto implícito na representação e o duto inferior explícito pelo índice de ordem. Um cruzamento, após a determinação dos dutos envolvidos, deve carregar a informação das cotas de ocorrência, convencionando primeiro o duto por cima, e depois o duto por baixo. A representação construída pode ser visualizada na Figura 21.

Figura 21 – Representação compacta de um arranjo

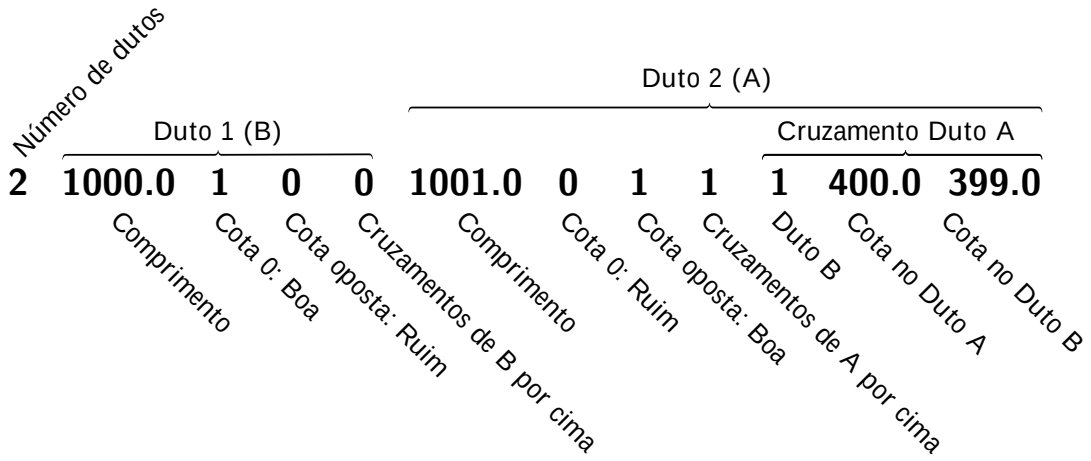


Esquema da representação compacta. Cada duto guardará as suas informações e os cruzamentos em que ele está por cima. Fonte: Autor (2020).

A título de exemplo, a representação do cruzamento da Figura 13, antes do recolhimento, pode ser representada pela Figura 22. Assume-se neste exemplo que o duto

A é ligeiramente maior que o duto B, e que por isso, A será descrito depois de B. Coloca-se também a hipótese de que A e B possuem uma das extremidades com restrição ao içamento.

Figura 22 – Exemplo de representação



Fonte: Autor (2020).

Para os testes executados, a representação compacta foi implantada e permitiu uma redução no armazenamento em aproximadamente dois terços, em relação à manter os objetos em estruturas nativas de classe.

Sem quaisquer regras acerca de ordem para os tubos e cruzamentos, há mais de uma representação possível para um arranjo. Por exemplo, inverter a ordem de dois cortes sucessivos não altera o arranjo, mas possivelmente na forma que é implantado, altera-se a ordem dos tramos cortados.

A fim de tornar a representação de um arranjo única, algumas convenções de ordem devem ser adotadas. Para os dutos, uma convenção satisfatória é ordenar os tubos pelo seu comprimento. Para os cruzamentos, um critério semelhante pode ser usado: ordenar pela cota de ocorrência. Com essas duas diretrizes, a representação binária de um arranjo se torna única.

Caso haja conflito de comprimento, pode-se forçar um desempate numa ordem de grandeza muito pequena comparado com o comprimento, impondo um erro insignificante na alocação de custos. O mesmo pode ser feito nas cotas de cruzamentos: pode-se introduzir propositalmente um pequeno erro na posição do cruzamento, de tal forma que ao executar um corte, não apareça um conflito de comprimento.

A representação dos cruzamentos em ordem de ocorrência também auxilia no processamento, visto que a operação “remanejar” é fortemente ligada à ordem dos eventos em um duto. Para as operações “remover” e “cortar”, a ordem de processamento não é relevante, visto que as operações independem da ordem dos eventos.

6.1.3 Código

Para a implantação do código, foi escolhida a linguagem Python, versão 3.8. A escolha da linguagem se dá pela simplicidade de implantação, fácil portabilidade entre sistemas, e grande quantidade de documentação disponível. Evidentemente, o uso desta linguagem penaliza a velocidade de execução em relação a outras linguagens compiladas.

Nos testes, computador utilizado possui as seguintes características:

- Processador: 4 núcleos, 2,2 GHz, 4 MB cache
- Memória: 8 GB
- Armazenamento: SSD 240 GB.

Primeiramente, uma versão que mantinha os arranjos completos na memória foi implantada. Esta implantação, apesar de mais rápida, causou estouro de memória muito rapidamente, após cerca de 12000 arranjos. Uma tentativa para diminuir o consumo de memória foi então executada, destruindo os objetos e armazenando os arranjos na notação compacta. Esta modificação teve ganho em relação à primeira tentativa, armazenando cerca de 45000 arranjos, mas ainda insuficiente para o problema.

Com a exaustão da memória, uma estratégia para colocar os arranjos em disco teve de ser implantada. A utilização de uma chave primária simples, ou seja, colocando em disco arquivos como `1.dat`, `2.dat` e `3.dat`; e na fila, em memória '`[1, 2, 3]`' já seria o suficiente para tratar o problema da memória.

A abordagem acima é bastante inconveniente para encontrar se um arranjo já está na fila de prioridade. Para contornar esta inconveniência, foi utilizada uma função de dispersão (*hash*), colocando apenas o resultado dessa função na memória e o arranjo completo em um arquivo correspondente em disco.

6.2 Testes em escala real

Feita a codificação do algoritmo de resolução, e validando os resultados nos casos controlados, passar-se-á a testar a implantação contra o caso real escolhido, o campo de Tainha. Descreve-se primeiramente o campo, e a seguir, aponta-se os resultados encontrados neste campo.

6.2.1 O campo de Tainha

O campo de Tainha possui 374 dutos, e, como já informado na Seção 5.8 — Cardinalidade, possui em toda a sua extensão 4764 cruzamentos. Estes dutos possuem

uma configuração bastante entrelaçada, em que o maior grupo fortemente conexo possui 176 dutos e 3722 cruzamentos. Os principais dados dos componentes fortemente conexos de Tainha estão na Tabela 2.

Tabela 2 – Componentes fortemente conexos de Tainha

Componente	Dutos	Cruzamentos
1	176	3722
2	29	423
3	9	66
4	4	45
5	3	20
6	2	18
7	5	15
8	2	10
9	3	8
10	3	5
11	2	3
12 a 146	1	0

Fonte: Empresa Operadora de Tainha. (2019)

Para se ter ideia do nível de entrelaçamento do Tainha, a Tabela 3 descreve o tamanho do maior componente fortemente conexo (abreviado SCC) encontrado em cada um dos cerca de 90 campos atualmente operados pela Petrobras. É nítido que o terceiro maior componente de Tainha (Tabela 2) é da mesma ordem de entrelaçamento que o campo de Traíra da Tabela 3, o 12º campo mais entrelaçado daquela empresa. Cabe lembrar que, novamente, os nomes dos campos são fictos.

6.2.2 Parâmetros de teste

Os valores reais para os parâmetros da Equação 5.1 foram obtidos junto à operadora do campo de Tainha. Entretanto, por questão de sigilo, estes valores não podem ser utilizados diretamente. Por conseguinte, uma unidade monetária de referência (U.M.) foi atribuída para os testes. Os valores das variáveis utilizados nos testes foram:

$D = 1$ U.M.

$C = 1$ U.M.

$I = 1$ U.M.

$M_r = 4$ U.M.

$M_d = 4$ U.M.

Tabela 3 – Maiores entrelaçamentos, por campo

Campo	Dutos no maior SCC	Cruzamentos no maior SCC
Tainha	176	3722
Baiacu	173	2209
Jamanta	99	1318
Enguia	104	1293
Manjuba	63	1053
Tilápia	91	592
Miraguaia	44	570
Sarda	43	186
Poraquê	11	156
Truta	12	109
Cachara	13	84
Traíra	9	80
Mandi	11	70
Salmão	13	60
Piaba	5	57

Fonte: Empresa Operadora de Tainha. (2019).

$R_c = 36$ U.M./km

$R_a = 36$ U.M./km

6.2.3 Resultados Iniciais

O Algoritmo 9 foi colocado para resolver o problema de Tainha. O arranjo geral deste campo foi separado em seus componentes fortemente conexos e tratados individualmente, como se fossem campos separados, seguindo o discutido na Seção 5.5 — Consequências da modelagem.

A partir disso, o algoritmo foi posto à prova nos componentes gerados, partindo da menor complexidade para a maior, obtendo soluções ótimas até o terceiro maior componente da Tabela 2. A solução deste componente foi obtida pelo algoritmo em cerca de 11 minutos.

Prosseguiu-se com a tentativa de solução do segundo maior componente. O programa não foi capaz de encontrar solução ótima em cerca de 12h de execução, após avaliar cerca de 500.000 arranjos. O fator de ramificação inicial deste grupo é na ordem de 300, tornando o problema excessivamente grande para a máquina disponível. Essa informação sugere que o uso do *beam search* pode contribuir para obtenção de alguma solução melhor que o algoritmo ingênuo.

6.3 Heurísticas

A técnica de *beam search* exige que seja feita uma classificação das alternativas de movimento. O recolhimento dispensa uma heurística, visto que nos casos em que pode ser recolhido a solução já é única.

Para as demais alternativas, remanejar e cortar, é necessária uma forma de limitar as movimentações mais favoráveis para prosseguir com a solução, descartando as demais.

6.3.1 Heurísticas de remanejamento

Para avaliar o remanejamento, o caso extremo em que não são permitidos cortes será estudado. Esta abordagem encontrará solução se em cada etapa em que um cruzamento for desfeito. Ao executar um remanejamento, o custo associado da remoção da aresta é dependente unicamente da distância dessa aresta até a extremidade.

Diferentemente do corte, o remanejamento não possui variante. Ele sempre opera no vértice divergente sobre o convergente. Outra diferença fundamental neste caso é que uma operação de remanejamento pode remover diversas arestas. Não obstante, não há comutatividade no processo de remanejamento: as movimentações de remanejamento possuem resultados distintos se executadas em ordem distinta.

Neste cenário de proibição de cortes, a solução ótima não necessariamente retira o conjunto mínimo de arcos de retorno, mas sim a menor movimentação que torna o grafo possuidor de uma ordem topológica. Para este cenário, foram avaliadas sete métricas para aplicação do *beam search*, e variou-se algumas larguras de busca para o mesmo método.

A primeira delas, mais simples, leva em conta apenas o comprimento movimentado, independente do número de cruzamentos removidos na operação. Outra heurística simples é o número total de cruzamentos retirados na movimentação.

A abordagem acima desconsidera uma característica do recolhimento: ela não é tão fortemente atrelada ao número de cruzamentos, simplesmente, e sim a quantidade de dutos que se tornam livres pelo movimento. Assim, a terceira heurística considera a quantidade de dutos distintos que o movimento libera.

A terceira heurística não considera nada relacionado à ordem desses cruzamentos. A Figura 11 compara dois arranjos com grafos representativos idênticos, mas com posicionamento distinto.

No primeiro desenho, os cruzamentos estão agrupados: todos os cruzamentos em que um duto está por cima estão juntos. No segundo desenho, os cruzamentos estão alternados. Enquanto no primeiro apenas uma movimentação retira a torção, no segundo são necessários cinco movimentos para remoção, em função da alternância entre os cruzamentos.

Quando os cruzamentos são agrupados, do ponto de vista de remanejamento, apenas o cruzamento mais distante da extremidade para um duto é relevante. Assim, surge a quarta heurística: medir quantas trocas de duto ocorrem no remanejamento.

Para as heurísticas baseadas em contagem, quando há um empate na quantidade, o comprimento movimentado foi o decisor para a ordenação da heurística.

As três heurísticas restantes para a solução global para o problema de zero cortes foram inspiradas na solução gulosa de Dantzig (1957) para o problema da mochila: usar a densidade de remoção de cruzamentos. A heurística número 5 divide o comprimento pelo número total de de cruzamentos. A sexta divide o comprimento pelo número de dutos distintos envolvidos nos cruzamentos retirados. Por fim, o última heurística avaliada é obtida dividindo o comprimentos pelo número de trocas de duto que ocorrem no remanejamento. Em resumo as heurísticas são:

(A) Mínimo comprimento

(B) Maior densidade de cruzamentos com dutos alternados

(C) Maior densidade de cruzamentos totais

(D) Maior densidade de cruzamentos com dutos distintos

(E) Maior quantidade de eventos totais

(F) Maior quantidade de dutos alternados

(G) Maior quantidade de dutos distintos

Inicialmente, foi obtida a solução ótima, sem cortes, para o terceiro maior componente conexo de Tainha. Este foi o maior conjunto de dutos em que foi possível obter a solução ótima. Em seguida, executou-se uma varredura nas sete heurísticas elencadas, cada uma variando a largura de feixe β de 1 — caso puramente guloso — até 5.

Na Tabela 4 são apresentadas os valores de solução encontrados e a quantidade de arranjos analisados durante a solução, por heurística e por largura de feixe permitida.

As heurísticas F e G retornaram resultados idênticos, possivelmente por, no caso do conjunto testado de dados, não apresentarem distinção na sua quantidade. Isso ocorreria, por exemplo, em um cenário em que todos os cruzamentos de um mesmo duto ocorrem em sequência.

Outro fato importante é que as heurísticas F e G retornam soluções melhores em relação às demais heurísticas, com a heurística E equivalendo no feixe de largura 5 à solução ótima, enquanto as demais heurísticas, baseadas em densidade e comprimento, não

atingem a solução ótima nesta largura, apesar da leve tendência a observar um número menor de arranjos.

Uma vez que as heurísticas F e G não se diferenciaram em termos de solução, vale a pena investigar se o tempo computacional para execução de cada uma das formas é significativamente distinto. A Tabela 5 compara os tempos de execução de cada heurística, em segundos, de acordo com a largura de feixe.

A heurística F foi pouco melhor com feixes menores, mas ela possui um crescimento muito maior que a heurística G, levando a ser desvantajosa em cerca de 15% no maior feixe testado (6).

A heurística E entregou um comportamento anômalo para o caso guloso, tendo um resultado absolutamente ruim. Esta heurística possui diversos empates, e, como a ordenação disponível na biblioteca padrão é estável, um movimento altamente desfavorável foi escolhido para avançar na solução. Então, por este comportamento, esta heurística não é adequada para estimativas grosseiras de solução.

É importante ressaltar que no cenário do terceiro maior componente conexo de Tainha, o máximo de movimentos possíveis em um arranjo é 18 (2×9), então um feixe de 6 representa um terço das movimentações possíveis. Em cenários maiores, utilizar um terço de todas as movimentações já é bastante proibitivo.

6.3.2 Heurística de corte

Em uma solução em que apenas cortes são permitidos, só pode-se garantir que os cruzamentos que não fazem parte de um grupo fortemente conexo podem ser retirados da lista de corte. Entretanto, o número de cruzamentos avaliados torna a busca excessivamente grande.

A pesquisa de corte visa identificar basicamente aqueles cruzamentos que pertencem ao conjunto mínimo de arcos de retorno. Este problema é vastamente explorado na literatura, e o trabalho de (SIMPSON; SRINIVASAN; THOMO, 2016) merece destaque por realizar a comparação de diversas heurísticas de obtenção desse conjunto mínimo.

No trabalho citado, foram comparados os desempenhos em 10 conjuntos de dados na obtenção destes conjuntos utilizando 11 algoritmos distintos. O menor dos grafos analisados possui 10617 vértices e 72172 arestas, claramente muito maior que o grafo de arranjo gerado pelo problema proposto.

Simpson, Srinivasan e Thomo (2016) coloca, que para os conjuntos de dados pequenos, o método GreedyFAS proposto por Eades, Lin e Smyth (1993) retornou o segundo menor conjunto de arcos de retorno, sendo o quarto algoritmo mais rápido na execução. Os três algoritmos mais rápidos que o GreedyFAS retornaram também os

maiores conjuntos de arcos, portanto piores, tornando-os pouco atrativos para a execução do algoritmo.

Assim, apenas a eficácia do GreedyFAS será avaliada, comparando o mesmo com a solução ótima global. Para este estudo retorna-se a resolver o terceiro maior conjunto fortemente conexo, utilizando os mesmos parâmetros, e a possibilidade de remanejamento. Para o remanejamento, todos os remanejamentos serão colocados na solução, sem limitar qualquer heurística.

Para o teste realizado, a pesquisa com o GreedyFAS retornou uma solução com o mesmo valor do ótimo global com a mesma quantidade de cortes e remanejamentos, mas os cortes ocorreram em dutos distintos da solução encontrada sem filtros. O resultado em si não representa uma garantia de que o método é sempre efetivo, mas o tempo de execução foi reduzido para 30,8 s, após avaliar 1138 arranjos.

A solução com cortes surpreendeu em tempo de execução, pois apesar de que haja um grande número de vizinhos, o fato de os cortes representarem uma operação associativa faz com que haja uma quantidade significativa deles que levam a arranjos já avaliados. Além disso, a operação de corte é muito mais barata que o remanejamento. Por conseguinte, na pesquisa, os cortes acabam sendo avaliados primeiro e a solução com cortes puros torna-se um limite superior satisfatório para a solução global.

Com este sucesso, tentou-se resolver o segundo maior componente fortemente conexo de Tainha — 29 dutos e 423 cruzamentos — com os mesmos parâmetros. Após 12h de execução, o algoritmo com heurísticas ainda não foi capaz de encontrar uma solução para este grupo considerando cortes e remanejamentos. Considerando apenas remanejamentos, o problema não possui solução, pois este conjunto um laço de sete tramos em que não há ponta solta. A tentativa apenas com cortes também não obteve solução em 12h de execução, apontando a limitação atual da ferramenta.

Tabela 4 – Comparação da performance entre as heurísticas e a largura de feixe.

Largura	Heurística	Custo	Nº de Arranjos
1	A	485.93	25
	B	479.10	24
	C	485.93	25
	D	479.10	24
	E	642.26	20
	F	339.85	16
	G	339.85	16
2	A	391.66	295
	B	391.66	245
	C	396.70	250
	D	391.66	275
	E	358.24	173
	F	333.02	193
	G	333.02	193
3	A	359.61	1420
	B	352.89	1224
	C	355.60	1269
	D	352.89	1316
	E	339.60	1151
	F	303.83	1184
	G	303.83	1184
4	A	338.22	4343
	B	320.85	3811
	C	320.85	3984
	D	320.85	3881
	E	322.22	4263
	F	297.00	4229
	G	297.00	4229
5	A	313.13	9377
	B	302.20	8410
	C	302.20	8773
	D	302.20	8446
	E	284.91	9764
	F	284.91	9517
	G	284.91	9517
Ótima		284.91	26317

Fonte: Autor (2020).

Tabela 5 – Tempo de execução das alternativas

Largura de feixe	Heurística F	Heurística G
1	0.051	0.054
2	0.437	0.418
3	2.831	2.547
4	10.70	11.26
5	38.68	31.80
6	78.16	67.69

Tempos de execução medidos em segundos. Fonte: Autor (2020).

7 Conclusão

Um modelo matemático baseado em teoria dos grafos para o problema do descomissionamento de dutos flexíveis submarinos foi concebido neste trabalho, e posteriormente testado em um caso real.

O objetivo principal deste trabalho foi conceber um algoritmo capaz de identificar a sequência operacional mais atrativa de realizar a retirada de uma massa de dutos do leito marinho, alcançado.

A simplificação da representação permitiu o uso de todo ferramental da teoria de grafos para encontrar estratégias e propriedades de um arranjo submarino qualquer. Além disso, com a discretização das ações do navio, foi possível criar um grafo ponderado que representa o espaço de solução do problema, permitindo o uso do algoritmo A*.

Apesar de que o estimador utilizado para o A* foi bastante simplório, o algoritmo estudado foi capaz de encontrar solução ótima para um conjunto de dutos com um tamanho razoável. Grupos maiores mostraram-se limitados em obter solução ótima, indicando o uso de heurísticas.

A fim de auxiliar na busca de soluções viáveis, foi proposta a utilização de *beam search*, enumerando sete possíveis heurísticas para o remanejamento e uma para a operação de corte. A heurística de remanejamento que obteve maior sucesso foi a de contagem de dutos distintos, em que consistentemente obteve soluções mais atrativas em qualquer nível de abertura de feixe, e possuiu um tempo de computação menor. Para a operação de corte a utilização de GreedyFAS para o corte também foi bem sucedida no sentido de prover um bom conjunto de arcos de retorno em um tempo curto.

Apesar das limitações, a solução computacional se mostra bastante promissora visto que a análise inicial do campo de Tainha, considerando apenas os dutos conectados na sua unidade teve dispêndio de cerca de 4 meses na sua execução manual. Cabe realizar o teste em uma máquina dedicada e conferir se esta execução automática é competitiva com a atual metodologia.

A sugestão para os trabalhos futuros é desenvolver melhores estimadores de solução para o problema como está posto, de forma a reduzir a quantidade de arranjos visitados, e incluir outros aspectos operacionais no modelo, como as capacidades do navio e da base, operações de arriamento de dutos, e utilização concomitante de mais de um recurso.

Referências

AHIAGA-DAGBUI, D. D. et al. Costing and Technological Challenges of Offshore Oil and Gas Decommissioning in the U.K. North Sea. *Journal of Construction Engineering and Management*, v. 143, n. 7, p. 05017008, jul. 2017. ISSN 0733-9364, 1943-7862. Disponível em: http://ascelibrary.org/doi/10.1061/(ASCE)CO.1943-7862.0001317 .

ARS, F.; RIOS, R. Decommissioning: A Call for a New Approach. In: *O shore Technology Conference*. Houston, Texas, USA: Offshore Technology Conference, 2017. Disponível em: http://www.onepetro.org/doi/10.4043/27717-MS .

BACCI, T.; MATTIA, S.; VENTURA, P. The bounded beam search algorithm for the block relocation problem. *Computers & Operations Research*, v. 103, p. 252–264, mar. 2019. ISSN 03050548. Disponível em: https://linkinghub.elsevier.com/retrieve/pii/S0305054818302934 .

BAI, Y.; BAI, Q. *Subsea Engineering Handbook.* St. Louis: Elsevier Science, 2014. OCLC: 1040038174. ISBN 9780123978059. Disponível em: http://qut.eblib.com.au/patron/FullRecord.aspx?p=842202 .

BECK, K. et al. Manifesto for agile software development. 2001.

BENNELL, J.; CABO, M.; MARTíNEZ-SYKORA, A. A beam search approach to solve the convex irregular bin packing problem with guillotine cuts. *European Journal of Operational Research*, v. 270, n. 1, p. 89–102, out. 2018. ISSN 03772217. Disponível em: https://linkinghub.elsevier.com/retrieve/pii/S0377221718302571 .

BLUM, C.; BLESA, M. J. Probabilistic beam search for the longest common subsequence problem. In: *Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing E ective Heuristics*. Springer Berlin Heidelberg, 2009. p. 150–161. Disponível em: https://doi.org/10.1007/978-3-540-74446-7_11 .

BONDY, J. A.; MURTY, U. S. R. *Graph theory*. New York: Springer, 2010. OCLC: 780644570. ISBN 9781849966900.

BRASIL. Constituição (1967). *Constituição da República Federativa do Brasil*. Brasília, DF: Senado, 1967.

BRASIL. Constituição (1988). *Artigo 225*. Brasília, DF: Senado, 1988.

BRASIL, A. *Resolução ANP Nº 817 DE 24/04/2020*. 2020.

BRESSAN, R.; ARTIGAS, D. Modelagem por grafos do recolhimento de dutos flexíveis submarinos para projetos de descomissionamento. In: *Anais eletrônicos...* Limeira/SP: Galoá, 2019. (107720, v. 2). Disponível em: https://proceedings.science/sbpo-2019/papers/modelagem-por-grafos-do-recolhimento-de-dutos-flexiveis-submarinos-para-projetos-de-descomissiona

DANTZIG, G. B. Discrete-variable extremum problems. *Operations Research*, Institute for Operations Research and the Management Sciences (INFORMS), v. 5, n. 2, p. 266–288, abr. 1957. Disponível em: https://doi.org/10.1287/opre.5.2.266 .

DASGUPTA, S. *Algorithms*. Boston: McGraw-Hill Higher Education, 2008. ISBN 9780073523408.

DIJKSTRA, E. W. A note on two problems in connexion with graphs. *Numerische Mathematik*, Springer Science and Business Media LLC, v. 1, n. 1, p. 269–271, dez. 1959. Disponível em: https://doi.org/10.1007/bf01386390 .

DJUKANOVIC, M.; RAIDL, G. R.; BLUM, C. Exact and Heuristic Approaches for the Longest Common Palindromic Subsequence Problem. In: BATTITI, R. et al. (Ed.). *Learning and Intelligent Optimization*. Cham: Springer International Publishing, 2019. v. 11353, p. 199–214. ISBN 9783030053475 9783030053482. Disponível em: http://link.springer.com/10.1007/978-3-030-05348-2_18 .

EADES, P.; LIN, X.; SMYTH, W. A fast and effective heuristic for the feedback arc set problem. *Information Processing Letters*, Elsevier BV, v. 47, n. 6, p. 319–323, out. 1993. Disponível em: https://doi.org/10.1016/0020-0190(93)90079-o .

FEOFILOFF, P.; KOHAYAKAWA, Y.; WAKABAYASHI, Y. *Uma Introdução Sucinta à Teoria dos Grafos*. São Paulo: Universidade de São Paulo, 2011. Disponível em: https://www.ime.usp.br/ yw/publications/books/TeoriaDosGrafos.pdf .

GASPAR, F.; BILDA, K. Field Decommissioning Obligations in Brazil: The Differences and Impacts of Current Methodologies for Concessions and Sharing Regimes in Light of Investment Attraction. In: *OTC Brasil*. Rio de Janeiro, Brazil: Offshore Technology Conference, 2017. Disponível em: http://www.onepetro.org/doi/10.4043/27994-MS .

HART, P.; NILSSON, N.; RAPHAEL, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, Institute of Electrical and Electronics Engineers (IEEE), v. 4, n. 2, p. 100–107, 1968. Disponível em: https://doi.org/10.1109/tssc.1968.300136 .

HART, P. E.; NILSSON, N. J.; RAPHAEL, B. ucorrection/u to "a formal basis for the heuristic determination of minimum cost paths". *ACM SIGART Bulletin*, Association for Computing Machinery (ACM), n. 37, p. 28–29, dez. 1972. Disponível em: https://doi.org/10.1145/1056777.1056779 .

KAHN, A. B. Topological sorting of large networks. *Communications of the ACM*, Association for Computing Machinery (ACM), v. 5, n. 11, p. 558–562, nov. 1962. Disponível em: https://doi.org/10.1145/368996.369025 .

KARP, R. M. Reducibility among combinatorial problems. In: *Complexity of Computer Computations*. Springer US, 1972. p. 85–103. Disponível em: https://doi.org/10.1007/978-1-4684-2001-2_9 .

KHEMANI, C. et al. Solving Rubik's Cube Using Graph Theory. In: VERMA, N. K.; GHOSH, A. K. (Ed.). *Computational Intelligence: Theories, Applications and Future Directions - Volume I*. Singapore: Springer Singapore, 2019. v. 798, p. 301–317. ISBN 9789811311314 9789811311321. Disponível em: http://link.springer.com/10.1007/978-981-13-1132-1_24 .

MACEDO, M. M. B. d. *Descomissionamento de Instalações Marítimas: Perspectivas para o Brasil*. Macaé: [s.n.], 2018. Disponível em: http://www.anp.gov.br/images/Palestras/redepetro_mafra_12_07_2018.pdf .

MASON, P. *Evolution of subsea well system technology*. Endeavor Business Media, 2006. Disponível em: https://www.offshore-mag.com/subsea/article/16754406/evolution-of-subsea-well-system-technology .

MELO, G. S. d. *Introdução à Teoria dos Grafos*. Tese (Dissertação) — Universidade Federal da Paraíba, João Pessoa (PB), jan. 2014. Disponível em: https://repositorio.ufpb.br/jspui/bitstream/tede/7549/5/arquivototal.pdf .

NUNES, F.; FAVARO, C. *Plataformas da Petrobrás começam a se tornar sucata - Economia*. Estadão, 2019. Disponível em: https://economia.estadao.com.br/noticias/geral,plataformas-da-petrobras-comecam-a-se-tornar-sucata,70003010750 .

OLIVEIRA, E. V. A. d. et al. *NOTA TÉCNICA*. Rio de Janeiro, 2015. 34 p. Disponível em: http://www.anp.gov.br/images/Consultas_publicas/Concluidas/2015/n12/Nota_Tecnica_no.132SSM2015.doc .

PEFFERS, K. et al. A design science research methodology for information systems research. *Journal of Management Information Systems*, Informa UK Limited, v. 24, n. 3, p. 45–77, dez. 2007. Disponível em: https://doi.org/10.2753/mis0742-1222240302 .

PETROBRAS. *Projeto de Descomissionamento da Unidade Estacionária de Produção P-12 nos Campos de Linguado, Badejo , Trilha e Bicudo Bacia de Campos*. [S.l.], 2015. 31 p. Disponível em: http://licenciamento.ibama.gov.br/Petroleo/Producao/Producao%20-%20Bacia%20de%20Campos%20-%20Termo%20de%20Ajustamento%20de%20Conduta%20-%20Petrorbas/P-12/PD_P12%20Anu%C3%AAncia%20Ibama%20Vers%C3%A3o%2018.pdf .

PRADO, D. D. *Desmobilização de dutos em sistemas marítimos de produção de petróleo - Uma proposta de método de suporte ao planejamento*. Tese (Dissertação) — Universidade Federal do Rio de Janeiro, Rio de Janeiro, abr. 2015. Disponível em: http://www.ppe.ufrj.br/images/publica\unhbox\voidb@x\setbox\z@\hbox{c}\accent24c oes/mestrado/Diogo_Diederichs_Prado.pdf .

ROUSE, S. et al. Offshore pipeline decommissioning: Scale and context. *Marine Pollution Bulletin*, v. 129, n. 1, p. 241–244, abr. 2018. ISSN 0025326X. Disponível em: https://linkinghub.elsevier.com/retrieve/pii/S0025326X18301292 .

SAPURA. *Nossa Frota — Sapura*. 2020. Disponível em: http://www.sapura.com.br/nossa-frota/ .

SEIN et al. Action design research. *MIS Quarterly*, JSTOR, v. 35, n. 1, p. 37, 2011. Disponível em: https://doi.org/10.2307/23043488 .

SHARIR, M. A strong-connectivity algorithm and its applications in data flow analysis. *Computers & Mathematics with Applications*, Elsevier BV, v. 7, n. 1, p. 67–72, 1981. Disponível em: https://doi.org/10.1016/0898-1221(81)90008-0 .

SIMPSON, M.; SRINIVASAN, V.; THOMO, A. Efficient computation of feedback arc set at web-scale. *Proceedings of the VLDB Endowment*, VLDB Endowment, v. 10, n. 3, p. 133–144, 2016.

SULAIMAN, M. Z.; ROS, M. I. M.; AZIZAN, M. A. Successful of SK305 Artificial Reefing Decommissioning Project. In: *SPE Symposium: Decommissioning and Abandonment*. Kuala Lumpur, Malaysia: Society of Petroleum Engineers, 2018. Disponível em: http://www.onepetro.org/doi/10.2118/193990-MS .

TANAKA, S.; TAKII, K. A Faster Branch-and-Bound Algorithm for the Block Relocation Problem. *IEEE Transactions on Automation Science and Engineering*, v. 13, n. 1, p. 181–190, jan. 2016. ISSN 1545-5955, 1558-3783. Disponível em: http://ieeexplore.ieee.org/document/7123687/ .

TARJAN, R. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, Society for Industrial & Applied Mathematics (SIAM), v. 1, n. 2, p. 146–160, jun. 1972. Disponível em: https://doi.org/10.1137/0201010 .

VALEUR, J. R. Environmental Impacts of NORM Disposal–With Emphasis on Discharges to Sea. *SPE Projects, Facilities & Construction*, v. 6, n. 03, p. 124–131, set. 2011. ISSN 1942-2431. Disponível em: http://www.onepetro.org/doi/10.2118/136312-PA .

WU, K.-C.; TING, C.-J. A beam search algorithm for minimizing reshuffle operations at container yards. *Proceedings of the International Conference on Logistics and Maritime Systems*, 01 2010.