UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE ESTATÍSTICA E INVESTIGAÇÃO OPERACIONAL

Determinação de rotas para um navio de investigação utilizado em campanhas de pesca

Katlene Aline Fortes Brito

Mestrado em Estatística e Investigação Operacional
Especialização em Investigação Operacional

Dissertação orientada por:
Prof.ª Doutora Ana Maria Duarte Silva Alves Paias
Prof.ª Doutora Marta Guerreiro Duarte Mesquita de Oliveira

2018

Agradecimentos

Agradeço primeiramente a Deus, porque sem ele não consegueria chegar até aqui e também agradeço pela família que me deu que sempre me apoiou e sempre esteve presente comigo em todas as etapas.

Um agradecimento especial a todos os meus professores que fizeram parte do meu percurso académico, principalmente às minhas orientadoras a Professora Doutora Ana Maria Duarte Silva Alves Paias e a Professora Doutora Marta Guerreiro Duarte Mesquita de Oliveira pelo apoio incondicional e por sempre estarem disponíveis para ajudar-me quando precisei. Não há palavras para descrever a minha gratidão por tudo que fizeram por mim.

Por último mas nem por isso menos importante, um agradecimento muito especial ao meu noivo pelo suporte ao longo desta longa caminhada e por encorajar-me quando me sentia desmotivada.

Esta dissertação foi financiada por fundos nacionais da FCT - Fundacão para a Ciência e a Tecnologia, ao abrigo do projecto: PTDC/MAT-NAN/2196/2014”

Resumo

O problema de determinação de rotas para navios de investigação para campanhas de pesca, designado como *Ship Routing Optimization Problem* (SROP), é um problema que até agora não foi muito estudado e explorado.

No problema são conhecidos o conjunto de estações de pesca a ser visitadas, o número de circuitos a determinar e o número de dias que no máximo deve durar cada circuito. Também são conhecidas as janelas temporais associadas às visitas às estações que determinam as horas de visita para cada dia e as janelas temporais associadas às visitas aos portos, que garantem que estes sejam visitados com a periodicidade previamente definida. Cada circuito deve começar e terminar no porto origem. Dadas as localizações geográficas das estações de pesca e dos portos, o objetivo do problema é minimizar a distância total percorrida e a duração total de cada circuito, garantindo que todas as estações são visitadas e respeitando as restrições das janelas temporais.

Nesta dissertação, apresentam-se três modelos matemáticos em programação linear inteira mista (PLIM) para resolver o problema. Para além dos modelos, sugerem-se também heurísticas que têm como objetivo obter soluções inteiras admissíveis utilizando os modelos propostos e um *solver* genérico.

Pretende-se comparar as três formulações a nível da qualidade dos limites inferiores para o valor ótimo, fornecidos por relaxações lineares totais e parciais e do tempo computacional gasto para obter os mesmos. Através das heurísticas pretende-se obter soluções admissíveis para todas as instâncias e os limites superiores para o valor ótimo associados aos respetivos valores. As instâncias utilizadas são baseadas em campanhas realizadas pelo IPMA (Instituto Português do Mar e da Atmosfera) para estimar a abundância e observar a distribuição geográfica de várias espécies marinhas da costa portuguesa e instâncias disponibilizadas na literatura para o Problema do Caixeiro Viajante com Seleção de Hotéis.

Os resultados obtidos confirmam a complexidade do problema e a dificuldade em resolve-lo usando um *solver* genérico, tendo existido instâncias para as quais não foi possível encontrar uma solução admissível.

Palavras-chave: Otimização de rotas de navios, Programação Linear Inteira Mista (PLIM), Heurísticas.

Abstract

The problem of determining routes for vessels in fisheries research, called the Ship Routing Optimization Problem (SROP), has not been much studied and explored.

The set of fishing stations to be visited, the number of circuits to establish and the maximum number of days that each circuit should last are known, as well as the time windows associated with station visits which determine the visiting hours for each day and the windows associated with visits to the seaports assuring a predefined periodicity. Each circuit must start and end at the home seaport. Given the geographical locations of the fishing stations and seaports, the objective is to minimize the total traveled distance and the total duration of each circuit, ensuring that all stations are visited and time windows constraints are satisfied.

In this dissertation, three mixed integer linear programming mathematical models are presented for the problem. Furthermore, heuristics that aim to obtain feasible integer solutions using the proposed models and a generic solver are suggested .

It is intended to compare the three models in what concerns the quality of the lower bounds for the optimal value provided by partial or total linear programming relaxations and the computational time spent to obtain them. By using the heuristics, it is intended to obtain feasible solutions that provide upper bounds for all instances. The instances used are based in the campaigns realized by IPMA (Portuguese Sea and Atmosphere Institute) to estimate the abundance and observe the geographical distribution of several demersal fish species from the Portuguese continental coast and instances available in literature for the Traveling Salesman Problem with Hotel Selection.

The results obtained confirm the complexity of the problem and the difficulty to solve it using a generic solver. There have been instances for which it was not possible to obtain a feasible solution.

Keywords: Ship Routing Optimization Problem, Mixed Integer Linear Programming (PLIM), Heuristics.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Introdução

Atualmente, mais do que nunca, a poluição tem alterado o habitat das espécies marítimas e causado modificações climáticas, que por sua vez têm interferido na abundância e localização geográfica das mesmas. Devido a isto, tem sido muito importante fazer estudos para analisar quais as mudanças que se têm verificado a cada ano, principalmente em relação às espécies que são comercializadas para consumo.

As campanhas de monitorização previstas no Plano Nacional de Amostragem Biológica, têm-se revelado muito importantes, uma vez que a recolha das amostras tem dado informações como a quantidade de espécies de peixes existentes em determinadas áreas num certo período, assim como outros aspetos referidos em [9].

Para realizar estas campanhas, existem muitos custos envolvidos, portanto convém ter conhecimento acerca das estações de pesca e dos portos que devem ser visitados. Para além disso, também é preciso definir uma rota (a melhor possível) para realizar estas visitas, para que a nível logístico seja mais fácil organizar todo o processo. Sendo que, a melhor rota depende do que cada instituição/empresa estabelece como objetivo.

Como referido acima, a rota é pré-determinada com início e fim num mesmo porto, assegurando que todos os lugares pretendidos sejam visitados e dentro dos prazos definidos. Pois, assim como em todas as áreas de trabalho, os investigadores têm um horário de trabalho o que implica que os estudos feitos em cada dia têm que respeitar esse horário. Então estabelecem-se janelas temporais para as estações de pesca que respeitem estas horas de trabalho.

Nesta dissertação, pretende-se estudar o problema de determinação de rotas para navios de investigação utilizados nestas campanhas de bio-oceanografia e pesca, *Ship Route Optimization Problem* (SROP), considerando que cada estação é visitada num intervalo de tempo pré-estabelecido, a janela temporal. Em relação aos portos, as janelas determinam os dias em que estes devem ser visitados. Neste problema o número de circuitos $(rotas)$ a determinar é previamente estabelecido, assim como as janelas temporais para as visitas às estações de pesca e aos portos e o número máximo de dias que deve durar cada circuito. É imprescindível garantir que todas as estações de pescas sejam visitadas exatamente uma vez, ou seja, cada estação só pode estar num dos circuitos. Relativamente aos portos, pode ser necessário visitar um porto a meio de um circuito, de acordo com uma periodicidade que também é pré-estabelecida. As razões que levaram à imposição da restrição que um navio deve visitar um porto depois de um certo período viajando são, por exemplo: o reabastecimento de comida e combustível, a eventual mudança de tripulação ou até mesmo de investigadores.

Neste problema, as janelas temporais acrescentam mais dificuldade e, podem fazer com que a rota seja mais demorada em termos de tempo e distância. Por exemplo, pode acontecer que quando o navio estiver saindo de uma estação para ir para a mais próxima, se percebe que não será possível executar a operação de pesca satisfazendo as janelas temporais pré-definidas, então poderá ser melhor optar por outra estação que não esteja tão próxima, aumentando assim a distância percorrida e o tempo de viagem de um lugar para outro, mas minimizando o tempo de espera o que poderá ser traduzido numa menor duração final para a rota. No caso desta dissertação, o objetivo será minimizar o tempo total de viagem e a duração total dos circuitos, respeitando todas as restrições impostas.

Este problema tem semelhanças com algumas variantes dum problema muito estudado e citado na literatura, o Problema do Caixeiro Viajante (*Travelling Salesman Problem* $- TSP$). Na secção da revisão bibliográfica evidenciam-se as semelhanças e as diferenças existentes do problema em estudo com o Problema de Roteamento de Veículos (*Vehicle Routing Problem* $- VRP$), o Problema do Caixeiro Viajante com Seleção de Hotéis (*Travelling salesman Problem with Hotel Selection - TSPHS*).

Na literatura têm sido sugeridos modelos matemáticos, algoritmos exatos e métodos heurísticos para a resolução deste tipo de problemas. Alguns têm se revelado mais eficientes do que outros em termos de obter soluções de melhor qualidade, no entanto há que levar em conta alguns fatores, como o tempo computacional gasto na obtenção dessas soluções. Uma solução pode ser muito boa mas se demora muitos dias para ser conseguida já não é tão vantajosa pois, as empresas têm uma certa urgência em obter resultados.

Qualquer que seja o objetivo estabelecido, pretende-se obter a melhor solução no menor tempo possível. Na realidade quando o problema em si é muito difícil ou quando os dados com que se trabalha são de grande dimensão, torna-se mais complicado conseguir resultados completamente satisfatórios. Sendo assim, convém estipular prioridades, isto é, decidir se será mais importante uma boa solução independentemente do tempo que se leva até obtê-la ou uma solução não tão boa obtida num tempo razoável.

As observações feitas acima, incentivaram à orientação deste trabalho, cujo o objetivo é propor três modelos formulados em programação linear inteira mista ($PLIM$), comparar e analisar os resultados obtidos através deles e, ainda sugerir uma heurística que permita obter soluções admissíveis num tempo razoável. Para implementar e testar todos os modelos, utilizou-se o *software* IBM ILOG CPLEX *Optimization Studio* 12.7.0.

A dissertação encontra-se dividida em 6 capítulos. No segundo capítulo faz-se referência aos artigos e documentos consultados para desenvolver este trabalho, bem como a relação entre o problema estudado e alguns outros problemas de otimização de rotas. No capítulo 3 são apresentados os modelos matemáticos, para além dos métodos utilizados para obter limites inferiores. No capítulo 4, apresenta-se uma heurística como forma de obter limites superiores para o valor ótimo do problema. No capítulo 5, são apresentados os resultados de todos os testes realizados. No capítulo 6 apresentam-se as conclusões acerca de todo o trabalho elaborado, assim como algumas sugestões para trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

O problema em estudo, determinação de rotas para navios de investigação para campanhas de pesca, designado como *Ship Route Optimization Problem* (SROP), pode ser definido da seguinte forma: conhecendo a localização de um conjunto de estações de pesca que necessitam ser visitadas, dentro de uma janela temporal admissível, pretende-se determinar K circuitos, em que K é fixo, que comecem e terminem no mesmo porto de modo que a distância total percorrida e a duração total dos circuitos seja o menor possível.

O problema pode ser considerado um problema bicritério devido ao fato de ter dois objetivos. Estes objetivos são conflituosos entre si, uma vez que ao minimizar a distância total não se consegue garantir que a duração dos circuitos é a mínima possível, e vice-versa.

Este capítulo, divide-se em duas secções. A primeira secção apresenta alguns modelos e métodos propostos na literatura para resolver diferentes tipos de problemas de determinação de rotas para navios; e na segunda mencionam-se alguns problemas relacionados com o SROP assim como, as semelhanças e diferenças existentes entre os mesmos.

2.1 Problemas de roteamento de navios na literatura

Os problemas de desenhos de rotas têm sido bastante estudados, devido às suas aplicações em problemas reais. Para o caso de rotas para navios pode-se citar alguns tipos de problemas: o *Ship Routing and Scheduling Problem with Time Window* (SRSP), o *Ship Route Optimization Problem* (SROP) e o *Ship Routing Problem*. Existem também outros tipos de problemas, em que a principal preocupação já não é só relativamente às rotas mas também, às questões ambientais.

Em 2010, Fagerholt *et al.* [8] estudaram o problema de roteamento de navios, com vista a diminuir o consumo de combustível. Explicam que normalmente o consumo de combustível é uma função cúbica da velocidade, tornando o problema em um problema não-linear. O modelo que sugerem é composto por uma função objetivo que é convexa e não-linear, e por algumas restrições que são também não-lineares. Devido a isso, propõem uma forma de resolver o problema que é discretizando os tempos das janelas temporais de cada nodo e depois resolvendo um problema de caminho mais curto num grafo acíclico, isto é, num grafo onde não há caminhos que começam e terminam no mesmo nodo. Fizeram uma comparação entre os resultados obtidos com o método proposto e com os resultados obtidos através de um solver de programação não-linear e, constataram que o método fornecia melhores soluções (quase ótimas) e era mais rápido.

Acerca do *ship routing and scheduling problem with time window*, em 2012 Al-Hamad *et al.* [3] estudaram uma variante que englobava restrições para os tempos de entrega (impostos pelos clientes), e, para a capacidade do navio. Sugerem-se dois métodos para resolver o problema: um algoritmo genético e um método exato que envolve o problema de particionamento de conjuntos (*Set Partitioning Problem - SPP*). Os resultados obtidos revelam que para problemas não muito grandes o SPP consegue soluções muito próximas do ótimo, enquanto que para problemas de maior dimensão o algoritmo genético é mais aconselhável. Também se faz referência ao fato do algoritmo genético produzir melhores resultados do que um outro método que utiliza o *Tabu search* [2].

Em 2013, Romero *et al.* [13] referem que os problemas de roteamento de navios (*ship routing problems*) não tiveram a mesma atenção que os problemas de roteamento de veículos, evidenciando que existem algumas características que têm que ser considerados no *ship routing* e que normalmente não se encontram nos *vehicles routing problems* mas, aponta também as semelhanças existentes entre os mesmos. Para resolver o problema de rotas para navios propõe-se um algoritmo GRASP (*Greedy Randomized Adaptive Search Procedure*), que é uma meta-heurística aplicada a problemas combinatórios, onde cada iteração é constituída por duas fases, a construção e a pesquisa local. Os resultados obtidos foram satisfatórios, uma vez que o algoritmo proposto tinha em consideração situações de entregas parciais e prioridades entre os clientes que são restrições que acrescentam mais dificuldade ao problema.

O SROP (*ship route optimization problem*), que é o problema estudado nesta dissertação, além de ser um tipo de problema de roteamento de navios, pode ser comparado a algumas variantes do problema do caixeiro viajante (*Travelling salesman problem*). É um problema que pode ter aplicações na área biológica (estudos de espécies marítimas), na indústria (por exemplo, distribuição de produtos). Em 2017, Mesquita *et al.* [11] apresentaram uma formulação em PLIM para o problema SROP, a qual serviu de base a uma das formulações apresentadas no capítulo seguinte desta dissertação. Constatou-se que a utilização do modelo para resolver problemas reais utilizando um *solver*, pré-estabelecendo o número de rotas, não seria muito eficiente devido ao elevado tempo computacional consumido. Devido a isso foram propostos em [11] duas heurísticas sequenciais, Seq e SeqPlus, que combinam algoritmos genéticos e pesquisas em vizinhanças. Isto é, para melhorar a qualidade das soluções obtidas pelo algoritmo genético aplicou-se um procedimento ALNS (*Adaptive Large Neighborhood Search*) tendo sido possível diminuir a distância total percorrida, num curto tempo computacional. No caso de problemas reais, ambas as heurísticas produziram boas soluções num curto tempo computacional.

2.2 Relação entre o SROP e alguns problemas de otimização de rotas

Como foi referido anteriormente, o SROP pode ser comparado a algumas variantes do TSP como por exemplo: o problema do caixeiro viajante com seleção de hotéis (TSPHS), o problema do caixeiro viajante com múltiplas janelas temporais e com seleção de hotéis (TSP-MTWHS), o problema do caixeiro viajante com múltiplas janelas temporais e com cidades seletivas (TSPSTW) e o problema do caixeiro viajante múltiplo (MTSP). Mas também pode ser relacionados a outros problemas como o problema de roteamento de veículos (VRP), o problema de orientação com seleção de hotéis (OPHS).

O TSPHS foi introduzido em 2012 por Vansteenwegen *et al.* [17], e consiste em determinar uma rota que tenha início e fim num mesmo hotel (depósito) e que visite todos os clientes, minimizando o número de percursos e a distância total percorrida. Sabendo que não se consegue visitar todos os clientes no mesmo dia devido ao número máximo de horas diárias de trabalho, é preciso escolher um hotel para passar a noite. Assim, a rota é constituída por percursos diários onde cada um deles inicia e termina num hotel. É de referir também que cada hotel pode ser utilizado mais do que uma vez. No SROP, os portos podem ser vistos como os hotéis do TSPHS e as estações de pesca como os clientes. Uma diferença que há entre estes problemas é que no SROP o navio pode fazer viagens durante a noite mas no TSPHS não acontecem movimentações à noite.

O TSP-MTWHS [4] é um caso do TSPHS onde os clientes são visitados em diferentes janelas temporais, cujo objetivo é minimizar os custos da rota, nomeadamente custos dos hotéis, das viagens, de horas extras e possíveis penalizações por não conseguir visitar um cliente. As diferenças e as semelhanças entre o TSP-MTWHS e o SROP são as mesmas em relação ao TSPHS, podendo acrescentar-se ainda que os dois problemas possuem a mesma dificuldade relativamente à combinação da seleção de hotéis ou portos com a minimização da distância percorrida ou duração dos circuitos.

Uma outra variante do TSP a que se pode comparar o SROP, é o TSPSTW [10]. No TSPSTW considera-se que existem dois conjuntos de cidades, as cidades obrigatórias e as seletivas, em que todas as obrigatórias devem ser visitadas uma vez dentro de uma das múltiplas janelas pré-definidas. Acerca das seletivas, apenas um subconjunto (cuja cardinalidade depende da duração da rota) delas deve ser visitada dentro de uma das múltiplas janelas associadas. Tanto o TSPSTW como o SROP têm como objetivo minimizar a distância total percorrida e a duração das rotas. As cidades obrigatórias e as seletivas podem ser vistas como as estações de pesca e os portos no SROP, respetivamente.

Na variante MTSP [19], considera-se o caso de haver m caixeiros viajantes. O objetivo é minimizar a distância percorrida por todos eles, tendo em conta que cada caixeiro viajante pode visitar no máximo Q cidades. No MSTP as cidades podem ser equiparadas às estações de pesca no SROP, assim como em alguns dos problemas acima mencionados. A principal diferença do MSTP em relação ao SROP, é a restrição imposta acerca do número de cidades visitadas por cada caixeiro viajante, pois o navio não tem um limite máximo de estações que pode visitar.

No VRP [15] há um certo número de veículos disponíveis, sendo que cada um deles tem uma capacidade conhecida e que não pode ser excedida. O objetivo é determinar uma rota para cada veículo de forma a minimizar a distância total percorrida. Como foi referido anteriormente Romero *et al.* [13] especificou algumas diferenças e semelhanças entre estes tipos de problemas, o VRP e os problemas de rotas para navios. Podem-se citar algumas diferenças: as viagens durante a noite, o elevado grau

de incerteza devido às condições meteorológicas que podem levar a mudanças de rotas, o fato de não existirem estradas ou caminhos no meio marítimo, entre outras. No entanto, em ambos os problemas podem encontrar-se aspetos como as janelas temporais para os clientes e a necessidade de visitar todos eles dentro das janelas definidas.

O OPHS, introduzido por Divsalar *et al.*[6], é um problema onde são dados um certo número de locais, cada um com uma pontuação associada. No fim pretende-se obter uma rota, composta por diferentes percursos diários, que maximize a pontuação obtida e que respeite o limite diário de tempo imposto a cada percurso. Em [7] foi proposto, um algoritmo memético para resolver o problema. Acerca das diferenças entre o OPHS e o SROP, a principal a ser mencionada é que no OPHS não é obrigatório visitar todos os locais. Se os percursos do OPHS forem comparados às rotas do SROP que tenham início e fim num porto, podendo os portos não coincidir, uma outra diferença é que o número de percursos no OPHS é pré-definido e no SROP não, porque depende da necessidade de visitar um porto após o período definido ou não. Por exemplo se uma das rotas pode ser feita em menos de 7 dias não é necessário visitar um porto intermédio.

Capítulo 3

Modelos Matemáticos

No presente capítulo define-se o problema de otimização de rotas para navios de investigação utilizados em campanhas de pesca e apresentam-se três modelos matemáticos para o mesmo. São feitas algumas considerações importantes acerca dos modelos e do problema. A última secção é dedicada à obtenção de limites inferiores.

3.1 Definição do Problema

O problema de otimização de rotas para navios pode ser definido num grafo *G=(V,A)*, em que *V= C* $\cup$ *D* é o conjunto de todos os vértices, onde cada vértice *i* $\in$ *C* = $\{1, ..., n\}$ corresponde a uma estação de pesca, e cada vértice *i* $\in$ *D* = $\{1, ..., d\}$ corresponde a um porto, sendo que os portos *d-1* e *d* correspondem ao porto de partida e de chegada, respetivamente. O conjunto *A* é o conjunto de todos os arcos possíveis entre estações de pesca e entre as estações e os portos. Para cada circuito k o conjunto A^k inclui os arcos entre estações, $\{(i,j) : (i,j) \in C \times C, i \neq j\}$, os arcos entre as estações e os portos e vice-versa, $\{(i,j) : (i,j) \in C \times D\}$ e $\{(i,j) : (i,j) \in D \times C\}$ que pertencem ao circuito.

Cada circuito deve começar no instante 0 e não pode ultrapassar os 14 dias. Define-se o parâmetro $b = 14 \times 24$ horas, que representa a duração máxima de cada circuito. Em cada circuito, k=1,...,K, define-se para cada dia *h*=1,...,$|H|$, com $|H| = 14$, uma janela temporal $\left[e^h, l^h\right]$ que delimita a hora de visita a uma estação. Sabendo que o serviço nas estações só pode iniciar às 7 da manhã, sendo a última entrada às 18h, as janelas serão $[7, 18]$,$[31, 42]$,$[55, 66]$,...,$[319, 336]$ respetivamente para os dias 1,2,3,..,14.

Uma das imposições do problema é que após 7 dias consecutivos em viagem o navio deve visitar um porto definindo assim a periodicidade *p*= 7×24 $(horas)$. Caso um porto seja visitado tem que ser dentro da janela $\left[e^{|H|+1}, l^{|H|+1}\right]$, que corresponde a $[160, 168]$.

A cada arco $(i,j) \in A$ associa-se um tempo de viagem t_{ij} que indica o tempo gasto pelo navio para se deslocar do vértice *i* para o vértice *j*. E ainda, para todos os vértices *i* $\in$ C $\cup$ D denota-se por s_i o tempo de serviço associado à duração da visita/operação ao mesmo.

O objetivo consiste em determinar um conjunto de K rotas, com início e fim no mesmo porto, que minimize a soma da distância total percorrida com a duração total dos circuitos respeitando as janelas temporais para estações e portos. É necessário assegurar que cada estação é visitada uma única vez por uma das rotas. Além disso, dependendo da duração de cada rota deve ou não ocorrer uma visita a um porto.

3.2 Formulações

Os três modelos propostos foram formulados em Programação Linear Inteira Mista (PLIM) com o objetivo de minimizar a soma da distância total com a duração total dos circuitos. As formulações apresentadas foram baseadas nas formulações propostas em Mesquita *et al.* [11] e Mesquita *et al.* [12]. O primeiro modelo é um modelo com janelas temporais e com restrições semelhantes às de Miller-Tucker-Zemlin (MTZ) [14], daí ser designado nesta dissertação como Modelo SR-MTZ (*Ship Routing*-MTZ). Os outros dois modelos propostos são modelos de fluxos com janelas temporais e portanto, designou-se o segundo modelo como Modelo SR-MF (*Ship Routing*-MultiFluxo) e o terceiro como SR-MFA (*Ship Routing*-MultiFluxo Agregado) .

Nas formulações são consideradas três tipos de variáveis: variáveis de tempo, de circuito e de rota. As variáveis de rota e de circuito são comuns a todos os modelos, no entanto consoante o modelo as variáveis que definem o tempo de serviço nos portos e nas estações são definidas de formas diferentes.

Designam-se por variáveis de rota as variáveis x_{ij}^k, $(i,j) \in A^k$, k=1,...K, que tomam o valor 1 se o navio viaja do vértice *i* diretamente para o vértice *j* no circuito *k*, e 0 caso contrário. Para as variáveis de circuito tem-se y_i^k = 1, i $\in$ C $\cup$ $\{d-1, d\}$, k=1,...K, se o vértice *i* é visitado no circuito *k*, e 0 caso contrário. As variáveis de tempo são designadas por δ_i^h, i $\in$ C, h=1,...,$|H|$, e tomam valor 1 se a estação de pesca *i* $\in$ C é visitada durante a janela *h*, e valor 0 caso contrário.

3.2.1 Modelo SR-MTZ

Na formulação do modelo SR-MTZ, definem-se as variáveis u_i, que indicam o início da operação na estação *i* $\in$ C, e w_i^k, que indicam o início da operação no porto *i* $\in$ D no circuito k, k=1,...,K. Note-se que, enquanto cada estação vai estar num único circuito, um porto pode ser visitado por diferentes circuitos. Denota-se por M uma constante suficientemente grande. O modelo SR-MTZ é definido por:

$$minimizar \sum_{k=1}^{K} \sum_{(i,j)\in A^k} t_{ij} x_{ij}^k + \sum_{k=1}^{K} w_d^k \tag{3.1}$$

Sujeito a:

$$\sum_{k=1}^{K} y_j^k = 1, \quad j \in C \tag{3.2}$$

$$\sum_{k=1}^{K} y_{d-1}^k = K \tag{3.3}$$

$$\sum_{k=1}^{K} y_d^k = K \tag{3.4}$$

$$\sum_{i:(i,j)\in A^k} x_{ij}^k = y_j^k, \quad j \in C \cup \{d\}, k = 1, ..., K \tag{3.5}$$

$$\sum_{j:(i,j)\in A^k} x_{ij}^k = y_i^k, \quad i \in C \cup \{d-1\}, k = 1, ..., K \tag{3.6}$$

$$\sum_{j:(i,j)\in A^k} x_{ij}^k - \sum_{j:(j,i)\in A^k} x_{ji}^k = 0, \quad i \in D \setminus \{d-1, d\}, k = 1, ..., K \tag{3.7}$$

$$\sum_{h=1}^{|H|} \delta_i^h = 1, \quad i \in C \tag{3.8}$$

$$\sum_{h=1}^{|H|} e^h \delta_i^h \leq u_i \leq \sum_{h=1}^{|H|} l^h \delta_i^h, \quad i \in C \tag{3.9}$$

$$\sum_{(i,j):j \in D \setminus \{d-1,d\}} x_{ij}^k \geq \frac{w_d^k}{p} - 1, \quad k = 1, ..., K \tag{3.10}$$

$$\sum_{i:(i,j) \in A^k} e^{|H|+1} x_{ij}^k \leq w_j^k \leq \sum_{i:(i,j) \in A^k} l^{|H|+1} x_{ij}^k, \quad j \in D \setminus \{d-1, d\}, k = 1, ..., K \tag{3.11}$$

$$w_{d-1}^k \leq 0, \quad k = 1, ..., K \tag{3.12}$$

$$w_d^k \leq b, \quad k = 1, ..., K \tag{3.13}$$

$$u_j \geq u_i + s_i + t_{ij} - M\left(1 - x_{ij}^k\right), \quad i, j \in C, k = 1, ..., K \tag{3.14}$$

$$w_j^k \geq u_i + s_i + t_{ij} - M\left(1 - x_{ij}^k\right), \quad i \in C, j \in D \setminus \{d-1\}, k = 1, ..., K \tag{3.15}$$

$$u_j \geq w_i^k + s_i + t_{ij} - M\left(1 - x_{ij}^k\right), \quad i \in D \setminus \{d\}, j \in C, k = 1, ..., K \tag{3.16}$$

$$x_{ij}^k \in \{0, 1\}, \quad (i, j) \in A^k, k = 1, ..., K \tag{3.17}$$

$$y_i^k \in \{0, 1\}, \quad i \in C \cup \{d-1, d\}, k = 1, ..., K \tag{3.18}$$

$$u_i \geq 0, \quad i \in C \tag{3.19}$$

$$w_i^k \geq 0, \quad i \in D, k = 1, ..., K \tag{3.20}$$

$$\delta_i^h \in \{0, 1\}, \quad i \in C, h = 1, ..., |H| \tag{3.21}$$

Como foi referido anteriormente a função objetivo 3.1 é composta por dois termos, a distância total percorrida e a duração total dos circuitos. As restrições 3.2 impõem que cada estação tem que estar num único circuito. As restrições 3.3 e 3.4 estão associadas ao facto de todos os circuitos terem que iniciar e terminar no porto origem. As restrições 3.5 e 3.6 garantem a conservação de fluxo nas estações de pesca e também que cada estação é visitada uma única vez. As restrições 3.7 asseguram a conservação de fluxo em todos os portos excluindo o porto origem cujo vértice foi desdobrado em dois de forma a representar o início e fim de cada rota. As visitas às estações devem ocorrer dentro de exatamente uma janela temporal pré-definida 3.8 e dentro de uma janela admissível 3.9. As restrições 3.10 definem o número de portos intermédios que devem ser visitados em cada circuito de acordo com a periodicidade p. E tal como acontece com as estações, as visitas ao portos têm que ocorrer dentro de uma janela admissível 3.11.

Cada circuito deve começar no instante 0 horas 3.12 e a sua duração não pode ultrapassar as *b* horas 3.13. As restrições 3.14 - 3.16 estabelecem a relação de precedência existente entre as variáveis de rota e as de tempo, para além de garantir que não hajam sub-circuitos. O que está expresso nestas restrições é que caso o vértice j ocorra imediatamente a seguir ao vértice i num circuito a hora de início da operação no vértice *j* tem que ser superior ou igual ao início da operação no vértice *i* somado ao tempo de serviço em *i* e ao tempo de viagem de *i* para *j*. Por fim, as restrições 3.17 - 3.21 definem o domínio das variáveis do modelo.

3.2.2 Modelo SR-MF

Em alternativa ao modelo acima propõe-se o modelo SR-MF em que se utilizam as variáveis w_{ij}^k, com $(i,j) \in A$, k=1,...,K, que indicam o início da operação na estação *j* ou no porto *j*, sabendo que se chega ao vértice *j* através do arco (i,j) no circuito *k*.

$$minimizar \sum_{k=1}^{K} \sum_{(i,j)\in A^k} t_{ij} x_{ij}^k + \sum_{k=1}^{K} \sum_{i\in C} w_{id}^k \tag{3.22}$$

Sujeito a:

$$\sum_{k=1}^{K} y_j^k = 1, \quad j \in C \tag{3.23}$$

$$\sum_{k=1}^{K} y_{d-1}^k = K \tag{3.24}$$

$$\sum_{k=1}^{K} y_d^k = K \tag{3.25}$$

$$\sum_{i:(i,j)\in A^k} x_{ij}^k = y_j^k, \quad j \in C \cup \{d\}, k = 1, ..., K \tag{3.26}$$

$$\sum_{j:(i,j)\in A^k} x_{ij}^k = y_i^k, \quad i \in C \cup \{d-1\}, k = 1, ..., K \tag{3.27}$$

$$\sum_{j:(i,j)\in A^k} x_{ij}^k - \sum_{j:(j,i)\in A^k} x_{ji}^k = 0, \quad i \in D \setminus \{d-1, d\}, k = 1, ..., K \tag{3.28}$$

$$\sum_{i\in C} x_{ij}^k \leq 1, \quad k = 1, ..., K, j \in D \setminus \{d-1, d\} \tag{3.29}$$

$$\sum_{h=1}^{|H|} \delta_i^h = 1, \quad i \in C \tag{3.30}$$

$$\sum_{h=1}^{|H|} e^h \delta_j^h \leq \sum_{k=1}^{K} \sum_{i:(i,j)\in A^k} w_{ij}^k \leq \sum_{h=1}^{|H|} l^h \delta_j^h, \quad j \in C \tag{3.31}$$

$$\sum_{(i,j):j\in D\setminus\{d-1,d\}} x_{ij}^k \geq \frac{\sum_{i\in C} w_{id}^k}{p} - 1, \quad k = 1, ..., K \tag{3.32}$$

$$\sum_{i:(i,j)\in A^k} e^{|H|+1} x_{ij}^k \leq \sum_{i:(i,j)\in A^k} w_{ij}^k \leq \sum_{i:(i,j)\in A^k} l^{|H|+1} x_{ij}^k, \quad j \in D \setminus \{d-1, d\}, k = 1, ..., K \tag{3.33}$$

$$\sum_{j:(i,j)\in A^k} w_{ij}^k \geq \sum_{h:(h,i)\in A^k} w_{hi}^k + \sum_{j:(i,j)\in A^k} (s_i + t_{ij})\, x_{ij}^k, \quad i \in C \cup D \setminus \{d\}, k = 1, ..., K \tag{3.34}$$

$$w_{ij}^k \leq b\, x_{ij}^k, \quad (i,j) \in A^k, k = 1, ..., K \tag{3.35}$$

$$x_{ij}^k \in \{0,1\}, \quad (i,j) \in A^k, k = 1, ..., K \tag{3.36}$$

$$y_i^k \in \{0,1\}, \quad i \in C \cup \{d-1, d\}, k = 1, ..., K \tag{3.37}$$

$$\delta_i^h \in \{0,1\}, \quad i \in C, h = 1, ..., |H| \tag{3.38}$$

$$w_{ij}^k \geq 0, \quad (i,j) \in A^k, k = 1, ..., K \tag{3.39}$$

A função objetivo 3.22 tem dois termos tal como a do modelo SR-MTZ a única diferença é que agora as durações dos circuitos são expressas de acordo com as novas variáveis definidas. As restrições 3.23 - 3.28 são iguais às restrições 3.2 - 3.7 do modelo anterior. As restrições 3.29 são acrescentadas neste modelo para assegurar que os portos intermédios não são visitados mais do que uma vez num mesmo circuito. O significado das restrições 3.30 - 3.33 também é semelhante ao das restrições 3.8 - 3.11 do modelo SR-MTZ, usando a variável de tempo definida para este modelo. Para evitar que haja sub-circuitos e para definir os instantes em que ocorrem as visitas aos portos e estações de forma a respeitar os tempos de serviço em cada vértice, estabelecem-se as restrições 3.34. As restrições 3.35 garantem que só passa fluxo nos arcos que façam parte das rotas e que a duração de cada rota não excede *b*. As restantes restrições definem o domínio das variáveis.

3.2.3 Modelo SR-MFA

Para o último modelo proposto, recorre-se ao uso de dois tipos de variáveis para representar o início do serviço nas estações e nos portos. O início do serviço na estação *j* sabendo que se chega a *j* através do arco (i,j) é representado de forma diferente consoante *i* seja uma estação ou um porto. Assim, as variáveis u_{ij} com *i*, *j* ∈ C designam o início do serviço na estação *j* sabendo que se chega *j* através do arco (i,j). As variáveis w_{ij}^k com k=1,...,K representam o início do serviço na estação *j* sabendo que se chega a *j* vindo do porto *i* no circuito k, quando *i* ∈ D e *j* ∈ C ou representam o início do serviço no porto *j* quando se chega a *j* vindo da estação *i* pelo arco (i,j) no circuito k quando *i* ∈ C e *j* ∈ D.

O Modelo SR-MFA formula-se da seguinte forma:

$$minimizar \sum_{k=1}^{K} \sum_{(i,j)\in A^k} t_{ij} x_{ij}^k + \sum_{k=1}^{K} \sum_{i\in C} w_{id}^k \tag{3.40}$$

Sujeito a:

$$\sum_{k=1}^{K} y_j^k = 1, \quad j \in C \tag{3.41}$$

$$\sum_{k=1}^{K} y_{d-1}^{k} = K \tag{3.42}$$

$$\sum_{k=1}^{K} y_{d}^{k} = K \tag{3.43}$$

$$\sum_{i:(i,j)\in A^k} x_{ij}^{k} = y_{j}^{k}, \quad j \in C \cup \{d\}, k = 1, ..., K \tag{3.44}$$

$$\sum_{j:(i,j)\in A^k} x_{ij}^{k} = y_{i}^{k}, \quad i \in C \cup \{d-1\}, k = 1, ..., K \tag{3.45}$$

$$\sum_{j:(i,j)\in A^k} x_{ij}^{k} - \sum_{j:(j,i)\in A^k} x_{ji}^{k} = 0, \quad i \in D \setminus \{d-1, d\}, k = 1, ..., K \tag{3.46}$$

$$\sum_{i\in C} x_{ij}^{k} \leq 1, \quad k = 1, ..., K, j \in D \setminus \{d-1, d\} \tag{3.47}$$

$$\sum_{h=1}^{|H|} \delta_{i}^{h} = 1, \quad i \in C \tag{3.48}$$

$$\sum_{h=1}^{|H|} e^{h}\delta_{j}^{h} \leq \sum_{i\in C} u_{ij} + \sum_{k=1}^{K}\sum_{i\in D} w_{ij}^{k} \leq \sum_{h=1}^{|H|} l^{h}\delta_{j}^{h}, \quad j \in C \tag{3.49}$$

$$\sum_{(i,j):j\in D\setminus\{d-1,d\}} x_{ij}^{k} \geq \frac{\sum_{i\in C} w_{id}^{k}}{p} - 1, \quad k = 1, ..., K \tag{3.50}$$

$$\sum_{i:(i,j)\in A^k} e^{|H|+1} x_{ij}^{k} \leq \sum_{i:(i,j)\in A^k} w_{ij}^{k} \leq \sum_{i:(i,j)\in A^k} l^{|H|+1} x_{ij}^{k}, \quad j \in D \setminus \{d-1, d\}, k = 1, ..., K \tag{3.51}$$

$$\sum_{j\in C} u_{ij} + \sum_{k=1}^{K}\sum_{j\in D\setminus\{d-1\}} w_{ij}^{k} \geq \sum_{h\in C} u_{hi} + \sum_{k=1}^{K}\sum_{h\in D\setminus\{d\}} w_{hi}^{k} + \sum_{k=1}^{K}\sum_{j\in C\cup D\setminus\{d-1\}} (s_i + t_{ij})\, x_{ij}^{k}, \quad i \in C \tag{3.52}$$

$$\sum_{j\in C} w_{ij}^{k} \geq \sum_{h\in C} w_{hi}^{k} + \sum_{j\in C} (s_i + t_{ij})\, x_{ij}^{k}, \quad i \in D \setminus \{d\}, k = 1, ..., K \tag{3.53}$$

$$w_{ij}^{k} \leq b\, x_{ij}^{k}, \quad i \in C \cup D \setminus \{d\}, j \in C \cup D \setminus \{d-1\}, k = 1, ..., K \tag{3.54}$$

$$u_{ij} \leq b \sum_{k=1}^{K} x_{ij}^{k}, \quad i, j \in C, i \neq j \tag{3.55}$$

$$x_{ij}^{k} \in \{0, 1\}, \quad (i, j) \in A^k, k = 1, ..., K \tag{3.56}$$

$$y_{i}^{k} \in \{0, 1\}, \quad i \in C \cup \{d-1, d\}, k = 1, ..., K \tag{3.57}$$

$$\delta_{i}^{h} \in \{0, 1\}, \quad i \in C, h = 1, ..., |H| \tag{3.58}$$

$$u_{ij} \geq 0, \quad i,j \in C \tag{3.59}$$

$$w_{ij}^{k} \geq 0, \quad i \in C \cup D \setminus \{d\}, j \in C \cup D \setminus \{d-1\}, k = 1, ..., K \tag{3.60}$$

O modelo SR-MFA apresenta também a mesma função objetivo que o segundo modelo, assim como as restrições 3.41 - 3.48, 3.50 - 3.51. Nesta formulação a restrição que estabelece as janelas temporais admissíveis para visitas às estações 3.49 é definida desta forma pois, é necessário explicitar separadamente as variáveis de tempo associadas a arcos que ligam estações e as variáveis de tempo associadas a ligações entre portos e estações, devido à forma como estas foram definidas. As restrições 3.52 e 3.53 garantem a eliminição de sub-circuitos e para além disso, definem as relações de precedência para as estações e para os portos, respetivamente. Finalmente, as restrições 3.54 e 3.55, assim como no modelo SR-MF, garantem que só passa fluxo em arcos da solução e impõem um limite superior ao fluxo que passa em cada arco que se encontra na solução. As restantes restrições indicam o domínio de cada variável presente no modelo.

3.3 Considerações Importantes

Nesta secção apontam-se alguns aspetos importantes acerca do problema em estudo e dos modelos propostos.

O SROP, assim como foi definido na secção 3.1, tem como objetivo determinar um conjunto de K rotas que minimizem a distância total percorrida e a duração total dos circuitos. Todos os modelos propostos para formular o problema apresentam uma característica que o torna altamente combinatório, a simetria. Neste caso, a simetria está relacionada com a identificação das rotas. Isto é, considere-se uma solução com duas rotas, uma designada por rota 1 e outra designada por rota 2. Se trocarmos as designações das rotas, isto é, a rota 1 passar a ser designada por rota 2 e vice versa, apesar de estarmos a falar da mesma solução, para os três modelos propostos estas soluções são distintas embora tenham o mesmo valor. Este facto vem agravar a natureza combinatória que o problema em estudo já apresenta havendo interesse em desenvolver algum processo que permita diminuir o número de combinações, quebrando assim a simetria existente.

Relativamente aos modelos, o primeiro ponto a ser referido é sobre as diferenças existentes entre os mesmos, a nível do número de variáveis e de restrições. Como no modelo SR-MTZ, as variáveis de tempo estão associadas a cada nodo então, o número de variáveis é menor do que o número de variáveis dos modelos SR-MF e SR-MFA nos quais estão associadas aos arcos. E ainda, como no modelo SR-MFA não é necessário fazer a discretização por circuito para as variáveis de tempo associadas às estações pois, sabe-se que cada estação vai estar num único circuito, tem-se que este possui um menor número de variáveis do que o modelo SR-MF. Em relação ao número de restrições, tem-se que o modelo SR-MTZ tem mais restrições que os outros modelos. A razão é a mesma, pelo facto das variáveis de tempo estarem associadas a cada nodo, é necessário escrever as restrições que eliminam subcircuitos $(3.14 - 3.16)$ para cada arco da solução, aumentando assim o número de restrições. O modelo SR-MF por sua vez, tem mais restrições que o modelo SR-MFA.

Os três modelos propostos, são equivalentes entre si, ou seja, dada uma solução admissível obtida através de um dos modelos, é possível construir uma solução para um dos outros com o mesmo valor da função objetivo. Relembrando que as variáveis de rota e circuito são as mesmas para todos os modelos, então só é necessário fazer a adaptação das variáveis de tempo ao passar de um modelo para outro. Relacionando as variáveis de tempo do modelo SR-MTZ às variáveis de tempo do modelo SR-MF, tem-se:

- $u_j = \sum_{i \in C \cup \{d-1\}} \sum_{k=1}^{K} w_{ij}^k, \quad j \in C$
- $w_j^k = \sum_{i \in C} w_{ij}^k, \quad j \in D \setminus \{d-1\}, k = 1,...,K$

E para o modelo SR-MFA, tem-se:

- $u_j = \sum_{i \in C} u_{ij} + \sum_{k=1}^{K} \sum_{i \in D} w_{ij}^k, \quad j \in C$
- $w_j^k = \sum_{i \in C} w_{ij}^k, \quad j \in D \setminus \{d-1\}, k = 1,...,K$

3.4 Exemplo de Aplicação

Para uma melhor compreensão dos modelos e do problema, apresenta-se um exemplo de aplicação dos mesmos, com base numa instância de pequena dimensão. Consideram-se 4 estações de pesca C= $\{1, 2, 3, 4\}$, 3 portos D=$\{1, 2, 3\}$ e 2 circuitos $(k = 2)$. Os portos 2 e 3 representam o porto inicial e o porto final, respetivamente. O tempo de serviço para todas as estações é 1 e para os portos é 4. Os conjuntos AEE, AEP e APE representam os arcos existentes entre as estações, entre as estações e os portos e, entre os portos e as estações, respetivamente. E as matrizes TEE, TEP e TPE, representam os tempos associados aos respetivos arcos existentes. O vetor *l* corresponde aos limites superiores para as janelas temporais e o vetor *e* aos limites inferiores. Os parâmetros *p* e *b*, tomam os valores 26 e 58, respetivamente.

AEE = $\{(1,2), (1,3), (1,4), (2,1), (2,3), (2,4), (3,1), (3,2), (3,4), (4,1), (4,2), (4,3)\}$

AEP = $\{(1,1), (1,3), (2,1), (2,3), (3,1), (3,3), (4,1), (4,3)\}$

APE = $\{(1,1), (1,2), (1,3), (1,4), (2,1), (2,2), (2,3), (2,4)\}$

TEE = $\begin{bmatrix} 3 & 2 & 6 \\ 7 & 9 & 10 \\ 5 & 2 & 1 \\ 8 & 4 & 5 \end{bmatrix}$

TEP = $\begin{bmatrix} 7 & 9 \\ 5 & 3 \\ 4 & 2 \\ 6 & 8 \end{bmatrix}$

TPE = $\begin{bmatrix} 2 & 4 & 6 & 8 \\ 9 & 3 & 5 & 7 \end{bmatrix}$

l = $[9, 15, 24, 30, 44, 53, 62, 86, 110, 134, 158, 202, 286, 340, 168]$

e = $[7, 11, 17, 27, 33, 47, 56, 75, 99, 123, 147, 171, 215, 289, 160]$

Esta instância foi resolvida utilizando os modelos SR-MTZ e SR-MFA. Abaixo encontram-se a solução obtida por cada um deles, destacando as variáveis de tempo usadas em cada modelo.

Com o modelo SR-MTZ, obteve-se a solução representada na figura 3.1.

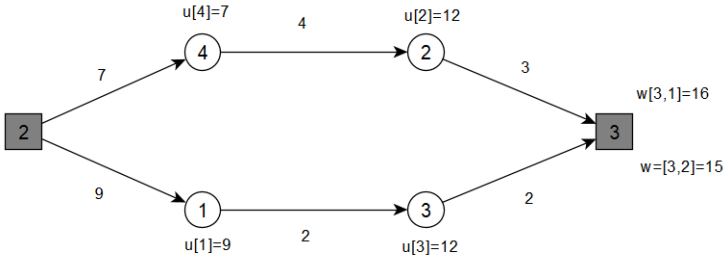

Figura 3.1: Solução admissível para a instância com 3 portos e 4 estações de pesca obtida com o modelo SR-MTZ

O valor desta solução é 58 = 15 + 16 + 7 + 9 + 4 + 2 + 3+ 2. Pode-se observar que todas as estações são visitadas, e que tanto o circuito **P2**-4-2-**P3** como o circuito **P2**-1-3-**P3** são concluídos antes das 58 horas. Também nota-se que pelo mesmo motivo, não foi necessário visitar um porto intermédio pois cada uma das rotas termina antes das 26 horas. Relativamente ao início das operações em cada estação, aqui denotadas por $u\,[i]$, observa-se que para as primeiras estações visitadas o início da operação ocorre dentro da primeira janela $[7, 9]$ e, para as últimas estações a operação decorre dentro da segunda janela temporal $[11, 15]$.

Resolvendo a mesma instância com o modelo SR-MFA, os circuitos e o valor da solução obtidos são iguais aos obtidos pelo modelo SR-MTZ. Sabendo que os modelos são equivalentes, ao obter uma solução admissível para um dos modelos pode-se construir uma solução admissível para os outros modelos também. Na figura 3.2 apresenta-se a solução do modelo SR-MFA destacando-se novamente as variáveis de tempo. No caso do modelo SR-MF teríamos o seguinte: w[**P2**, 4, 1] = 7, w[**P2**, 1, 2] = 9, w[4, 2, 1] = 12, w[1, 3, 2] = 12, w[2, **P3**, 1] = 16 e w[3, **P3**, 2] = 15.

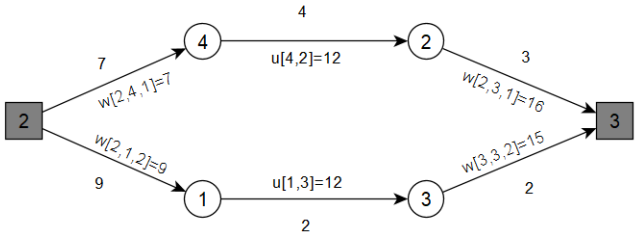

Figura 3.2: Solução admissível para a instância com 3 portos e 4 estações de pesca obtida com o modelo SR-MFA

Após a ilustração da diferença entre as variáveis de tempo dos modelos SR-MTZ e SR-MFA, será apresentado um exemplo de uma solução não admissível. Considerando agora que p=16, na figura 3.3 , observa-se que se chega à estação 4 no instante 16 portanto, a próxima visita deveria ser a um porto e não a uma estação devido à restrição da periodicidade com que se deve visitar os portos.

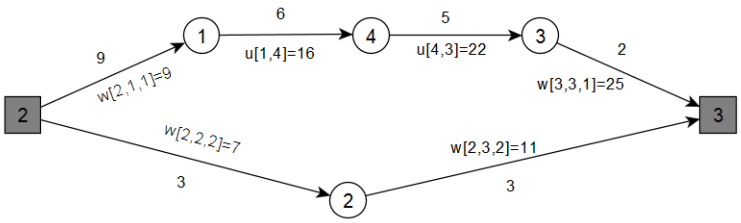

Figura 3.3: Solução não admissível para a instância com 3 portos e 4 estações de pesca e p=14

3.5 Determinação de Limites Inferiores

Os modelos apresentados nas secções anteriores são modelos alternativos que permitem obter a solução ótima do problema em estudo. Qualquer que seja o modelo escolhido, o valor da respetiva relaxação linear fornece um limite inferior para o valor ótimo. Apesar dos modelos serem equivalentes as suas relaxações lineares não são e tem interesse estuda-las e compara-las quer do ponto de vista da qualidade dos limites inferiores quer do ponto de vista do tempo computacional gasto para os obter. As referidas relaxações lineares podem ser fortalecidas considerando desigualdades válidas.

3.5.1 Relaxações Lineares

A relaxação linear é um problema formulado como o problema original, onde as restrições de domínio das variáveis inteiras são relaxadas, isto é, as variáveis inteiras passam a tomar valores reais. Como foi referido, resolvendo as relaxações lineares consegue-se obter limites inferiores para o problema.

Assim sendo, serão consideradas relaxações lineares totais, ou seja, relaxações onde o domínio de todas as variáveis inteiras são relaxadas e, relaxações lineares parciais, onde se relaxa o domínio de apenas algumas variáveis.

Nos modelos propostos na secção 3.2 deste capítulo, existem três conjuntos de variáveis que tomam valores inteiros, x_{ij}^k com i,j $\in A^k$, k=1,...K, y_i^k com i $\in$ C $\cup \{d-1, d\}$, k=1,...K e, δ_i^h com i $\in$ C, h=1,...,$|H|$. Assim sendo, na relaxação linear os domínios destas variáveis passarão a ser:

- variáveis δ_i^h com i $\in$ C, passam a tomar valores entre 0 e 1, ou seja, $0 \leq \delta_i^h \leq 1$, mas como existe a restrição $\sum_{h=1}^{|H|} \delta_i^h = 1, \quad i \in C$, sabe-se que δ_i^h será sempre menor ou igual a 1, portanto basta impor que $\delta_i^h \geq 0$, i $\in$ C, h = 1,...,$|H|$;

- com as variáveis y_i^k, i $\in$ C $\cup \{d-1, d\}$, acontece o mesmo, a nova restrição de domínio $y_i^k \geq 0$, i $\in$ C $\cup \{d-1, d\}$, k=1,...,K, é suficiente devido à restrição $\sum_{k=1}^{K} y_i^k = 1, \quad i \in C$ que fará com que $y_i^k \leq 1$, i $\in$ C seja sempre verificado. No entanto, quando i $\in \{d-1, d\}$ não se garante que $y_i^k \leq 1$, por causa das restrições que impõem que todos os circuitos têm que começar e terminar no porto origem, logo nas relaxações lineares tem que se acrescentar as seguintes restrições:

  $$y_{d-1}^k \leq 1, \quad k = 1, ..., K \tag{3.61}$$

  $$y_d^k \leq 1, \quad k = 1, ..., K \tag{3.62}$$

  Estas restrições garantem que o porto origem (partida e chegada) seja visitado apenas uma vez em cada circuito;

- analogamente às outras variáveis, para a variável x_{ij}^k com i,j $\in A^k$, k=1,...K, a nova restrição de domínio passa a ser apenas $x_{ij}^k \geq 0$, sendo que não é necessário obrigar que x_{ij}^k seja menor ou igual a 1 pois, as restrições que garantem que cada estação é visitada apenas uma vez consequentemente, acabam por garantir que $x_{ij}^k \leq 1$, i,j $\in A^k$, k=1,...K.

Para a obtenção de limites inferiores consideraram-se algumas relaxações do problema baseadas na relaxação linear de subconjuntos de variáveis inteiras. Estas propostas, surgiram com vista à tentativa de obter melhores limites do que os limites fornecidos pela relaxação linear em que se considera que todas as variáveis são contínuas não negativas e que será denotada por **RL**.

No segundo caso propõe-se que as variáveis x_{ij}^k com i $\in$ C, j $\in$ D ou i $\in$ D, j $\in$ C , k=1,...K, continuassem a ser binárias como no modelo inicial e apenas as restantes fossem relaxadas. Este teste foi designado como **RL_xb**. O objetivo deste é observar se obrigando que o fluxo que sai e entra nos portos seja unitário se consegue obter mais fluxos unitários entre as estações ficando assim mais próximo de uma solução inteira.

Também se sugeriu uma relaxação linear parcial, **RL_yb**, em que as variáveis y_i^k, i $\in$ C $\cup \{d-1, d\}$ seriam binárias, mantendo as restantes entre 0 e 1.

Uma das considerações feitas sobre os modelos propostos, é a questão da simetria. Para tentar eliminar a simetria das soluções pensou-se em fixar algumas variáveis de circuito. As restrições para quebrar simetria podem ser consideradas logo no modelo inteiro e dessa forma aparecem também na relaxação linear. Ao fixar uma estação num circuito, eliminam-se metade das combinações de soluções que poderiam ser feitas. Assim sendo, sugeriu-se que a estação 1 fosse fixada no circuito 1, isto é, y_1^1 = 1 para tentar eliminar alguma simetria.

Também foi sugerido que as variáveis x_{ij}^k com i $\in$ C, j $\in$ D ou i $\in$ D, j $\in$ C , k=1,...K fossem binárias e que a estação 1 fosse fixada ao circuito 1 $\left(y_1^1 = 1\right)$. Daí designar este teste por **RL_xb_y1**. Fixando então esta variável, pretende-se averiguar se a simetria entre as soluções é quebrada e se o tempo computacional deste teste é inferior quando comparado com o **RL_xb**. A mesma restrição $\left(y_1^1 = 1\right)$ foi acrescentada à relaxação **RL_yb**, dando origem ao teste **RL_yb_y1**.

Depois, pensou-se no teste que englobasse todas as alterações mencionadas acima, isto é, com x_{ij}^k com i $\in$ C, j $\in$ D ou i $\in$ D, j $\in$ C , k=1,...K, e y_i^k, i $\in$ C $\cup \{d-1, d\}$ binários e y_1^1 = 1. Denominou-se este teste por **RL_xyb_y1**, pretendendo estudar o efeito dessas imposições na qualidade dos limites inferiores obtidos mas também no tempo computacional.

O objetivo de realizar todos estes testes, será obter o melhor limite inferior no menor tempo computacional possível. Os resultados dos mesmos são apresentados no capítulo dos Resultados Computacionais.

3.5.2 Desigualdades Válidas

As desigualdades válidas, são desigualdades que ao serem incluídas no modelo relaxado reduzem a correspondente região admissível mas, garantindo que todas as soluções inteiras admissíveis permanecem nessa região. Sendo assim, obtém-se um novo modelo válido para o problema original mas que a respetiva relaxação linear pode permitir obter melhores limites inferiores para o valor ótimo do problema.

Foram acrescentadas estes dois conjuntos de desigualdades ao modelo SR-MFA:

$$u_{ij} \geq \sum_{k=1}^{K} \left(s_i + t_{ij} \times x_{ij}^k \right), \quad i, j \in C \tag{3.63}$$

$$x_{ij}^k + x_{ji}^k \leq 1, \quad i,j \in C, k = 1,...,K \tag{3.64}$$

As desigualdades 3.63 são restrições que obrigam a passar fluxo em todos os arcos que se encontram nas soluções. Elas foram adicionadas porque assim como o problema está formulado, nas relaxações lineares pode acontecer que uma variável de fluxo seja nula mesmo quando para o respetivo arco a variável x_{ij}^k seja estritamente positiva. Portanto, excluindo aquele tipo de soluções a qualidade do limite pode melhorar. Note-se que nas soluções inteiras, no máximo sai um arco de cada vértice e é neste que tem que existir fluxo diferente de zero. A adição destas desigualdades dá origem ao teste **RL_xb_y1_u**.

Em todos os modelos, as restrições que impedem que haja subcircuitos só conseguem garantir isso quando as variáveis x_{ij}^k, $(i,j) \in A^k$, k=1,...,K são inteiras. Por exemplo as restrições 3.14 - 3.16 do modelo SR-MTZ, são ineficazes quando as variáveis x_{ij}^k, $(i,j) \in A^k$, k=1,...,K são fracionárias. Por isso, as soluções das relaxações lineares tendem a ter muitos subcircuitos. Assim sendo, as desigualdades 3.64 são acrescentadas para tentar eliminar alguns subcircuitos de dois elementos, surgindo então o teste **RL_xb_y1_x**.

Com o intuito de tentar melhorar os limites inferiores obtidos, acrescentou-se as duas desigualdades ao modelo, originando o teste **RL_xb_y1_ux**.

Como foi dito anteriormente, as relaxações lineares permitem obter limites inferiores para o valor ótimo do problema. Portanto, as desigualdades válidas podem ser vistas como ferramentas para melhor a qualidade dos limites inferiores.

Capítulo 4

Heurística Relaxar e Fixar

As heurísticas são métodos utilizados para tentar resolver problemas de forma mais rápida e eficaz.

Em particular para o caso em estudo, o SROP, uma heurística fornece soluções admissíveis para o problema sendo que os valores dessas soluções são limites superiores para o valor ótimo. Sendo assim, tendo os valores das relaxações lineares e das heurísticas consegue-se estabelecer um intervalo para o valor ótimo de cada instância testada.

Neste capítulo serão apresentadas três heurísticas.

4.1 Motivação e Objetivos

O problema em estudo é complexo e difícil de resolver quando se consideram instâncias reais como as abordadas neste trabalho. Não é por acaso que as metodologias de resolução propostas na literatura por Mesquita *et al.* [11] para resolver o problema se baseiem em metaheurísticas. Um dos objetivos nesta dissertação é precisamente avaliar a capacidade dos modelos apresentados para resolver instâncias reais usando um *solver* comercial, identificando as suas limitações e propondo heurísticas baseadas na resolução faseada destes mesmos modelos tirando partido da informação obtida resolvendo a relaxação linear e fixando o valor de algumas variáveis.

Experiência computacional preliminar revela grande dificuldade na obtenção de soluções admissíveis. Para todas as instâncias consideradas não foi possível encontrar uma solução admissível em 72 horas utilizando os modelos propostos e o *solver* IBM ILOG CPLEX Optimization Studio 12.7.0. Daí a necessidade de desenvolver heurísticas que permitam de algum modo fasear a resolução dos modelos e obter soluções num tempo razoável mesmo que a qualidade das soluções não seja muito boa.

4.2 Apresentação das heurísticas

Nesta secção apresentam-se as heurísticas baseadas na resolução sequencial dos modelos e/ou relaxações dos mesmos.

As heurísticas propostas são baseadas na fixação de subconjuntos de variáveis onde estas fixações são feitas a partir das soluções obtidas resolvendo relaxações lineares parciais. Com estas fixações espera-se que o problema se torne mais fácil e portanto, mais rápido de se resolver. No entanto, este processo tem a dificuldade de à partida não se saber quais e/ou quantas variáveis se devem fixar. Fixar poucas pode ser insuficiente para tornar o modelo mais rápido mas, ao fixar muitas pode-se estar a impor muitas

condições tornando o problema mais difícil ou mesmo impossível. Assim sendo, optou-se por considerar um parâmetro que define o valor a partir do qual se fixaria uma variável a 1.

Para cada modelo considera-se a relaxação linear parcial que em média fornece melhores limites inferiores e escolhem-se as variáveis a fixar. Vamos optar por fixar as variáveis x_{ij}^k com i,j ∈ C, k=1,...K e/ou as y_i^k, i ∈ C, k=1,...,K. As x_{ij}^k estabelecem os arcos que se encontram na solução e as y_i^k determinam em que circuito está cada estação de pesca. Mas fixar as variáveis x_{ij}^k é mais forte do que fixar y_i^k porque estas fixam o arco e as estações a um circuito enquanto que, as y_i^k só fixam estações a um circuito. Por causa disso fixam-se x_{ij}^k = 1, i,j ∈ C, k=1,...K para as que tenham valor superior ou igual a um parâmetro β, com $\beta > 0$. Já as variáveis y_i^k, i ∈ C, k=1,...,K são fixas a 1 quando tomam valor superior ou igual a α, com $\alpha > 0$.

No passo 1 de cada uma das heurísticas, resolve-se **RL_xb_y1** no caso de SR-MTZ e **RL_xb_y1_ux** no caso de SR-MFA.

Algorithm 1 Heurística 1

1. Escolher o valor de α e β.
2. Relaxar todas as variáveis do modelo, exceto as variáveis x_{ij}^k com i ∈ C, j ∈ D ou i ∈ D, j ∈ C, k=1,...K as quais se mantém binárias. Resolver o correspondente problema relaxado. Considere-se a solução ótima deste problema e vai para o passo 3.
3. Considerar que as variáveis x_{ij}^k com i ∈ C, j ∈ D ou i ∈ D, j ∈ C, k=1,...K, e y_i^k, i ∈ C, k=1,...,K são binárias e fixar y_i^k = 1, i ∈ C, k=1,...,K, tal que $y_i^k \geq \alpha$. Resolver o correspondente problema relaxado. Considere-se a solução ótima deste problema e segue-se para o passo 4.
4. Fixar x_{ij}^k = 1, $(i, j) \in$ C, k=1,...K, para as variáveis que tenham valor $\geq \beta$, na solução ótima do passo 3, garantindo que cada estação é no máximo extremo de um arco fixo na solução. Manter as fixações feitas no passo 2 e resolver o modelo PLIM resultante, onde as restantes variáveis são inteiras. Se existir solução, esta é admissível para o problema SROP.

A heurística 2 é semelhante à primeira mas, no passo 4 em vez de resolver o problema inteiro resultante considera-se a relaxação linear parcial onde as variáveis δ_i^h, i ∈ C, h=1,...,$|H|$ podem ser fracionárias.

Algorithm 2 Heurística 2

1. Escolher o valor de α e β.
2. Relaxar todas as variáveis do modelo, exceto as variáveis x_{ij}^k com i ∈ C, j ∈ D ou i ∈ D, j ∈ C, k=1,...K que se mantém binárias. Resolver o correspondente problema relaxado.
3. Considerar que as variáveis x_{ij}^k com i ∈ C, j ∈ D ou i ∈ D, j ∈ C, k=1,...K, e y_i^k, i ∈ C, k=1,...,K são binárias e fixar y_i^k = 1, i ∈ C, k=1,...,K, tal que $y_i^k \geq \alpha$. Resolver o correspondente problema relaxado.
4. Fixar x_{ij}^k = 1, (i,j) ∈ C, k=1,...,K, para as variáveis que tenham valor $\geq \beta$, $\beta > 0$, garantindo que cada estação é no máximo extremo de um arco fixo na solução. Manter as fixações feitas no passo 3 e resolver o correspondente problema relaxado, com as variáveis x_{ij}^k $(i,j) \in A^k$, k=1,...,K e y_i^k, i ∈ C, k=1,...,K binárias e as variáveis δ_i^h, i ∈ C, h=1,...,$|H|$ contínuas. Se na solução obtida as variáveis δ_i^h, i ∈ C, h=1,...,$|H|$ forem todas inteiras então, encontrou-se uma solução admissível para o problema. Caso contrário, segue-se para o passo 5.
5. Se as variáveis δ_i^h, i ∈ C, h=1,...,$|H|$ não forem todas inteiras, considerar os k circuitos induzidos pela solução. Em cada circuito, ignorar as eventuais visitas a portos intermédios. Aplicar o descodificador, um algoritmo para obter os instantes de início de operação em cada estação e em cada porto, verificando as restrições das janelas temporais. Além disso, sempre que a duração do circuito seja superior a 7 dias o descodificador vai incluir um porto no circuito assim como estabelecer a hora de entrada neste porto. Se a duração total de cada circuito for inferior ou igual a $b = 14 \times 24$ horas, encontrou-se uma solução admissível. Caso contrário, segue para o passo 6.
6. Para os circuitos com duração superior a $b = 14 \times 24$ horas, aplica-se um algoritmo de melhoramento local que faz trocas de estações entre circuitos de forma a minimizar a distância percorrida nos circuitos com duração superior a b sem violar as restrições das janelas temporais.

A última heurística proposta baseia-se apenas na fixação das variáveis y_i^k, i ∈ C, k=1,...,K.

Algorithm 3 Heurística 3

1. Escolher o valor de α
2. Relaxar todas as variáveis do modelo exceto as variáveis x_{ij}^k com i ∈ C, j ∈ D ou i ∈ D, j ∈ C, k=1,...K que se mantém binárias. Resolver o correspondente problema relaxado.
3. Considerar que as variáveis y_i^k, i ∈ C, k=1,...,K e todas as variáveis x_{ij}^k, $(i,j) \in A^k$, k=1,...,K são binárias e fixar y_i^k = 1, i ∈ C, k=1,...,K, tal que $y_i^k \geq \alpha$. Resolver o correspondente problema relaxado, em que as variáveis δ_i^h, i ∈ C, h=1,...,$|H|$ são contínuas. Se na solução obtida as variáveis δ_i^h, i ∈ C, h=1,...,$|H|$ forem todas inteiras, encontrou-se uma solução admissível para o problema.
4. Se as variáveis δ_i^h, i ∈ C, h=1,...,$|H|$ não forem todas inteiras, considerar os K circuitos induzidos pela solução. Em cada circuito, ignorar as eventuais visitas a portos intermédios e aplicar o descodificador para obter os instantes de início de operação em cada estação e em cada porto, verificando as restrições das janelas temporais. Se a duração total de cada circuito for inferior ou igual a $b = 14 \times 24$ horas, encontrou-se uma solução admissível.
5. Se a duração de um ou mais circuito, for superior a $b = 14 \times 24$ horas, fazer trocas de estações entre circuitos de forma a minimizar a distância percorrida sem violar as restrições das janelas temporais.

Segue-se abaixo um esquema que mostra os passos de cada heurística sugerida.

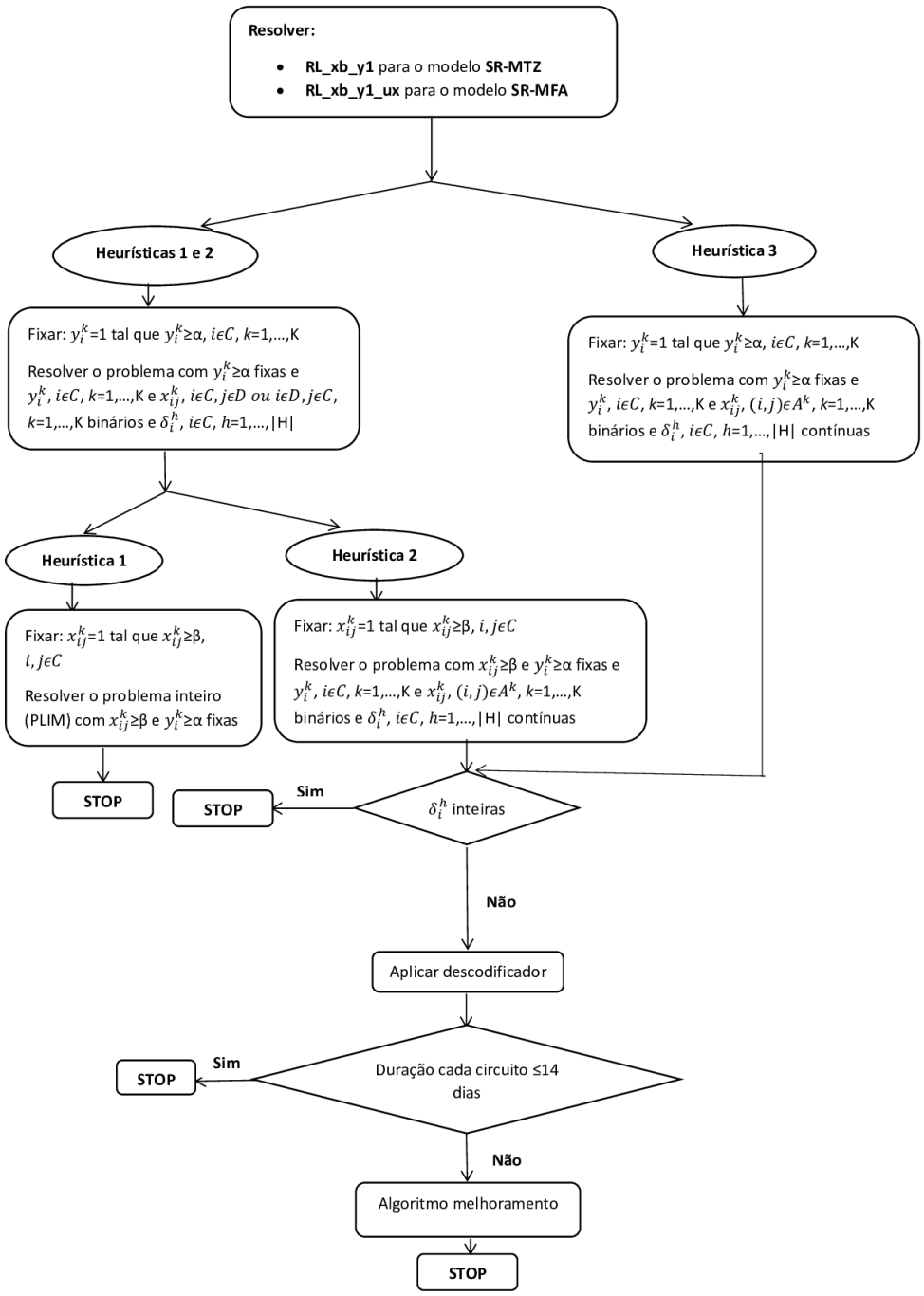

Figura 4.1: Esquema de representação dos passos das heurísticas

A solução encontrada no passo 3 da heurística 3 e no passo 4 da heurística 2 quando as variáveis δ_i^h, i $\in$ C, h=1,...,$|H|$ não são inteiras, não respeita as janelas temporais. Mas podemos usar a sequência de visitas sugeridas na solução e depois à posteriori definir os instantes de início de operação de forma a satisfazer as restrições das janelas temporais.

Dado uma lista das estações visitadas em cada circuito o descodificador vai definir os instantes de início das operações nos portos e nas estações, onde os índices *n*+1,...,*n*+d correspondem aos portos. Uma lista de estações pode ser definida como um circuito onde se ignoram os portos e só se consideram as estações pela ordem que elas são visitadas. As visitas aos portos intermédios são ignorados porque no descodificador ao fim de *p* dias, escolhe-se o porto mais próximo e que respeite as restrições das janelas temporais, para ser visitado. Cada lista está associada ao circuito respetivo. O descodificador mencionado nas heurísticas, é o descodificador que se encontra em Mesquita *et al.* [11].

O algoritmo é aplicado para cada circuito k, ou seja, para cada lista de estações. Apesar do descodificador respeitar todas as restrições das janelas temporais e de obrigar a visita a um porto de acordo com a periodicidade *p*, ele não garante a admissibilidade das soluções obtidas. Isto é, não garante que os circuitos finais tenham uma duração inferior ou igual a $b = 14 \times 24$ horas.

Caso as soluções obtidas não sejam admissíveis e as listas de estações sejam desequilibradas em termos do número de estações que cada uma visite, sugere-se um procedimento de trocas de estações entre circuitos. O desequilíbrio das listas está relacionada com o número de estações que se encontra em cada uma tendo em conta o total de estações. Estas trocas são feitas da seguinte forma, aplicando o descodificador à maior lista e vendo até onde é possível concluir o circuito antes de $b = 14 \times 24$ horas. As estações que não foram incluídas nesta lista passam para a lista com menor número de estações. E aplica-se o descodificador às listas de estações resultantes das trocas efetuadas.

Por exemplo, se tivermos uma instância com 15 estações de pesca e, as seguintes listas de estações:

2 5 8 12 9 13 15 7 10

1 3 14 6

4 11

Suponhamos que, ao aplicar o descodificador se tem que a primeira lista só é admissível até a estação 13, isto é, depois de visitar a estação 13 tem que regressar a Lisboa porque senão excede a duração máxima do cada circuito. Transferindo as restantes estações da primeira lista para a última, obtemos o seguinte:

2 5 8 12 9 13

1 3 14 6

4 11 15 7 10

Se ao aplicar o descodificador a estas listas de estações, a duração total de cada circuito for inferior ou igual a $b = 14 \times 24$ horas, obtém-se uma solução inteira admissível.

Algorithm 4 Pseudocódigo do Descodificador

Require: *lista_estacoes*, *n*-tamanho da *lista_estacoes*

window = 1; *pre* = 0; $w_{d-1} = 0$
for d = n+1, n+d **do**
  $w_d = 0$
end for
station = *lista_estacoes*(1)
$w_{estation}$ = w_0 + $t_{0\ station}$
totaldist = $t_{0\ station}$
for $k = 2, n$ **do**
  pre = *lista_estacoes*$(k-1)$
  station = *lista_estacoes*(k)
  $w_{estation}$ = w_{pre} + s_{pre} + $t_{pre\ station}$
  if $w_{station} \geq e_{window}$ **then**
    if *window* = *p* **then**
      pmin = *n* + 1
      for $d = n + 2, n + d$ **do**
        if $t_{pre\ pmin} \geq t_{pre\ d}$ **then**
          pmin = *d*
        end if
      end for
      w_{pmin} = w_{pre} + s_{pre} + $t_{pre\ pmin}$
      $w_{station}$ = w_{pmin} + s_{pmin} + $t_{pmin\ station}$
      totaldist = *totaldist* + $t_{pre\ pmin}$ + $t_{pmin\ station}$
    else
      totaldist = *totaldist* + $t_{pre\ station}$
    end if
    window = *window* + 1
    if $w_{station} \geq e_{window}$ **then**
      while $w_{station} \geq l_{window}$ **do**
        window = *window* + 1
      end while
    end if
    if $w_{station} \leq e_{window}$ **then**
      $w_{station} = e_{window}$
    end if
  else
    totaldist = *totaldist* + $t_{pre\ station}$
  end if
end for
w_{n+d+1} = $w_{station}$ + $s_{station}$ + $t_{station\ n+d+1}$
totaldist + *totaldist* + $t_{station\ n+d+1}$
objective_score = *totaldist* + w_{d+p+1}
return *objective_score* and w_j for all j $\in V$

Capítulo 5

Resultados Computacionais

Neste capítulo apresentam-se os resultados dos testes computacionais realizados, com base nos modelos propostos no capítulo anterior. Começa-se por descrever as instâncias de teste utilizadas e, por explicar alguns parâmetros presentes nos modelos e nas instâncias. Posteriormente serão apresentados os valores obtidos, isto é, os limites inferiores e superiores para o valor ótimo de cada instância do problema. Por fim, com base nos valores obtidos, será feita uma comparação entre os 3 modelos.

Todos os modelos foram implementados utilizando o *software* IBM ILOG CPLEX *Optimization Studio* 12.7.0 num computador com o processador Intel® Core™ 2 Duo CPU E7500 @ 2.93GHz, com 4GB de RAM.

5.1 Instâncias de teste

Para testar os modelos foram utilizadas três instâncias presentes na literatura, das quais duas, r201.tsphs e pr03.tsphs, foram apresentadas em [5] e podem ser consultadas em [1]. Estas foram geradas para o problema do TSP com seleção de hotéis. A instância r201.tsphs foi gerada considerando um conjunto de 100 clientes. Para adaptar esta instância ao problema em estudo nesta dissertação, estabeleceu-se de acordo com a localização geográfica que o cliente 43 corresponderia ao porto de partida e de chegada e os clientes 23, 38 e 67 corresponderiam aos portos intermédios, e os restantes às estações de pesca. Resultando numa instância com 96 estações e 4 portos. Já a instância pr03.tsphs considera originalmente 144 clientes. Para adaptar ao caso do problema SROP considerou-se que o cliente 30 seria o porto de origem e de chegada e os clientes 9, 49 e 116 corresponderiam aos portos intermédios e os restantes 140 clientes às estações de pesca. A outra instância utilizada (dadosP) possui dados referentes às estações e aos portos de Portugal de uma campanha realizada pelo IPMA, em que foram considerados 4 portos: o porto de partida e chegada (Lisboa) e três possíveis portos intermédios (Portimão, Figueira da Foz e Matosinhos). Em relação ao número de estações são consideradas 95, 96 e 140 estações de pesca, nas instâncias dadosP, r201.tsphs e pr03.tsphs, respetivamente.

Para além das matrizes de distância, dos conjuntos dos arcos existentes e do número de estações e portos, considera-se um vetor s_i associado ao tempo gasto na operação em *i*. Se *i* for uma estação este valor é função da profundidade e dado por $s_i = \frac{prof[i]+\varepsilon}{60}$, em que prof$[i]$ representa a profundidade da estação e ε está relacionado com o tipo de dispositivo que foi utilizado na operação, isto é, ε=15 se se utiliza o dispositivo mais eficiente ou ε=30 no caso contrário. Se *i* for um porto s_i=4. Para a instância pr03.tsphs considera-se dois conjuntos de dados, um utilizando o parâmetro ϵ = 15 e outro ϵ = 30. Para

as outras instâncias não se fez esta distinção, pois a diferença era pouco significativa.

Existe também um parâmetro designado por *conv* que serve para converter as distâncias em tempo. Este parâmetro está relacionado com a velocidade (**VEL**) do navio medida em nós (um nó corresponde a 1,852 Km/h). Esta velocidade é influenciada pelas condições climáticas. Portanto, quanto melhor forem as condições climáticas maior será a velocidade do navio. Neste caso consideram-se três velocidades possíveis: 11 nós (*conv* = 0.05) quando as condições climáticas são muito boas, 9 nós (*conv* = 0.0588) quando as condições são boas e, 7 nós (*conv* = 0.0714) quando não são muito boas.

Todos os outros parâmetros antes mencionados, como a duração máxima de cada circuito (b=14×24 horas), o número máximo de dias em cada circuito ($|H|$=14) e a periodicidade (p=7×24 horas) foram utilizados com estes valores durante toda a experiência computacional.

5.2 Relaxações Lineares

Começou-se por determinar limites inferiores baseados na relaxação linear.

Para perceber melhor os resultados que serão apresentados, é importante perceber a dimensão dos problemas a serem resolvidos por cada modelo e para cada instância. Apresenta-se assim a tabela abaixo com o número de variáveis e restrições fornecidos pelo IBM ILOG CPLEX *Optimization Studio* 12.7.0, tanto para os casos de 2 circuitos como para os casos de 3 circuitos para todas as instâncias.

Tabela 5.1: Comparação dos modelos relativamente ao número de variáveis e restrições

	Modelo SR-MTZ		**Modelo SR-MF**		**Modelo SR-MFA**	
	# Variáveis	**# Restrições**	**# Variáveis**	**# Restrições**	**# Variáveis**	**# Restrições**
DadosP_2	21391	20845	41044	20762	32114	11737
DadosP_3	31374	31029	60901	30952	43041	12902
r201_2	21422	20866	41092	20778	31972	11562
r201_3	31413	31058	60966	30974	42726	12542
pr03_2	43554	42734	84564	42602	65002	23002
pr03_3	64126	63750	125866	63622	86946	24422

Na tabela 5.2 encontram-se os resultados das relaxações lineares totais para os modelos apresentados no capítulo 3, onde os tempos são registados em segundos e a coluna **VEL** representa a velocidade. Pode-se ver que em média as relaxações são rápidas mas, é notável a superioridade dos tempos do modelo SR-MFA relativamente aos outros. Observa-se que os valores ótimos das relaxações lineares para os modelos SR-MF e SR-MFA são exatamente os mesmos, no entanto o tempo necessário para obter estes mesmos resultados é diferente. Como se esperava, tendo em conta a diferença nas dimensões dos referidos modelos para uma mesma instância. A diferença entre os limites inferiores dados pelos modelos SR-MF e SR-MFA e os limites inferiores dados por SR-MTZ é muito significativa, sendo que os limites obtidos com os modelos SR-MF e SR-MFA são muito superiores.

O valor da função objetivo diminui à medida que a velocidade do barco aumenta. Como era expectável, as instâncias pr03.tsphs demoram mais tempo a obter uma solução por terem maior número de estações de pesca.

Tabela 5.2: Valores das Relaxações lineares obtidos em todos os modelos

			SR-MTZ		SR-MF		SR-MFA	
		VEL	**RL**	**Tempo**	**RL**	**Tempo**	**RL**	**Tempo**
DadosP	2 circuitos	7	82.97	3.67	275.42	11.09	275.42	7.51
		9	68.33	2.92	244.55	9.14	244.55	4.67
		11	58.10	2.32	223.11	8.49	223.11	5.13
	3 circuitos	7	86.18	3.31	286.06	12.17	286.06	5.71
		9	70.97	3.35	254.01	13.24	254.01	5.60
		11	60.35	3.25	231.74	11.80	231.74	5.41
r201	2 circuitos	7	79.31	2.36	263.64	7.65	263.64	4.10
		9	65.32	2.17	235.41	7.67	235.41	4.49
		11	55.54	2.25	215.70	7.26	215.70	4.50
	3 circuitos	7	81.41	3.21	272.00	11.66	272.00	5.12
		9	67.04	3.23	242.98	11.40	242.98	5.02
		11	57.01	3.28	222.74	11.48	222.74	5.94
pr03_15	2 circuitos	7	90.63	4.39	355.28	29.87	355.28	10.69
		9	74.63	4.39	322.05	28.81	322.05	10.24
		11	63.47	4.37	298.89	28.25	298.89	10.03
	3 circuitos	7	95.31	6.76	368.25	39.97	368.25	13.12
		9	78.49	6.75	333.46	39.73	333.46	12.69
		11	66.75	6.81	309.21	42.63	309.21	12.24
pr03_30	2 circuitos	7	90.63	4.43	405.00	29.98	405.00	10.80
		9	74.63	4.34	371.64	27.76	371.64	9.61
		11	63.47	4.51	348.39	31.42	348.39	10.31
	3 circuitos	7	95.31	6.74	417.91	39.24	417.91	12.80
		9	78.49	6.68	382.99	40.50	382.99	13.12
		11	66.75	6.71	358.65	39.68	358.65	12.19
Média	2 circuitos		72.25	3.51	296.59	18.95	296.59	7.67
	3 circuitos		75.34	5.01	306.67	26.13	306.67	9.08

Assim como foi explicado na secção 3.4 do capítulo 3, para além dos limites fornecidos pelas relaxações lineares totais, pretende-se realizar outros testes com vista a tentar melhorar os limites inferiores já encontrados. Na tabela 5.3 estão os resultados dos testes **RL_xb**, onde as variáveis x_{ij}^k relacionadas com os arcos que saem e entram no porto de chegada e partida se mantém binárias, onde se vê que os limites obtidos são melhores do que os limites da **RL**. Portanto, constata-se que mudando o domínio das variáveis x_{ij}^k com i ∈ C, j ∈ D ou i ∈ D, j ∈ C , k=1,...K para binários, e mantendo o domínio das outras variáveis na relaxação, consegue-se realmente melhorar os limites. No entanto o tempo computacional necessário para resolver os problemas parcialmente relaxados aumentou consideravelmente, em média os tempos aumentaram mais de 10 vezes.

Ao analisar os valores obtidos observa-se mais uma vez que os modelos SR-MF e SR-MFA produzem os mesmos valores, mas neste caso a diferença entre os tempos é muito maior. O teste que demorou mais tempo no modelo SR-MF foi o teste com a instância pr03.tsphs com ϵ = 30, **VEL** = 11 e K = 3, e demorou 30005 segundos (cerca de 8 horas). Ao passo que no modelo SR-MFA, a instância que mais demorou foi a pr03.tsphs também com ϵ = 30, **VEL** = 9 e K = 3 e, demorou 2162.50 segundos, que é aproximadamente 36 minutos.

Outro facto importante a mencionar, é que com esta alteração a melhoria nos limites obtidos com o modelo SR-MTZ é mais visível do que nos outros modelos. Mas mesmo assim, os modelos SR-MF e SR-MFA continuam a fornecer melhores limites do que o primeiro modelo.

Tabela 5.3: Relaxação linear parcial: as variáveis x_{ij}^k com i ∈ C, j ∈ D ou i ∈ D, j ∈ C são binárias

			SR-MTZ		**SR-MF**		**SR-MFA**	
		VEL	**RL_xb**	**Tempo**	**RL_xb**	**Tempo**	**RL_xb**	**Tempo**
DadosP	2 circuitos	7	102.68	43.78	278.92	344.16	278.92	103.87
		9	87.38	28.04	248.07	329.79	248.07	94.87
		11	76.70	36.23	226.98	360.04	226.98	93.87
	3 circuitos	7	115.96	41.93	290.42	1045.65	290.42	173.72
		9	99.73	48.95	258.77	966.67	258.77	210.16
		11	88.40	51.53	237.14	1320.21	237.14	201.44
r201	2 circuitos	7	97.65	49.63	269.52	588.70	269.52	128.80
		9	83.24	3.57	239.73	366.59	239.73	86.56
		11	73.18	3.71	220.29	477.66	220.29	105.48
	3 circuitos	7	109.27	94.43	279.14	5106.96	279.14	1020.26
		9	94.22	70.16	249.15	3300.64	249.15	489.81
		11	83.71	95.80	229.34	5958.31	229.34	536.15
pr03_15	2 circuitos	7	113.09	86.24	358.86	1695.61	358.86	540.72
		9	95.96	125.66	325.47	2520.47	325.47	509.99
		11	83.99	85.94	302.22	1781.08	302.22	638.96
	3 circuitos	7	128.25	132.42	371.18	11308.79	371.18	1890.09
		9	109.85	221.34	336.87	9832.52	336.87	1808.18
		11	97.01	205.22	312.97	8107.66	312.97	1698.84
pr03_30	2 circuitos	7	113.09	57.58	408.81	2221.03	408.81	562.02
		9	95.96	93.73	375.24	2389.66	375.24	543.17
		11	83.99	109.58	351.85	1821.62	351.85	660.99
	3 circuitos	7	128.25	224.55	421.06	12601.01	421.06	1909.64
		9	109.85	179.16	386.58	10983.88	386.58	2162.50
		11	97.01	271.78	362.56	30004.96	362.56	1680.55
Média	2 circuitos		92.24	60.31	300.50	1241.37	300.50	339.11
	3 circuitos		113.98	136.44	311.26	8378.11	311.26	1148.45

Como foi referido, os modelos sugeridos apresentam problemas de simetria e no sentido de a quebrar foi proposto acrescentar a restrição $y_1^1 = 1$, eliminando-se assim algumas soluções simétricas que se obteriam trocando a identificação do circuito que visita a estação 1. Esta restrição foi acrescentada a todos os modelos e resolveu-se a relaxação linear parcial mantendo as variáveis x_{ij}^k com i ∈ C, j ∈ D ou i ∈ D, j ∈ C binárias (RL_xb_y1). Pela tabela 5.4, pode-se ver que os limites resultantes são exatamente os mesmos que foram obtidos na **RL_xb**. A inclusão desta restrição permite eliminar algumas simetrias reduzindo o tempo computacional médio.

Mais uma vez, constata-se que os modelos SR-MF e SR-MFA produzem sempre os mesmos limites. Assim, optou-se por continuar o estudo computacional apenas com os modelos SR-MTZ e SR-MFA pois, o modelo SR-MF demora muito mais tempo para fornecer os mesmos limites que o SR-MFA. E o objetivo, é obter melhores resultados possíveis no menor tempo computacional possível.

Posto isto, de todos os testes realizados para todos os modelos o **RL_xb_y1** pode ser escolhido como a melhor forma de obter limites inferiores.

Tabela 5.4: Valores das Relaxações obtidos em todos os modelos, considerando que as variáveis x_{ij}^k com i ∈ C, j ∈ D ou i ∈ D, j ∈ C , k=1,...K são binárias e que $y_1^1 = 1$

			SR-MTZ		**SR-MF**		**SR-MFA**	
		VEL	**RL_xb_y1**	**Tempo**	**RL_xb_y1**	**Tempo**	**RL_xb_y1**	**Tempo**
DadosP	2 circuitos	7	102.68	5.40	278.92	311.10	278.92	96.41
		9	87.38	6.01	248.07	295.27	248.07	78.06
		11	76.70	5.37	226.98	340.95	226.98	92.70
	3 circuitos	7	115.96	43.22	290.42	802.71	290.42	145.09
		9	99.73	35.43	258.77	852.27	258.77	154.86
		11	88.40	50.17	237.14	1239.21	237.14	149.08
r201	2 circuitos	7	97.65	3.87	269.52	551.92	269.52	158.03
		9	83.24	3.71	239.73	350.46	239.73	65.17
		11	73.18	3.67	220.29	398.64	220.29	90.38
	3 circuitos	7	109.27	75.18	279.14	3628.98	279.14	938.64
		9	94.22	69.04	249.15	2189.77	249.15	339.50
		11	83.71	74.04	229.34	4574.17	229.34	446.77
pr03_15	2 circuitos	7	113.09	81.33	358.86	1316.54	358.86	479.29
		9	95.96	114.15	325.47	1801.49	325.47	468.73
		11	83.99	23.30	302.22	1564.38	302.22	578.47
	3 circuitos	7	128.25	120.68	371.18	11282.01	371.18	1847.44
		9	109.85	178.75	336.87	9020.63	336.87	1630.42
		11	203.27	189.66	312.97	7906.98	312.97	1380.55
pr03_30	2 circuitos	7	113.09	54.72	408.81	2072.73	408.81	418.44
		9	95.96	84.34	375.24	1874.71	375.24	457.85
		11	83.99	103.29	351.85	1323.30	351.85	511.33
	3 circuitos	7	128.25	196.33	421.06	10214.13	421.06	1201.16
		9	109.85	123.06	386.58	7550.67	386.58	1691.17
		11	97.01	204.85	362.56	24831.38	362.56	1577.04
Média	2 circuitos		92.24	40.76	300.50	1016.79	300.50	291.24
	3 circuitos		113.98	113.37	311.26	7007.74	311.26	958.48

Para observar melhor a diferença entre os tempos de computação dos modelos SR-MF e SR-MFA, construiu-se os seguintes gráficos com base nos tempos de computação da tabela 5.4. Onde é notável a discrepância entre os tempos computacionais obtidos.

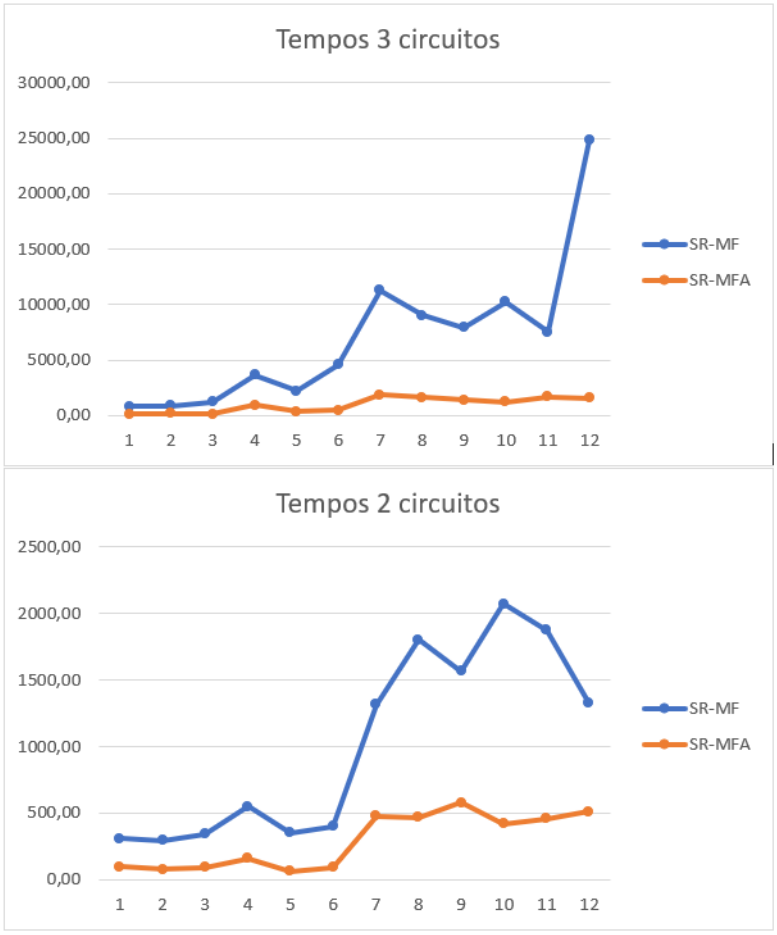

Figura 5.1: Comparação dos tempos dos Modelos SR-MF e SR-MFA obtidos em **RL_xb_y1**

Como foi antes referido a melhoria nos limites inferiores obtidos pelo modelo SR-MTZ, relativamente aos testes **RL** e **RL_xb_y1**, é mais visível do que nos outros modelos. No gráfico abaixo, percebe-se a melhoria que houve nos limites do modelo SR-MTZ.

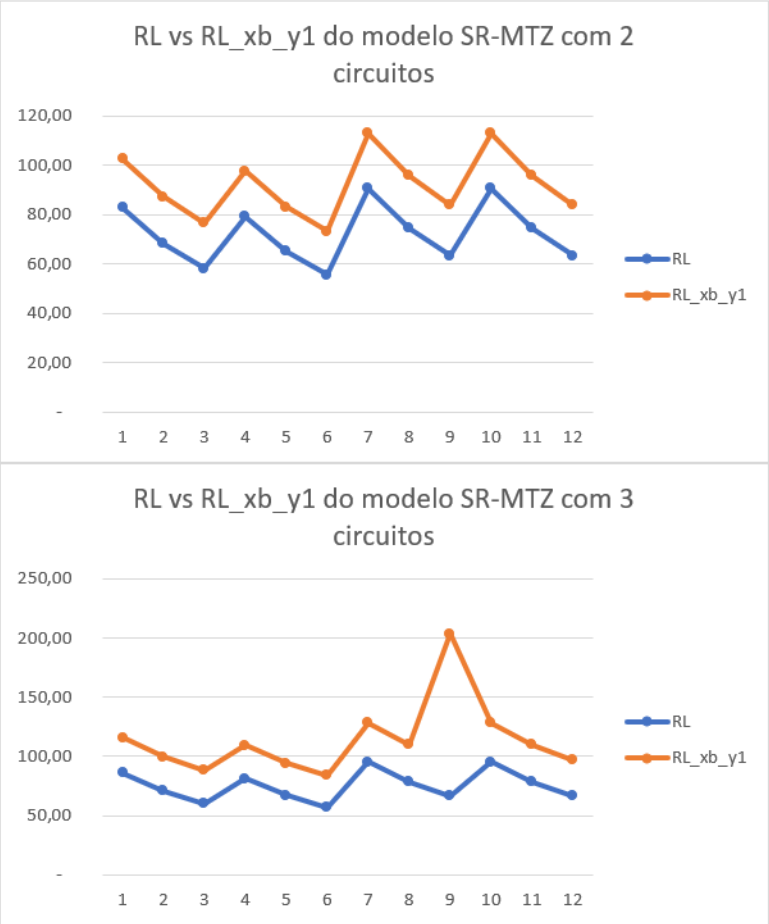

Figura 5.2: Comparação dos limites inferiores obtidos pelo modelo SR-MTZ

Outra relaxação linear parcial que se testou foi a obtida relaxando todas as variáveis exceto as y_i^k, i $\in$ C $\cup$ $\{d-1, d\}$, k=1,...,K, que se mantêm binárias. Com o objetivo de tentar melhorar os limites e obter soluções mais próximas das soluções inteiras. Isto é, ao obrigar que as variáveis que indicam em que circuito cada estação está, sejam binárias espera-se que isto obrigue a que as variáveis x_{ij}^k venham maioritariamente binárias nas soluções.

Na tabela 5.5, nas colunas **RL_yb** vê-se que os limites inferiores obtidos não são melhores que os limites obtidos nos outros testes e que, o tempo computacional é superior aos testes realizados anteriormente. Notando que, os resultados obtidos são muito semelhantes aos obtidos na **RL**, concluindo assim que estes testes não trazem nenhuma informação nova.

Para tentar diminuir os tempos computacionais dos testes **RL_yb**, acrescentou-se a restrição $y_1^1 = 1$. Pelas médias, consegue-se perceber que os tempos computacionais realmente diminuíram.

Para o modelo SR-MFA, considerou-se apenas um subconjunto de instâncias para testar a **RL_yb** e a **RL_yb_y1**, pois pode-se ver que nos casos de 3 circuitos os tempos computacionais utilizados são muito elevados. E, como já se tinha concluído que estes não melhoravam os limites obtidos pela **RL**, então decidiu-se não realizar os testes para todas as instâncias. Analisando com base nos resultados obtidos com este subconjunto, percebe-se que os limites não melhoram e, que os tempos médios da **RL_yb_y1** são inferiores aos da **RL_yb**.

Tabela 5.5: Valores obtidos nos testes **RL_yb_y1** e **RL_yb** realizados com os modelos SR-MTZ e SR-MFA

			SR-MTZ				**SR-MFA**			
		VEL	**RL_yb**	**Tempo**	**RL_yb_y1**	**Tempo**	**RL_yb**	**Tempo**	**RL_yb_y1**	**Tempo**
DadosP	2 circuitos	7	82.98	16.93	82.98	8.92	275.90	1312.50	275.90	166.08
		9	68.33	16.32	68.33	15.44				
		11	58.11	5.03	58.11	13.69				
	3 circuitos	7	86.19	38.17	86.19	24.58				
		9	70.98	52.06	70.98	34.69				
		11	60.36	21.14	60.36	46.97				
r201	2 circuitos	7	79.32	15.51	79.32	13.23				
		9	65.32	5.39	65.32	2.60	235.43	790.92	235.43	168.37
		11	55.54	3.45	55.54	3.70				
	3 circuitos	7	81.42	74.45	81.42	45.74				
		9	67.05	41.98	67.05	41.87				
		11	57.02	59.75	57.02	38.87				
pr03_15	2 circuitos	7	90.64	84.91	90.64	79.60				
		9	74.65	96.39	74.65	53.52				
		11	63.47	114.51	63.47	90.24				
	3 circuitos	7	95.32	605.32	95.32	595.85				
		9	78.50	790.25	78.50	675.62				
		11	66.75	610.79	66.75	520.87	310.20	118800.00	310.20	88125.57
pr03_30	2 circuitos	7	90.64	100.17	90.64	69.13				
		9	74.65	123.67	74.65	116.58				
		11	63.47	111.97	63.47	68.81				
	3 circuitos	7	95.33	884.14	95.33	706.93				
		9	78.50	720.56	78.50	701.34				
		11	66.75	847.68	66.75	693.99	359.03	86408.18	359.03	71040.56
Média	2 circuitos		72.26	57.85	72.26	44.62	255.66	175.29	255.67	167.23
	3 circuitos		75.35	395.52	75.35	343.94	334.61	102604.09	334.62	79583.07

Testou-se para o mesmo subconjunto de instâncias a relaxação **RL_xyb_y1**, isto é, fazendo as variáveis x_{ij}^k com i ∈ C, j ∈ D ou i ∈ D, j ∈ C , k=1,...K, e y_i^k, i ∈ C ∪ $\{d-1, d\}$ binárias e fixando y_1^1 = 1. Mas, na tabela 5.6 percebe-se que os limites obtidos são parecidos aos obtidos na **RL_xb** e na **RL_xb_y1** acrescentando ainda o facto de demorar muito mais tempo para obter estes mesmos resultados. Assim sendo, conclui-se que todos esses testes realizados com os modelos SR-MTZ e SR-MFA não produziram melhores limites inferiores para além de consumirem maior tempo computacional. Portanto, a **RL_xb_y1** continua a ser a relaxação que conseguiu melhores resultados em menor tempo.

Tabela 5.6: Valores obtidos nos testes **RL_xyb_y1** realizados com os modelos SR-MTZ e SR-MFA

			SR-MTZ		**SR-MFA**	
		VEL	**RL_xyb_y1**	**Tempo**	**RL_xyb_y1**	**Tempo**
DadosP	2 circuitos	7	102.68	174.12	278.92	838.85
r201	2 circuitos	9	83.25	11.46	239.73	739.79
pr03_15	3 circuitos	11	97.01	1840.89	317.07	90014.48
pr03_30	3 circuitos	11	97.01	1891.82	368.28	90006.45
Média	2 circuitos		92.97	92.79	259.32	789.32
	3 circuitos		97.01	1866.36	342.67	90010.47

Para concluir a experiência computacional, considerou-se o modelo SR-MFA e acrescentaram-se as desigualdades válidas propostas em 3.5.2. a RL_xb_y1. Estas desigualdades foram adicionadas apenas ao modelo SR-MFA por ser o modelo que produziu melhores limites inferiores em melhor tempo nos testes anteriores.

A coluna **RL_xb_y1_u** corresponde a testar a **RL_xb_y1** adicionando a desigualdade $u_{ij} \geq \sum_{k=1}^{K} \left(s_i + t_{ij} \times x_{ij}^k\right)$, i,j ∈ C. Na coluna **RL_xb_y1_x** encontram-se os resultados dos testes acrescentando a desigualdade $x_{ij}^k + x_{ji}^k \leq 1$, i,j ∈ C, k=1,...,K. Por fim, fizeram-se os testes incluindo as duas desigualdades no modelo, **RL_xb_y1_ux**.

Com base na tabela 5.7, observa-se que praticamente não há diferença entre os limites obtidos. Mas considerando as pequenas diferenças, constata-se que em média a **RL_xb_y1_ux** fornece os melhores resultados. Aqui não é possível usar como critério o tempo computacional, porque em média o teste que demora menos tempo quando se consideram 2 circuitos, é o **RL_xb_y1_x**. E quando se trata de 3 circuitos é o **RL_xb_y1_u**.

Como o objetivo era obter melhores limites possíveis e melhores soluções, para servir de base para as heurísticas, escolhe-se a relaxação **RL_xb_y1_ux** para o modelo SR-MFA. Pois, para além de obter na maioria dos casos o melhor limite, garante que haja fluxo positivo em todos os arcos da solução e que existam menos subcircuitos, fazendo com que as soluções estejam mais próximas de uma solução inteira.

Tabela 5.7: Valores obtidos nos testes incluindo as desigualdades válidas realizados com o modelo SR-MFA

			SR-MFA					
		VEL	**RL_xb_y1_u**	**Tempo**	**RL_xb_y1_x**	**Tempo**	**RL_xb_y1_ux**	**Tempo**
Dados_p	2 circuitos	7	278.98	83.40	278.92	96.84	278.98	96.07
		9	248.11	57.83	248.07	93.34	248.11	102.65
		11	227.01	81.73	226.99	91.86	227.01	96.27
	3 circuitos	7	290.47	131.06	290.42	157.99	290.47	196.43
		9	258.81	138.51	258.77	154.59	258.81	157.98
		11	237.17	164.08	237.14	179.16	237.17	194.61
r201	2 circuitos	7	269.53	217.16	269.90	188.58	269.90	151.98
		9	239.73	88.23	240.04	92.68	240.05	103.01
		11	220.29	118.09	220.56	104.62	220.56	119.51
	3 circuitos	7	279.15	589.72	279.49	749.26	279.50	774.06
		9	249.15	426.51	249.46	341.90	249.46	442.92
		11	229.34	716.55	229.61	1345.97	229.61	489.49
pr03_15	2 circuitos	7	358.88	476.54	358.89	542.58	358.91	769.21
		9	325.48	582.17	325.49	559.38	325.50	693.25
		11	302.23	480.28	302.24	531.37	302.25	597.19
	3 circuitos	7	371.20	1325.64	371.21	2335.20	371.23	1227.19
		9	336.88	1675.38	336.89	1314.49	336.90	1677.92
		11	312.98	1748.31	312.99	1082.98	313.00	1944.21
pr03_30	2 circuitos	7	408.84	701.85	408.84	569.92	408.87	805.53
		9	375.26	591.63	375.26	625.09	375.28	640.27
		11	351.87	612.41	351.87	543.99	351.89	633.25
	3 circuitos	7	421.09	1509.56	421.09	1692.04	421.12	2049.76
		9	386.60	1700.23	386.60	1752.81	386.63	2615.94
		11	362.58	1805.69	362.58	2291.49	362.60	1815.26
Média	2 circuitos		300.52	340.94	300.59	336.69	300.61	400.68
	3 circuitos		311.29	994.27	311.35	1116.49	311.37	1132.15

Abaixo apresenta-se uma figura onde se pode visualizar as relaxações lineares propostas e testadas.

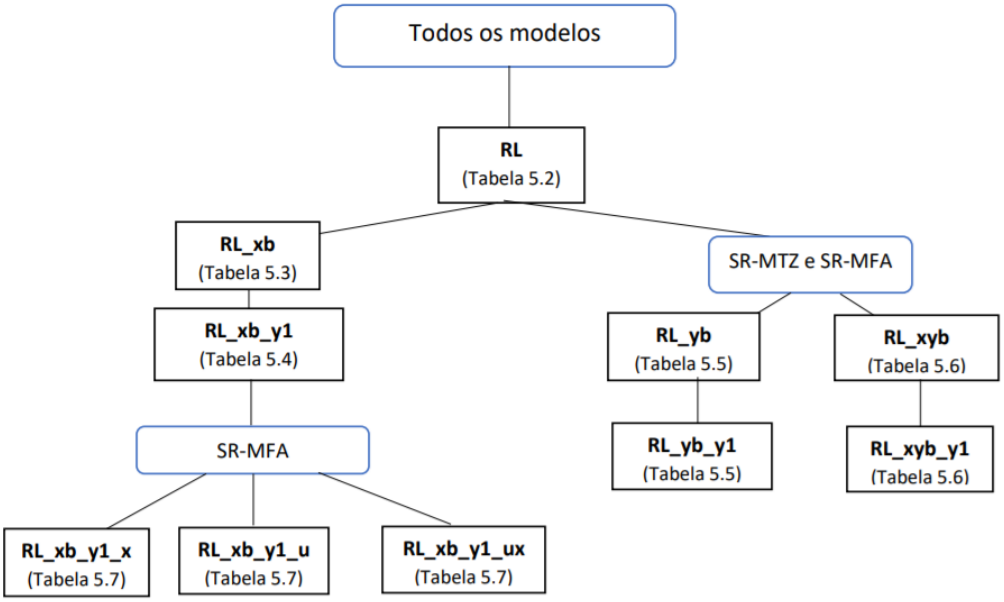

Figura 5.3: Esquema de representação das Relaxações Lineares

5.3 Resultados da Heurística

No capítulo 4, referiu-se que as heurísticas forneciam soluções admissíveis para o problema, e que o valor destas soluções constituem limites superiores. Portanto, tendo já os valores das relaxações lineares espera-se que os limites superiores que venham a ser obtidos pelas heurísticas não sejam muito elevados, de forma a estabelecer bons intervalos para o valor ótimo de cada instância.

Antes de testar as heurísticas realizaram-se testes computacionais implementando em IBM ILOG CPLEX *Optimization Studio* 12.7.0 os modelos SR-MTZ e SR-MFA e resolvendo os modelos PLIM (Programação Linear Inteira Mista). Estabeleceu-se um tempo máximo de 72 horas, para resolver todas as instâncias. Ao fim destas 72 horas não se conseguiu nenhuma solução admissível. Era expectável que isso acontecesse com o modelo SR-MFA mas, esperava-se conseguir alguma solução com o modelo SR-MTZ sabendo que é o modelo com menos variáveis e portanto, poderia ser mais fácil de resolver.

Perante a dificuldade em encontrar soluções admissíveis, experimentou-se alterar as opções que o *solver* utiliza por defeito na resolução do problema, nomeadamente no *branch-and-bound*. Uma das opções que foi acrescentada foi o *strong branching*, que faz uma seleção de variáveis baseada numa resolução parcial de um certo número de subproblemas com ramificações para ver qual a ramificação é mais promissora, com o fim de encontrar soluções admissíveis mais rapidamente.

Como não foi possível obter soluções admissíveis apenas através dos modelos, usando um tempo computacional razoável, passou-se à experiência com as heurísticas.

Nas heurísticas 1 e 2, estabeleceu-se um limite máximo de 12 horas para resolver os problemas do passo 3. E na heurística 3, o limite de tempo estabelecido para o passo 2 é o mesmo.

Relembrando o procedimento das heurísticas, tem-se que no passo 1 se resolve o problema com as variáveis x_{ij}^k com i $\in$ C, j $\in$ D ou i $\in$ D, j $\in$ C, k=1,...K binárias e as restantes relaxadas, que corresponde ao teste **RL_xb_y1** para o modelo SR-MTZ. E para o modelo SR-MFA se resolve o mesmo problema acrescentando as duas desigualdades válidas, **RL_xb_y1_ux**, mencionadas no capítulo 3.

Considerem-se as soluções obtidas no passo 1 das heurísticas 1 e 2. Para executar o passo 2, que é igual nas duas heurísticas, foi necessário escolher as variáveis y_i^k, i $\in$ C, k=1,...,K a serem fixas. Neste caso específico, decidiu-se que para os casos de 2 circuitos, fixavam-se as variáveis que tivessem valor superior ou igual a 0.8 (α = 0.8). E para os casos de 3 circuitos, seriam as variáveis com valor superior ou igual a 0.9 (α = 0.9).

Nas tabelas abaixo encontra-se na coluna **%ys fixos** a percentagem das variáveis y_i^k, i $\in$ C, k=1,...,K fixas no passo 2 das heurísticas 1 e 2, para um subconjunto de instâncias. Na coluna **V.da solução** pode ver-se o valor das soluções obtidas ao resolver o problema correspondente ao passo 2 das heurísticas 1 e 2. Isto é, o problema onde as variáveis x_{ij}^k com i $\in$ C, j $\in$ D ou i $\in$ D, j $\in$ C, k=1,...K e y_i^k, i $\in$ C, k=1,...,K são binárias e algumas variáveis y_i^k, i $\in$ C, k=1,...,K são fixas a 1.

Note-se que não é possível estabelecer uma relação entre o valor desta solução e o valor ótimo de SROP. Por um lado, existem variáveis binárias que foram consideradas contínuas e por outro fixou-se o valor de um subconjunto de variáveis.

Tabela 5.8: Modelo SR-MTZ: valor da solução obtida no passo 2 das heurísticas 1 e 2

Modelo SR-MTZ					
		VEL	**%ys fixos**	**V.da solução**	**Tempo**
DadosP	2 circuitos	7	47.28%	328.30	12h
r201	2 circuitos	9	47.80%	285.91	12h
pr03_15	3 circuitos	11	51.66%	482.38	12h
pr03_30	3 circuitos	11	50.92%	487.46	12h

Tabela 5.9: Modelo SR-MFA: valor da solução obtida no passo 2 das heurísticas 1 e 2

Modelo SR-MFA					
		VEL	**%ys fixos**	**V.da solução**	**Tempo**
DadosP	2 circuitos	7	59.72%	575.46	12h
r201	2 circuitos	9	58.07%	282.54	12h
pr03_15	3 circuitos	11	40.07%	508.72	12h
pr03_30	3 circuitos	11	42.37%	616.22	12h

No caso das heurísticas 1 e 2, apenas no passo 2 já se gastaram 12 horas. Para completá-las ainda é preciso fixar as variáveis x_{ij}^k = 1 com $(i,j) \in A^k$, k=1,...,K, e correr novamente o modelo utilizando no máximo 12 horas. Para a fixação destas variáveis x_{ij}^k = 1 com $(i,j) \in A^k$, k=1,...,K, estabeleceu-se que as variáveis x_{ij}^k com valor superior ou igual a β = 0.95 seriam fixas a 1. Pode-se questionar o porquê das 12 horas e não um tempo menor. Isto é porque, experimentou-se deixar algumas instâncias a correr por menos horas nos dois modelos e não se obtiveram soluções admissíveis para o modelo em questão.

No passo 3 da heurística 1, resolve-se o problema inteiro (PLIM) com as variáveis x_{ij}^k =1 $(i,j) \in A^k$, k=1,...,K, tal que $x_{ij}^k \geq 0.95$, partindo da solução obtida no passo 2. Utilizando o tempo máximo de 12 horas, só foi possível obter soluções admissíveis para uma das 4 instâncias consideradas em ambos os modelos. Na tabela 5.10 encontra-se os valores obtidos. Por só conseguir uma solução admissível nos dois modelos, escolheu-se não executar a heurística 1 para as restantes instâncias.

Tabela 5.10: Valores obtidos no passo 3 da heurística 1

Heurística 1						
			VEL	**%x fixos**	**V.da solução**	**Tempo**
SR-MTZ	**r201**	2 circuitos	9	32.86%	596.80	12h
SR-MFA	**r201**	2 circuitos	9	35.68%	497.32	12h

Já no passo 3 da heurística 2, foi possível encontrar soluções para as 4 instâncias somente com o modelo SR-MFA. Com o modelo SR-MTZ não se conseguiu nenhuma solução nas 12 horas. Mas é importante lembrar que estas soluções não são admissíveis para o problema pois as variáveis δ_i^h, i $\in$ C, h=1,...,$|H|$ não são inteiras, e portanto as restrições das janelas temporais não são respeitadas. A tabela 5.11 apresenta os valores obtidos.

Tabela 5.11: Valores obtidos no passo 3 da heurística 2

Heurística 2					
		SR-MFA			
		VEL	**%x fixos**	**V.da solução**	**Tempo**
DadosP	2 circuitos	7	6.25%	506.84	12h
r201	2 circuitos	9	35.68%	278.60	12h
pr03_15	3 circuitos	11	4.39%	433.90	12h
pr03_30	3 circuitos	11	15.55%	447.54	12h

Como foi referido, as soluções obtidas não eram admissíveis por isso, foi necessário aplicar o descodificador às soluções obtidas no passo 3. Ao aplicar o descodificador, todas as soluções obtidas concluíram os circuitos antes dos 14 dias e portanto são admissíveis. A tabela 5.12 mostra os valores obtidos no passo 4.

Para além de só conseguir soluções admissíveis com o modelo SR-MFA, os valores obtidos pela heurística 2 revelam-se piores que os valores obtidos pela heurística 3 (apresentados nas páginas seguintes). Sendo assim, optou-se por não executar a heurística 2 para as outras instâncias.

Tabela 5.12: Valores obtidos no passo 4 da heurística 2

Heurística 2					
		SR-MFA			
		VEL	**%x fixos**	**V.da solução**	**Tempo**
DadosP	2 circuitos	7	6.25%	748.55	12h
r201	2 circuitos	9	35.68%	507.18	12h
pr03_15	3 circuitos	11	4.39%	714.01	12h
pr03_30	3 circuitos	11	15.55%	790.90	12h

O passo 1 da heurística 3 é idêntica às outras, ou seja, a partir das soluções da **RL_xb_y1** para o modelo SR-MTZ e das soluções da **RL_xb_y1_ux** para o modelo SR-MFA, fixam-se as variáveis y_i^k, i $\in$ C, k=1,...,K com os mesmos critérios utilizados nas heurísticas 1 e 2.

Na tabela 5.13 encontra-se a percentagem das variáveis y_i^k, i $\in$ C, k=1,...,K que foram fixas a 1 para cada instância em cada modelo. Pelos valores consegue-se perceber que o número de variáveis fixas no modelo SR-MTZ é muito superior relativamente ao número de variáveis fixas no modelo SR-MFA. Isto pode fazer com que, as soluções obtidas ao resolver o modelo com as variáveis x_{ij}^k, $(i,j) \in A^k$, k=1,...,K e y_i^k, i $\in$ C, k=1,...,K binárias, sejam muito distintas também.

Tabela 5.13: Percentagem das variáveis y_i^k, i ∈ C, k=1,...,K fixadas a 1 no passo 2 da heurística 3

			Modelo SR-MTZ	**Modelo SR-MFA**
		VEL	**% y fixos**	**% y fixos**
DadosP	2 circuitos	7	47.28%	12.78%
		9	50.80%	13.97%
		11	48.37%	13.04%
	3 circuitos	7	46.49%	5.45%
		9	49.44%	8.56%
		11	46.24%	7.87%
r201	2 circuitos	7	50.27%	17.51%
		9	47.80%	16.11%
		11	46.99%	17.22%
	3 circuitos	7	43.39%	8.49%
		9	46.91%	10.05%
		11	47.62%	9.59%
pr03_15	2 circuitos	7	42.53%	9.56%
		9	49.27%	8.23%
		11	50.37%	7.72%
	3 circuitos	7	50.91%	4.55%
		9	49.09%	5.67%
		11	51.66%	4.34%
pr03_30	2 circuitos	7	49.08%	10.70%
		9	48.91%	8.42%
		11	49.82%	8.39%
	3 circuitos	7	49.46%	5.64%
		9	49.28%	6.83%
		11	50.92%	5.99%

Na tabela 5.14 encontram-se os resultados do teste citado no passo 2 da heurística 3, ou seja, os resultados obtidos a partir da fixação das variáveis y_i^k e da resolução do problema em que as variáveis y_i^k, i ∈ C, k=1,...,K e as variáveis x_{ij}^k, $(i,j) \in A^k$, k=1,...,K são binárias e as variáveis δ_i^h, i ∈ C, h=1,...,$|H|$ são contínuas. Para estes testes também foi utilizado um limite máximo de 12 horas. De notar que estas soluções obtidas no passo 2, não são admissíveis para o problema pois nem todas as variáveis δ_i^h, i ∈ C, h=1,...,$|H|$ são admissíveis. Pode-se ver que ao fim das 12 horas as únicas instâncias para as quais não se consegue nenhuma solução em nenhum dos modelos são dadosP com 3 circuitos e **VEL** = 9 e pr03.tsphs com 3 circuitos e **VEL** = 9. É curioso, como isso acontece para o mesmo número de circuitos e para a mesma velocidade do navio, em cada uma das instâncias. Relativamente à instância pr03.tsphs, é compreensível que ao fim do tempo limite não se consiga uma solução, pois trata-se da instância com maior número de estações de pesca. Já a instância dadosP tem apenas 95 estações e portanto, pode ser que a percentagem de variáveis fixas tenha influenciado a resolução do problema. Consultando a tabela 5.13 pode-se constatar que a percentagem fixa nos 2 modelos no caso de 3 circuitos com **VEL** = 9 é superior aos outros cenários de 3 circuitos da mesma instância.

Como referido, nas soluções obtidas, nem todas as variáveis δ_i^h, i ∈ C, h=1,...,$|H|$ eram inteiras, fazendo com que fosse necessário passar para o passo 3 da heurística. Para implementar o descodificador utilizou-se o *software* **MATLAB** versão R2017b.

Tabela 5.14: Valores das soluções obtidas no passo 2 da heurística 3

			Soluções com ys e xs binários e ys fixos	
		VEL	**Modelo SR-MTZ**	**Modelo SR-MFA**
DadosP	2 circuitos	7		469.82
		9	429.93	
		11	545.20	
	3 circuitos	7	709.69	
		9	*	*
		11	577.15	
r201	2 circuitos	7	310.63	347.28
		9	285.91	292.80
		11	250.90	261.35
	3 circuitos	7	458.13	330.29
		9	343.54	342.07
		11	284.69	288.19
pr03_15	2 circuitos	7	513.70	
		9	533.55	446.37
		11		390.66
	3 circuitos	7	896.43	
		9	669.14	390.92
		11	482.38	
pr03_30	2 circuitos	7		702.77
		9	481.27	
		11	474.54	513.05
	3 circuitos	7	583.98	556.49
		9	*	*
		11	487.46	

Ao aplicar o descodificador às soluções obtidas no passo 2 da heurística 3, obtiveram-se as soluções cujos valores que se encontram presentes na tabela 5.15. Repare-se que não se consegue obter um limite superior para todas as instâncias, portanto será necessário realizar trocas de estações entre circuitos. De notar também que, há um aumento significativo no valor das soluções quando se passa do passo 2 para o passo 3.

Como não foi possível conseguir limites superiores para todas as instâncias, vai aplicar-se um algoritmo baseado em trocas de estações entre circuitos de forma a minimizar a distância percorrida, sem violar as restrições das janelas temporais. Sendo assim, será necessário aplicar o passo 4 da heurística 3.

Tabela 5.15: Valores das soluções obtidas no passo 3 da heurística 3 após aplicar o descodificador

			Soluções da Heurística 3	
		VEL	**Modelo SR-MTZ**	**Modelo SR-MFA**
DadosP	2 circuitos	7		-
		9	635.40	
		11	765.75	
	3 circuitos	7	951.08	
		9	*	*
		11	795.10	
r201	2 circuitos	7	494.64	571.41
		9	467.60	487.25
		11	408.80	422.00
	3 circuitos	7	696.15	531.64
		9	525.76	553.19
		11	429.39	458.45
pr03_15	2 circuitos	7	-	
		9	827.16	-
		11		637.02
	3 circuitos	7	-	
		9	1012.78	673.96
		11	762.22	-
pr03_30	2 circuitos	7		-
		9	-	
		11	772.21	
	3 circuitos	7	-	-
		9	*	*
		11	795.99	

Ao analisar as soluções obtidas no passo 2, constatou-se que havia um desequilíbrio visível no que respeita ao número de estações visitadas por cada circuito, para os circuitos das instâncias que se revelaram não admissíveis após aplicação do descodificador. Este desequilíbrio, provinha das fixações feitas para as variáveis y_i^k, i $\in$ C, k=1,...,K.

Para as instâncias em que não se conseguiram soluções admissíveis no passo 3, aplica-se o descodificador e verifica-se até onde as listas de estações com maior número de elementos correspondem a soluções admissíveis. De seguida, distribuem-se as estações restantes para as listas com menor número de elementos e aplica-se o descodificador às novas listas.

Obtiveram-se as seguintes soluções inteiras admissíveis:

Note-se que foi possível obter soluções inteiras admissíveis para todas as instâncias que ainda não

Tabela 5.16: Valores obtidos no passo 4 da heurística 3

			Soluções passo 4 da Heurística 3	
		VEL	**Modelo SR-MTZ**	**Modelo SR-MFA**
DadosP	**2 circuitos**	7		710.83
pr03_15	**2 circuitos**	7	885.30	746.06
		9		
	3 circuitos	7	1065.25	
pr03_30	**2 circuitos**	9	816.62	
		11		821.05
	3 circuitos	7	955.47	1127.33

tinham uma solução, exceto para a pr03.tsphs com 2 circuitos e **VEL** = 7. Neste caso específico, os circuitos não eram desequilibrados por isso, não era completamente visível como deveriam ser feitas as trocas de estações entre circuitos. Isto leva a crer que, as rotas estabelecidas pelas fixações eram más.

É importante citar também que, pelo facto destas instâncias terem muitas estações acaba por dificultar ainda mais os casos de 2 circuitos. E, como se considera que as variáveis δ_i^h, i $\in$ C, h=1,...,$|H|$ são maiores ou iguais a 0, o modelo não contabiliza os tempos de espera fazendo com que, a solução possa ter ligações entre estações ou entre estações e portos que obriguem a avançar as janelas temporais muito rapidamente quando se aplica descodificador. Sendo assim, torna-se mais difícil ainda conseguir concluir cada circuito antes dos 14 dias ($b = 14 \times 24$).

Conclui-se então que, com esta heurística conseguiu-se obter soluções inteiras admissíveis para a maioria das instâncias utilizadas nos testes.

5.4 Comparação dos Modelos

Após a obtenção dos resultados da experiência computacional realizada, pode-se fazer uma análise comparativa entre os modelos a nível da qualidade dos limites inferiores, da qualidade dos limites superiores e do tempo de CPU. Mas para fazer esta análise, é preciso perceber que, a forma como são definidas as variáveis e o número de restrições presentes no modelo, influenciam a rapidez do *software* em resolver os problemas. Apresenta-se novamente a tabela com o número de variáveis e restrições de cada modelo para cada instância.

Tabela 5.17: Comparação dos modelos relativamente ao número de variáveis e restrições

	Modelo SR-MTZ		**Modelo SR-MF**		**Modelo SR-MFA**	
	# Variáveis	**# Restrições**	**# Variáveis**	**# Restrições**	**# Variáveis**	**# Restrições**
DadosP_2	21391	20845	41044	20762	32114	11737
DadosP_3	31374	31029	60901	30952	43041	12902
r201_2	21422	20866	41092	20778	31972	11562
r201_3	31413	31058	60966	30974	42726	12542
pr03_2	43554	42734	84564	42602	65002	23002
pr03_3	64126	63750	125866	63622	86946	24422

O modelo SR-MTZ é o modelo com o menor número de variáveis e o modelo SR-MFA é o modelo com o menor número de restrições.

De uma forma geral, pode-se dizer que o modelo SR-MFA é o que produz melhores limites inferiores num tempo razoável, uma vez que produz limites muito melhores que o modelo SR-MTZ utilizando um tempo inferior ao tempo utilizado pelo modelo SR-MF.

A nível dos limites superiores, as conclusões são contrárias. O modelo SR-MTZ é o que produz melhores limites superiores, sendo que aqui não se coloca a questão do tempo computacional porque foi estabelecido o mesmo tempo limite para testar a heurística nos dois modelos.

O *gap* fornece informação acerca da distância existente entre as relaxações lineares e as soluções admissíveis obtidas. No caso do problema em estudo, como é um problema de minimização o *gap* é calculado da seguinte forma:

$gap = \frac{LS-LI}{LI}$, onde LI é o melhor limite inferior e LS o limite superior (solução admissível). Como foi visto antes, os melhores limites inferiores para o modelo SR-MTZ foram os obtidos pela **RL_xb_y1** e os melhores limites inferiores obtidos com o modelo SR-MFA foram com a **RL_xb_y1_ux**.

Apesar do modelo SR-MTZ fornecer melhores limites superiores, repare-se na tabela 5.18 que os seus *gaps* são maiores. Isto acontece porque, os limites inferiores obtidos pelo modelo SR-MTZ são muito baixos, em relação aos obtidos com o modelo SR-MFA.

Para concluir, sugere-se com base nos testes e nos procedimentos elaborados, que o modelo SR-MTZ é o melhor modelo para obter limites superiores e o modelo SR-MFA é o melhor para obter limites inferiores.

Tabela 5.18: Valores dos *Gaps* obtidos comparando os limites superiores com os melhores limites inferiores de cada modelo

			Gap	
		VEL	**Modelo SR-MTZ**	**Modelo SR-MFA**
DadosP	2 circuitos	7		1.55
		9	6.27	
		11	8.98	
	3 circuitos	7	7.20	
		9	*	*
		11	7.99	
r201	2 circuitos	7	4.07	1.12
		9	4.62	1.03
		11	4.59	0.91
	3 circuitos	7	5.37	0.90
		9	4.58	1.22
		11	4.13	1.00
pr03_15	2 circuitos	7	6.83	
		9	7.62	1.29
		11		
	3 circuitos	7	7.31	
		9	8.22	1.00
		11	6.86	
pr03_30	2 circuitos	7		-
		9	7.51	
		11	8.19	1.33
	3 circuitos	7	6.45	1.68
		9	*	*
		11	7.21	

Capítulo 6

Conclusões

O SROP (*Ship Route Optimization Problem*), é um problema de otimização que até agora foi pouco estudado e explorado na literatura mas, que pode ter aplicações reais muito úteis em algumas áreas, como por exemplo nas campanhas de bio-oceanografia e pesca. Na presente dissertação, apresentaram-se três modelos, SR-MTZ (Ship Routing-MTZ), SR-MF (Ship Routing-Multifluxo) e SR-MFA (Ship Routing-MultiFluxo Agregado), para resolver o problema.

A partir dos modelos propostos foram realizados testes com base na relaxação linear que permitiram obter limites inferiores para o valor ótimo do problema. Foram realizados testes considerando a relaxação de todas as variáveis inteiras e testes considerando a relaxação de apenas algumas variáveis (relaxação linear parcial). Os resultados obtidos mostraram que o modelo SR-MFA é o modelo que fornece melhores limites inferiores e que o modelo SR-MTZ é o que consome menos tempo computacional. De todos os testes realizados para obter limites inferiores, o que forneceu melhores resultados em menor tempo computacional foi a **RL_xb_y1**, isto é, a relaxação linear parcial onde as variáveis x_{ij}^k com i $\in$ C, j $\in$ D ou i $\in$ D, j $\in$ C , k=1,...K são binárias e as restantes são relaxadas, e a estação 1 é fixada no circuito 1 y_1^1 = 1.

Acerca do modelo SR-MF, concluiu-se que produzia sempre os mesmos limites que o modelo SR-MFA mas no entanto, o tempo computacional utilizado pelo mesmo era sempre superior ao utilizado pelos outros modelos. Com base nisto para obter soluções admissíveis, escolheu-se aplicar as heurísticas aos resultados obtidos pelos modelos SR-MFA e SR-MTZ.

Estudaram-se procedimentos que permitissem obter soluções inteiras admissíveis através dos modelos e utilizando um *solver* genérico. Desenvolveram-se heurísticas para obter soluções admissíveis utilizando os modelos estudados e um solver genérico. As heurísticas sugeridas baseiam-se em procedimentos muito simples e faseados que partem de soluções das relaxações obtidas para os modelos. Soluções estas, que são conseguidas através da fixação e relaxação de algumas variáveis. Inicialmente foram propostas três heurísticas mas, em duas delas o tempo gasto para realizar apenas um dos passos intermédios era muito elevado.

Optou-se pela heurística, baseada na fixação das variáveis y_i^k , i ∈ C, k=1,...,K e na resolução do problema considerando as variáveis y_i^k = 1, i ∈ C, k=1,...,K e x_{ij}^k, $(i, j) \in A^k$, k=1,...,K binárias e relaxando as variáveis δ_i^h, i ∈ C, h=1,...,$|H|$. Após a aplicação do descodificador aos resultados, conseguiu-se obter soluções admissíveis para quase todas as instâncias. Nos casos em que não se conseguiram soluções admissíveis, aplicou-se uma heurística de melhoramento que efetua algumas trocas de estações entre os circuitos resultantes, respeitando as restrições das janelas temporais.

Com base em todos os testes realizados e nos resultados obtidos, conclui-se que os melhores limites inferiores são conseguidos pelo modelo SR-MFA e os melhores limites superiores são conseguidos pelo modelo SR-MTZ. Também é importante referir que todo o trabalho desenvolvido nesta dissertação mostra a grande complexidade do problema relativamente à obtenção de soluções inteiras admissíveis.

Relativamente a trabalhos futuros, existem aspetos que podem ser melhorados. O primeiro aspeto que poderia ser melhorado, são os critérios usados para a fixação das variáveis. Como foi referido, as fixações feitas já impunham um certo desequilíbrio entre os circuitos. Então, mudando os critérios de forma a que estas fixações não sejam desequilibradas, pode ser que as soluções sejam diferentes e os circuitos obtidos não sejam tão desequilibrados. Outro facto que pode ser considerado, é a aplicação de pesquisa local para tentar otimizar as soluções obtidas a nível de distância percorrida ou de tempo de espera.

Referências

[1] http://antor.uantwerpen.be/instances-in-the-paper-a-memetic-algorithmfor-the-travelling-salesperson-problem-with-hotel-selection.

[2] K. Al-Hamad. Tabu search algorithm for ship routing and scheduling problem with time window. Ph.D.Thesis, Brunel University, London, 2006.

[3] K. Al-Hamad, M. Al-Ibrahim, and E. Al-Enezy. A genetic algorithm for ship routing and scheduling problem with time window. *American Journal of Operations Research*, 2(03):417, 2012.

[4] A. Baltz, M. El Ouali, G. Jäger, V. Sauerland, and A. Srivastav. Exact and heuristic algorithms for the travelling salesman problem with multiple time windows and hotel selection. *Journal of the Operational Research Society*, 66(4):615–626, 2015.

[5] M. Castro, K. Sörensen, P. Vansteenwegen, and P. Goos. A memetic algorithm for the travelling salesperson problem with hotel selection. *Computers & Operations Research*, 40(7):1716–1728, 2013.

[6] A. Divsalar, P. Vansteenwegen, and D. Cattrysse. A variable neighborhood search method for the orienteering problem with hotel selection. *International Journal of Production Economics*, 145(1):150–160, 2013.

[7] A. Divsalar, P. Vansteenwegen, K. Sörensen, and D. Cattrysse. A memetic algorithm for the orienteering problem with hotel selection. *European Journal of Operational Research*, 237(1):29–49, 2014.

[8] K. Fagerholt, G. Laporte, and I. Norstad. Reducing fuel emissions by optimizing speed on shipping routes. *Journal of the Operational Research Society*, 61(3):523–529, 2010.

[9] E. Jardim and P. J. Ribeiro Jr. Geostatistical assessment of sampling designs for portuguese bottom trawl surveys. *Fisheries Research*, 85(3):239–247, 2007.

[10] M. Mesquita, A. Murta, A. Paias, and L. Wise. Tsp with multiple time-windows and selective cities. In *International Conference on Computational Logistics*, pages 158–172. Springer, 2013.

[11] M. Mesquita, A. G. Murta, A. Paias, and L. Wise. A metaheuristic approach to fisheries survey route planning. *International Transactions in Operational Research*, 24(3):439–464, 2017.

[12] M. Mesquita and A. Paias. Optimizing fisheries survey routes by a hybrid metaheuristic based on benders'cuts. *submetido para publicação*, 2017.

[13] G. Romero, G. Durán, J. Marenco, and A. Weintraub. An approach for efficient ship routing. *International Transactions in Operational Research*, 20(6):767–794, 2013.

[14] J. Silva. Planeamento de rotas de distribuição. Dissertação de Mestrado em Estatística e Investigação Operacional, Faculdade de Ciências da Universidade de Lisboa, 2016. `http://hdl.handle.net/10451/24881`.

[15] P. Toth and D. Vigo. An overview of vehicle routing problems. In *The vehicle routing problem*, pages 1–26. SIAM, 2002.

[16] T. Tsiligirides. Heuristic methods applied to orienteering. *Journal of the Operational Research Society*, 35(9):797–809, 1984.

[17] P. Vansteenwegen, W. Souffriau, and K. Sörensen. The travelling salesperson problem with hotel selection. *Journal of the Operational Research Society*, 63(2):207–217, 2012.

[18] P. Vansteenwegen, W. Souffriau, and D. Van Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1–10, 2011.

[19] X. Xu, H. Yuan, M. Liptrott, and M. Trovati. Two phase heuristic algorithm for the multiple-travelling salesman problem. *Soft Computing*, pages 1–15, 2017.

Apêndice A

Implementação em MATLAB

A.1 Implementação das listas de estações

A.1.1 Listas de estações com 2 circuitos

```
clear all
[num,txt,raw]=xlsread('r201_2_11_xy_mod3');
c1=0;
c2=0;
xpe='xpe';  %1
xee='xee';  %2
xep='xep';  %3
wep='wep';  %4
wpe='wpe';  %5
uee='uee';  %6
%% criar circuito 1
x=length(num);
for i=1:x
    if( (strcmp(txt(i,1),xpe) || strcmp(txt(i,1),xee) || strcmp(txt(
        i,1),xep)) && (num(i,1)==1 ))
        c1=c1+1;
        for j=1:4
            circuito1(c1,j)=num(i,j);
            if(strcmp(txt(i,1),xpe))
                circuito1(c1,5)=1;
                for l=1:x
                    if((num(i,2:3)==num(l,2:3)) & (strcmp(txt(l,1),
                        wpe)))
                        circuito1(c1,6)=num(l,4);
                    end
                end
            end
            if(strcmp(txt(i,1),xee))
                circuito1(c1,5)=2;
                for l=1:x
                    if((num(i,2:3)==num(l,1:2)) & (strcmp(txt(l,1),
                        uee)))
                        circuito1(c1,6)=num(l,3);
                    end
```

```
32                  end
33              end
34              if(strcmp(txt(i,1),xep))
35                  circuito1(c1,5)=3;
36                  for l=1:x
37                      if((num(i,2:3)==num(1,2:3)) & (strcmp(txt(1,1),
                            wep)))
38                          circuito1(c1,6)=num(1,4);
39                      end
40                  end
41              end
42          end
43      end
44      %% criar circuito 2
45      if( (strcmp(txt(i,1),xpe) || strcmp(txt(i,1),xee) || strcmp(txt(
            i,1),xep)) && (num(i,1)==2 ))
46          c2=c2+1;
47          for j=1:4
48              circuito2(c2,j)=num(i,j);
49              if(strcmp(txt(i,1),xpe))
50                  circuito2(c2,5)=1;
51                  for l=1:x
52                      if((num(i,2:3)==num(1,2:3)) & (strcmp(txt(1,1),
                            wpe)))
53                          circuito2(c2,6)=num(1,4);
54                      end
55                  end
56              end
57              if(strcmp(txt(i,1),xee))
58                  circuito2(c2,5)=2;
59                  for l=1:x
60                      if((num(i,2:3)==num(1,1:2)) & (strcmp(txt(1,1),
                            uee)))
61                          circuito2(c2,6)=num(1,3);
62                      end
63                  end
64              end
65              if(strcmp(txt(i,1),xep))
66                  circuito2(c2,5)=3;
67                  for l=1:x
68                      if((num(i,2:3)==num(1,2:3)) & (strcmp(txt(1,1),
                            wep)))
69                          circuito2(c2,6)=num(1,4);
70                      end
71                  end
72              end
73          end
74      end
75  end
76  %% ordenar circuito por ordem crescente do tempo
77  y1=length(circuito1);
```

```
y2=length(circuito2);
for j=1:y1-1
    for i=1:y1-1
        if(circuito1(i+1,6)<circuito1(i,6))
            aux=circuito1(i+1,1:6);
            circuito1(i+1,1:6)=circuito1(i,1:6);
            circuito1(i,1:6)=aux;
        end
    end
end

for j=1:y2-1
    for i=1:y2-1
        if(circuito2(i+1,6)<circuito2(i,6))
            aux=circuito2(i+1,1:6);
            circuito2(i+1,1:6)=circuito2(i,1:6);
            circuito2(i,1:6)=aux;
        end
    end
end
%% criar as permutacoes
k=0;
for i=1:y1
    if(circuito1(i,5)==1)
        k=k+1;
        permutacao1(1,k)=circuito1(i,3);

    end
    if(circuito1(i,5)==2)
        k=k+1;
        permutacao1(1,k)=circuito1(i,3);
    end
end

k=0;
for i=1:y2
    if(circuito2(i,5)==1)
        k=k+1;
        permutacao2(1,k)=circuito2(i,3);

    end
    if(circuito2(i,5)==2)
        k=k+1;
        permutacao2(1,k)=circuito2(i,3);
    end
end
%% guardar dados das permutacoes
filename='r201_int3test_per.xls'
xlswrite(filename,permutacao1,'2_11','A1')
xlswrite(filename,permutacao2,'2_11','A2')
```

A.1.2 Listas de estações com 3 circuitos

```
clear all
clc
[num,txt,raw]=xlsread('r201_3_11_xy_mod3');
c1=0;
c2=0;
c3=0;
xpe='xpe';  %1
xee='xee';  %2
xep='xep';  %3
wep='wep';  %4
wpe='wpe';  %5
uee='uee';  %6
%% criar circuito 1
x=length(num);
for i=1:x
    if( (strcmp(txt(i,1),xpe) || strcmp(txt(i,1),xee) || strcmp(txt(
        i,1),xep)) && (num(i,1)==1 ))
        c1=c1+1;
        for j=1:4
            circuito1(c1,j)=num(i,j);
            if(strcmp(txt(i,1),xpe))
                circuito1(c1,5)=1;
                for l=1:x
                    if((num(i,2:3)==num(1,2:3)) & (strcmp(txt(1,1),
                        wpe)))
                        circuito1(c1,6)=num(1,4);
                    end
                end
            end
            if(strcmp(txt(i,1),xee))
                circuito1(c1,5)=2;
                for l=1:x
                    if((num(i,2:3)==num(1,1:2)) & (strcmp(txt(1,1),
                        uee)))
                        circuito1(c1,6)=num(1,3);
                    end
                end
            end
            if(strcmp(txt(i,1),xep))
                circuito1(c1,5)=3;
                for l=1:x
                    if((num(i,2:3)==num(1,2:3)) & (strcmp(txt(1,1),
                        wep)))
                        circuito1(c1,6)=num(1,4);
                    end
                end
            end
        end
    end
    %% criar circuito 2
```

```
47      if( (strcmp(txt(i,1),xpe) || strcmp(txt(i,1),xee) || strcmp(txt(
            i,1),xep)) && (num(i,1)==2 ))
48          c2=c2+1;
49          for j=1:4
50              circuito2(c2,j)=num(i,j);
51              if(strcmp(txt(i,1),xpe))
52                  circuito2(c2,5)=1;
53                  for l=1:x
54                      if((num(i,2:3)==num(1,2:3)) & (strcmp(txt(1,1),
                            wpe)))
55                          circuito2(c2,6)=num(1,4);
56                      end
57                  end
58              end
59              if(strcmp(txt(i,1),xee))
60                  circuito2(c2,5)=2;
61                  for l=1:x
62                      if((num(i,2:3)==num(1,1:2)) & (strcmp(txt(1,1),
                            uee)))
63                          circuito2(c2,6)=num(1,3);
64                      end
65                  end
66              end
67              if(strcmp(txt(i,1),xep))
68                  circuito2(c2,5)=3;
69                  for l=1:x
70                      if((num(i,2:3)==num(1,2:3)) & (strcmp(txt(1,1),
                            wep)))
71                          circuito2(c2,6)=num(1,4);
72                      end
73                  end
74              end
75          end
76      end
77
78      %% criar circuito 3
79       if( (strcmp(txt(i,1),xpe) || strcmp(txt(i,1),xee) || strcmp(txt
            (i,1),xep)) && (num(i,1)==3 ))
80          c3=c3+1;
81          for j=1:4
82              circuito3(c3,j)=num(i,j);
83              if(strcmp(txt(i,1),xpe))
84                  circuito3(c3,5)=1;
85                  for l=1:x
86                      if((num(i,2:3)==num(1,2:3)) & (strcmp(txt(1,1),
                            wpe)))
87                          circuito3(c3,6)=num(1,4);
88                      end
89                  end
90              end
91              if(strcmp(txt(i,1),xee))
```

```
92                  circuito3(c3,5)=2;
93                  for l=1:x
94                      if((num(i,2:3)==num(1,1:2)) & (strcmp(txt(l,1),
                            uee)))
95                          circuito3(c3,6)=num(l,3);
96                      end
97                  end
98              end
99              if(strcmp(txt(i,1),xep))
100                 circuito3(c3,5)=3;
101                 for l=1:x
102                     if((num(i,2:3)==num(1,2:3)) & (strcmp(txt(l,1),
                            wep)))
103                         circuito3(c3,6)=num(l,4);
104                     end
105                 end
106             end
107         end
108     end
109 end
110 %% ordenar circuito por ordem crescente do tempo
111 y1=length(circuito1);
112 y2=length(circuito2);
113 y3=length(circuito3);
114 for j=1:y1-1
115     for i=1:y1-1
116         if(circuito1(i+1,6)<circuito1(i,6))
117             aux=circuito1(i+1,1:6);
118             circuito1(i+1,1:6)=circuito1(i,1:6);
119             circuito1(i,1:6)=aux;
120         end
121     end
122 end
123
124 for j=1:y2-1
125     for i=1:y2-1
126         if(circuito2(i+1,6)<circuito2(i,6))
127             aux=circuito2(i+1,1:6);
128             circuito2(i+1,1:6)=circuito2(i,1:6);
129             circuito2(i,1:6)=aux;
130         end
131     end
132 end
133
134 for j=1:y3-1
135     for i=1:y3-1
136         if(circuito3(i+1,6)<circuito3(i,6))
137             aux=circuito3(i+1,1:6);
138             circuito3(i+1,1:6)=circuito3(i,1:6);
139             circuito3(i,1:6)=aux;
140         end
```

```
141     end
142 end
143 %%  criar as permutacoes
144 k=0;
145 for i=1:y1
146     if(circuito1(i,5)==1)
147         k=k+1;
148         permutacao1(1,k)=circuito1(i,3);
149
150     end
151     if(circuito1(i,5)==2)
152         k=k+1;
153         permutacao1(1,k)=circuito1(i,3);
154     end
155 end
156
157 k=0;
158 for i=1:y2
159     if(circuito2(i,5)==1)
160         k=k+1;
161         permutacao2(1,k)=circuito2(i,3);
162
163     end
164     if(circuito2(i,5)==2)
165         k=k+1;
166         permutacao2(1,k)=circuito2(i,3);
167     end
168 end
169
170 k=0;
171 for i=1:y3
172     if(circuito3(i,5)==1)
173         k=k+1;
174         permutacao3(1,k)=circuito3(i,3);
175
176     end
177     if(circuito3(i,5)==2)
178         k=k+1;
179         permutacao3(1,k)=circuito3(i,3);
180     end
181 end
182 %% guardar dados das permutacoes
183 filename='r201_int3test_per.xls'
184 xlswrite(filename,permutacao1,'3_11','A1')
185 xlswrite(filename,permutacao2,'3_11','A2')
186 xlswrite(filename,permutacao3,'3_11','A3')
```

A.2 Implementação do descodificador

```
function [objective_score,w1,w2,w3] = descodificador3(P,prof,TEE,TEP,
    TPE)

filename='Novos_circ_corrig.xlsx';
[num,txt,raw] = xlsread(filename,'pr30_2_7');
[nli,ncol]=size(num);
window=1;
pre=0;
f1=0;
f2=0;
f3=0;
temp=4;
conv=0.0714;
e =[7 31 55 79 103 127 151 175 199 223 247 271 295 319 160];
l =[18 42 66 90 114 138 162 186 210 234 258 282 306 330 168];

for i=1:nli
    n(i)=0;
    for j=1:ncol
        if (num(i,j)>0)
            n(i)=n(i)+1;
        end
    end
end

w1=zeros(1,P+n(1));
w2=zeros(1,P+n(2));
if(nli>2)
    w3=zeros(1,P+n(3));
end
%% aplicar descodificador a permutacao 1
for j=1:n(1)
    permutation(j)=num(1,j);
end
for p=n(1)+1:n(1)+P
    w1(p)=0;
end
cont=1;
station=permutation(1);
w1(cont)=w1(n(1)+P-1)+ temp + conv*TPE(P-1,station);
totaldist=conv*TPE(P-1,station);

if (w1(cont)>l(window))
    window=window+1;
end
if(w1(cont)<=e(window))
    w1(cont)=e(window);
end

```

```
for k=2:n(1)
    pre=permutation(k-1);
    station=permutation(k);
    if (station>pre)
        aux=station-1;
    else
        aux=station;
    end
    cont=cont+1;
    w1(cont)=w1(cont-1)+prof(pre)+conv*TEE(pre,aux);
    if(w1(cont)>=l(window))
        if(window==7)
            pmin=1;
            for p=2:P-1
                if(TEP(pre,pmin)>=TEP(pre,p))
                    pmin=p;
                end
            end
            w1(n(1)+pmin)=w1(cont-1)+prof(pre)+conv*TEP(pre,pmin);
            w1(cont)=w1(n(1)+pmin)+temp+conv*TPE(pmin,station);
            totaldist=totaldist+conv*TEP(pre,pmin)+conv*TPE(pmin,
                station);
        else
            totaldist=totaldist+conv*TEE(pre,aux);
        end
        window=window+1;
        if(w1(cont)>= e(window))
            while(w1(cont)>=l(window))
                window=window+1;
            end
        end
        if (w1(cont)<=e(window))
            w1(cont)=e(window);
        end
    else
        totaldist=totaldist+conv*TEE(pre,aux);
    end
end
w1(n(1)+P)=w1(cont)+prof(station)+conv*TEP(station,P-1);
totaldist=totaldist+conv*TEP(station,P-1);
f1=totaldist+w1(n(1)+P);

%% aplicar descodificador a permutacao 2
for j=1:n(2)
    permutation(j)=num(2,j);
end
for p=n(2)+1:n(2)+P
    w2(p)=0;
end
cont=1;
station=permutation(1);
```

```
99 w2(cont)=w2(n(2)+P-1)+conv*TPE(P-1,station);
100 totaldist=conv*TPE(P-1,station);
101
102 if (w2(cont)>l(window))
103     window=window+1;
104 end
105 if(w2(cont)<=e(window))
106     w2(cont)=e(window);
107 end
108
109 for k=2:n(2)
110     pre=permutation(k-1);
111     station=permutation(k);
112     cont=cont+1;
113     if (station>pre)
114         aux=station-1;
115     else
116         aux=station;
117     end
118     w2(cont)=w2(cont-1)+prof(pre)+conv*TEE(pre,aux);
119     if(w2(cont)>=l(window))
120         if(window==7)
121             pmin=1;
122             for p=2:P-1
123                 if(TEP(pre,pmin)>=TEP(pre,p))
124                     pmin=p;
125                 end
126             end
127             w2(n(2)+pmin)=w2(cont-1)+prof(pre)+conv*TEP(pre,pmin);
128             w2(cont)=w2(n(2)+pmin)+temp+conv*TPE(pmin,station);
129             totaldist=totaldist+conv*TEP(pre,pmin)+conv*TPE(pmin,
                    station);
130         else
131             totaldist=totaldist+conv*TEE(pre,aux);
132         end
133         window=window+1;
134         if(w2(cont)>= e(window))
135             while(w2(cont)>=l(window))
136                 window=window+1;
137             end
138         end
139         if (w2(cont)<=e(window))
140             w2(cont)=e(window);
141         end
142     else
143         totaldist=totaldist+conv*TEE(pre,aux);
144     end
145 end
146 w2(n(2)+P)=w2(cont)+prof(station)+conv*TEP(station,P-1);
147 totaldist=totaldist+conv*TEP(station,P-1);
148 f2=totaldist+w2(n(2)+P);
```

```
149
150 %% aplicar descodificador a permutacao 3
151 if (nli>2)
152     for j=1:n(3)
153         permutation(j)=num(3,j);
154     end
155     for p=n(3)+1:n(3)+P
156         w3(p)=0;
157     end
158     cont=1;
159     station=permutation(1);
160     w3(cont)=w3(n(3)+P-1)+conv*TPE(P-1,station);
161     totaldist=conv*TPE(P-1,station);
162
163     if (w3(cont)>l(window))
164         window=window+1;
165     end
166     if(w3(cont)<=e(window))
167         w3(cont)=e(window);
168     end
169
170     for k=2:n(3)
171         pre=permutation(k-1);
172         station=permutation(k);
173         cont=cont+1;
174         if (station>pre)
175             aux=station-1;
176         else
177             aux=station;
178         end
179         w3(cont)=w3(cont-1)+prof(pre)+conv*TEE(pre,aux);
180         if(w3(cont)>=l(window))
181             if(window==7)
182                 pmin=1;
183                 for p=2:P-1
184                     if(TEP(pre,pmin)>=TEP(pre,p))
185                         pmin=p;
186                     end
187                 end
188                 w3(n(3)+pmin)=w3(cont-1)+prof(pre)+conv*TEP(pre,pmin)
                        ;
189                 w3(cont)=w3(n(3)+pmin)+temp+conv*TPE(pmin,station);
190                 totaldist=totaldist+TEP(pre,pmin)+TPE(pmin,station);
191             else
192                 totaldist=totaldist+conv*TEE(pre,aux);
193             end
194             window=window+1;
195             if(w3(cont)>= e(window))
196                 while(w3(cont)>=l(window))
197                     window=window+1;
198                 end
```

```
199                 end
200                 if (w3(cont)<=e(window))
201                     w3(cont)=e(window);
202                 end
203             else
204                 totaldist=totaldist+conv*TEE(pre,aux);
205             end
206         end
207         w3(n(3)+P)=w3(cont)+prof(station)+conv*TEP(station,P-1);
208         totaldist=totaldist+conv*TEP(station,P-1);
209         f3=totaldist+w3(n(3)+P);
210 end
211 objective_score=f1+f2+f3;
212 end
```