UMA ABORDAGEM HEURÍSTICA PARA A SOLUÇÃO DE PROBLEMAS DE RECOBRIMENTO DE CONJUNTOS DE GRANDE PORTE, COM APLICAÇÃO À ALOCAÇÃO DE TRIPULAÇÕES PARA COMPANHIAS AÉREAS.

Ana Lúcia Gouveia Pimentel

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Abílio Pereira de Lucena, Ph.D.

Prof. Amaranto Lopes Pereira, Dr.Ing.

Prof. Nelson Maculan Filho, D.Habil.

Prof. Lucídio dos Anjos Formiga Cabral , D.Sc.

Prof. Claudio Thomas Bornstein, Dr.Rer.Nat.

Prof. Nair Maria Maia de Abreu, D.Sc.

Prof. Luiz Satoru Ochi, D.Sc.

RIO DE JANEIRO, RJ – BRASIL

MARÇO DE 2005

GOUVEIA PIMENTEL, ANA LÚCIA

Uma abordagem heurística para a solução de problemas de recobrimento de conjuntos de grande porte, com aplicação à alocação de tripulações para companhias aéreas [Rio de Janeiro] 2005

VIII, 91 p. 29,7 cm (COPPE/UFRJ, D.Sc., Engenharia de Sistemas e Computação, 2005)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1. Programação Inteira.

2. Modelos de Recobrimento de Conjuntos.

3. Métodos Heurísticos e de Branch-and-Bound.

I. COPPE/UFRJ II. Título (série).

A Deus,

a meus pais, Clóvis e Laélia (*in memoriam*),

e a meus irmãos e sobrinhos.

Agradecimentos

Ao professor Abílio Pereira de Lucena Filho, pela valiosa orientação, por todos os ensinamentos transmitidos e pelo incentivo ao desenvolvimento deste trabalho.

Ao professor Nelson Maculan Filho, também pela orientação e pelo apoio à minha formação.

Ao professor Amaranto Lopes Pereira, por ter despertado em mim o interesse pelo tema e pela realização do estágio de doutourado no LAAS (*Laboratoire d'Analyse et d'Architecture des Systèmes*), em Toulouse.

Ao Programa de Engenharia de Sistemas e Computação - PESC/COPPE/UFRJ, pelo curso de doutourado e à CAPES pelo apoio financeiro.

Aos professores Lucídio dos Anjos Formiga Cabral, Claudio Thomas Bornstein, Nair Maria Maia de Abreu e Luiz Satoru Ochi, pela participação na banca examinadora.

A Fernando Barreto, por ter me cedido a utilização do cluster de máquinas do CENPES, Petrobrás, para a realização dos experimentos computacionais.

Aos colegas do projeto da Rio Sul, Lucídio, Marcone e Roberto.

A Maria de Fátima Cruz Marques, pelo carinho e agradável convívio.

A Carlos Henrique Medeiros de Sabóia pelo companheirismo, compreensão e grande apoio à realização deste trabalho.

A Djalene Maria Rocha, pela grande amizade e incentivo.

Aos amigos que fiz no PESC: Rosa, Talita, Henrique, Prata, Élder, Ana Flávia, Luidi, Sérgio, Michele, Vítor, Elivelton, Jorge, Carlile, Shane, Manoel, Jurandir, Tibérius e Mara, pelo agradável convívio no laboratório.

Aos amigos que fiz no Rio de Janeiro: Lucineide, Fernando, Nei, Geórgia, Cristiane, Kamila e Patrícia; em Toulouse: Olga, Rodrigo, Rosely, Ilce, Walid, Souhir, Sílvia, Fernando, Paulo, Laura, Marcos, Luciana, Rafaella, Marco Antônio, Géraud e Cristiane; e no CEPEL: Luciano, Daniela, André, Débora, Wagner, Rachel, Vítor, Fábio, Pablo, Márcia, e em especial, Fernanda da Costa Serra pelo apoio e incentivo.

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

UMA ABORDAGEM HEURÍSTICA PARA A SOLUÇÃO DE PROBLEMAS DE RECOBRIMENTO DE CONJUNTOS DE GRANDE PORTE, COM APLICAÇÃO À ALOCAÇÃO DE TRIPULAÇÕES PARA COMPANHIAS AÉREAS.

Ana Lúcia Gouveia Pimentel

Março/2005

Orientadores: Abílio Pereira de Lucena
Amaranto Lopes Pereira
Nelson Maculan Filho

Programa: Engenharia de Sistemas e Computação

Este trabalho apresenta uma nova metodologia para a solução de problemas de recobrimento de conjuntos de grande porte, com aplicação ao problema de alocação de tripulações para companhias aéreas.

O problema de alocação de tripulações é definido e contextualizado, e são apresentados alguns métodos de solução existentes.

É proposta uma formulação que divide o processo de solução em duas etapas: geração das rotações e otimização. A abordagem escolhida envolve a prévia geração de um conjunto de rotações, e a partir deste, a busca de uma boa solução inteira.

Para a fase de geração das rotações foi desenvolvido um algoritmo paralelo, com base na meta-heurística GRASP. Na fase de otimização foi utilizado o modelo de recobrimento de conjuntos para o qual foi desenvolvido um algoritmo híbrido, baseado em técnicas de Branch-and-Bound.

Os algoritmos foram testados utilizando um problema real de uma companhia aérea nacional e os resultados obtidos são apresentados.

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

AN HEURISTIC APPROACH FOR LARGE SCALE SET COVERING PROBLEMS WITH APPLICATION TO AIRLINE CREW PAIRING

Ana Lúcia Gouveia Pimentel

March/2005

Advisors: Abílio Pereira de Lucena
Amaranto Lopes Pereira
Nelson Maculan Filho

Department: Computing and Systems Engineering

The present work proposes a new method for solving large scale set covering problems with application to the airline crew pairing problem.

The definition, context and main characteristics of the crew pairing problem are presented, as well as some existing solution methods.

A problem formulation is proposed, dividing the solution process into two phases: pairing generation and optimization. A solution approach is chosen: the pairing generation phase is conducted first, originating a subproblem of the global crew pairing problem; and the optimization process searches for a good integer solution within the subproblem generated.

A parallel algorithm, based on the meta-heuristic GRASP, is developed and tested in the pairings generation phase. In the second phase, a set covering based model is proposed and an hybrid algorithm based on a branch-and-bound approach developed.

The algorithms are tested using a real problem from a domestic airline carrier and the computational results presented.

Sumário

Introdução **1**

1 Descrição do problema **4**

1.1 Contexto do problema: o processo de alocação de recursos em companhias aéreas 4

1.1.1 Construção do quadro de horários 4

1.1.2 Alocação de equipamentos 5

1.1.3 Construção das rotações 6

1.1.4 Alocação dos tripulantes 10

2 Definição e formulação matemática **11**

2.1 Formulação da etapa de geração das rotações 12

2.2 Formulação da etapa de otimização 15

3 Métodos de solução existentes **20**

3.0.1 Solução exata de problemas de pequeno porte 20

3.0.2 Algoritmos de otimização por subconjuntos dos vôos 21

3.0.3 Algoritmos de otimização por subproblemas 25

3.0.4 Algoritmos do tipo "Branch-and-Cut" 31

3.0.5 Algoritmos de geração de colunas 32

3.0.6 Decomposição por Relaxação Lagrangeana 38

3.0.7 Algoritmos heurísticos específicos para o problema 38

4 Justificativas para a abordagem proposta **40**

5 Geração das rotações **42**

5.1 A meta-heurística GRASP e sua aplicação em problemas de otimização combinatória 43
5.2 Condições de viabilidade adotadas para as rotações 44
5.3 Critérios de qualidade de uma rotação e função de custo adotada . . . 46
5.4 Descrição do algoritmo 48
5.4.1 Procedimento de construção das rotações 49
5.4.2 O procedimento de comparação e armazenamento das rotações 55
5.4.3 O algorimo paralelo de geração das rotações 58

6 Otimização das rotações **61**
6.1 Obtenção do limite inferior 61
6.1.1 Algoritmo por Subproblemas - SPRINT 62
6.1.2 Algoritmo Primal-Dual por Subproblemas 62
6.1.3 Heurística para a obtenção de uma solução dual viável 64
6.2 Obtenção de soluções inteiras 65
6.2.1 Algoritmo de Rubin Restrito 66
6.3 Algoritmo Híbrido 68

7 Resultados Computacionais **75**
7.1 Resultados para o gerador 75
7.2 Resultados para o otimizador 76

Referências Bibliográficas **89**

Introdução

A redução de custos operacionais constitui atualmente uma meta essencial no processo de gestão das companhias aéreas. A alta competitividade imposta pela globalização dos mercados vem tornando ainda mais urgente esta tarefa, a ponto de fazê-la definir a própria sobrevivência das empresas.

Neste contexto, técnicas de Pesquisa Operacional vêm sendo utilizadas por companhias aéreas com uma intensidade sem igual em outros setores de atividade ecônomica.

São inúmeros os exemplos bem sucedidos de adoção de soluções envolvendo a Pesquisa Operacional em companhias aéreas.

Em 1997, a Delta Airlines, atualmente a maior empresa de transporte aéreo doméstico dos Estados Unidos, estimou que o sistema Coldstart de alocação de frota proporcionava uma economia de cerca de 100 milhões de dólares por ano [18]. O TRIP ("Trip Reevaluation and Improvement Program"), sistema utilizado pela American Airlines para alocação de tripulantes, conduziu a reduções de custo de mais de 20 milhões de dólares por ano, no período de 1987 a 1992 [3]. Na United Airlines, o sistema de alocação de tripulantes empregado gerou ganhos anuais estimados, em 1993, em cerca de 12 milhões de dólares em utilização do pessoal e cerca de 4 milhões de dólares em custos de hospedagem [15]. Outras grandes companhias, como a Northwest Airlines, a Federal Express e a Air France, também utilizam sistemas de otimização que têm levado a uma economia significativa em termos de redução de custos e aumento da utilização de recursos [18].

De outra parte, os problemas de alocação de recursos envolvidos na atividade de transporte aéreo constituem uma classe de problemas amplamente estudada no campo da programação matemática. Os problemas de alocação de aeronaves e tripulação podem ser representados perfeitamente por generalizações de modelos como

os de recobrimento e particionamento de conjuntos, cujos métodos de solução têm sido alvo de inúmeras pesquisas na área. Devido à generalidade desses modelos, a busca de novos métodos de solução para os mesmos representa, portanto, uma grande contribuição para o campo da programação matemática, com implicações em diversas outras áreas produtivas da sociedade.

Objetivos

Os objetivos do presente trabalho consistem em:

- desenvolver um novo algoritmo para o problema de geração de rotações, uma das etapas componentes do processo de alocação de tripulantes em companhias aéreas;
- apresentar uma contribuição ao estudo das técnicas de solução de problemas de programação inteira através do desenvolvimento e teste de um algoritmo híbrido para a solução de problemas de recobrimento de conjuntos de grande porte;

Metodologia

A metodologia a ser desenvolvida neste trabalho envolve as seguintes etapas:

- Revisão bibliográfica das abordagens correntes de solução para o problema de alocação de tripulantes em companhias aéreas;
- Desenvolvimento e implementação de um algoritmo para solução do problema de construção de rotações de vôos para empresas aéreas;
- Análise dos resultados obtidos para o caso de uma companhia aérea nacional.

Estrutura do texto

O presente texto foi estruturado da seguinte forma:

O primeiro capítulo consiste na apresentação do problema a ser enfocado neste trabalho, no qual é feita uma breve descrição do contexto no qual o problema se

insere e são destacadas algumas características relevantes do mesmo. No segundo capítulo é apresentada uma formulação matemática para o problema. No terceiro, são descritas algumas abordagens de solução existentes na literatura. No quarto capítulo é apresentada a motivação para a realização do presente trabalho. No quinto e sexto capítulos são descritos os algoritmos de solução desenvolvidos para a fase de construção das rotações e para a fase de otimização, respectivamente.

No sétimo capítulo são apresentados os resultados computacionais obtidos e, finalmente, no oitavo capítulo, as conclusões do trabalho.

Capítulo 1

Descrição do problema

O problema a ser enfocado neste trabalho é o de construção de "rotações de vôos" para empresas aéreas, chamado na literatura de "Crew Pairing Problem".

A construção de rotações consiste na fase inicial do processo de alocação de tripulantes. O objetivo é gerar as unidades de trabalho, denominadas de rotações ou "pairings", a serem atribuídas à tripulação. Antes de detalhar o problema envolvido nesta tarefa, será apresentada uma breve descrição da alocação de recursos em companhias aéreas.

1.1 Contexto do problema: o processo de alocação de recursos em companhias aéreas

Devido à sua complexidade, o planejamento da alocação de recursos em companhias aéreas é usualmente divido em quatro etapas [4]. Faremos a seguir uma breve apresentação destas etapas a fim de situar o problema aqui enfocado.

1.1.1 Construção do quadro de horários

Inicialmente, é estabelecido o conjunto de vôos a serem operados pela empresa e seus respectivos horários de saída e chegada. Nesta fase são considerados basicamente os aspectos relacionados com a demanda e as restrições referentes à quantidade de aeronaves disponíveis na empresa e à disponibilidade de operação nos aeroportos.

1.1.2 Alocação de equipamentos

Em seguida, é realizado o processo de alocação de equipamentos aos trechos de vôos previamente estabelecidos.

O problema consiste em, dada uma determinada malha de vôos referente a um dado período de operação da empresa e as diferentes frotas disponíveis, cada uma delas composta de aeronaves de um mesmo tipo, designar que vôos serão operados por que aeronaves.

Este problema é geralmente dividido em duas etapas: inicialmente é feita a alocação da frota, ou seja, é determinado o tipo de equipamento a ser utilizado em cada vôo, referenciado na literatura como "fleet assignment problem", e em seguida, são determinadas as sequências de vôos a serem realizadas por cada aeronave de uma dada frota, o que denomina-se "aircraft routing problem".

No problema de "fleet assignement" o objetivo é minimizar custos operacionais com aeronaves e possíveis perdas de passageiros ocasionadas por uma insuficiência de assentos ofertados ou ainda pelo não atendimento de algumas exigências da demanda.

Nesta fase são consideradas informações relativas à demanda em cada trecho de vôo e dados de capacidade e custo operacional das aeronaves em cada frota. A solução obtida deve conter, para cada frota, um conjunto de sequências de vôos passíveis de serem operadas por suas aeronaves.

As restrições do problema compreendem: a designação de apenas um tipo de frota para cada vôo, a cobertura de todos os vôos, as possibilidades de conexão entre segmentos de vôos (um vôo pode suceder um outro se partir do local de destino do anterior após um período mínimo de tempo), a utilização de limites superiores para o número de aeronaves em cada frota, e também a possibilidade de se efetuar a manutenção das aeronaves. Como as sequências de vôo ainda não são atribuídas, nesta fase, às aeronaves propriamente ditas, o que se garante são oportunidades de manutenção para aeronaves de um referido tipo, ou seja, durante um período de tempo suficiente para a realização da manutenção, existe uma ou mais aeronaves daquele tipo no solo em uma determinada estação de manutenção.

Na fase seguinte, chamada de "aircraft routing", o objetivo é construir rotas de vôos para cada aeronave de maneira a minimizar os custos envolvidos na operação.

A função objetivo pode variar segundo os interesses da empresa. Nos casos onde o número de aeronaves ainda não foi fixado, pode-se estabelecer como objetivo minimizar o número de aeronaves necessárias para cumprir a programação. Em uma outra abordagem pode-se minimizar os custos de manutenção ou maximizar algum fator referente à satisfação dos passageiros, como por exemplo, minimizar a mudança de aeronaves em alguns trajetos onde existe uma forte demanda entre os pontos de origem do primeiro trecho e destino do último, ou ainda referente a preferências da empresa, como balanceamento da utilização das aeronaves. As restrições adicionais nesta fase são relacionadas ao cumprimento da programação de manutenção para cada aeronave.

1.1.3 Construção das rotações

Na terceira etapa, é realizada a tarefa de construção de rotações, enfocada no presente trabalho.

Uma rotação ("pairing") consiste em um conjunto de trechos de vôos consecutivos e interligados , que tem, como pontos de partida e chegada, a base domiciliar do tripulante a ela designado, conforme ilustra a figura 1.1.

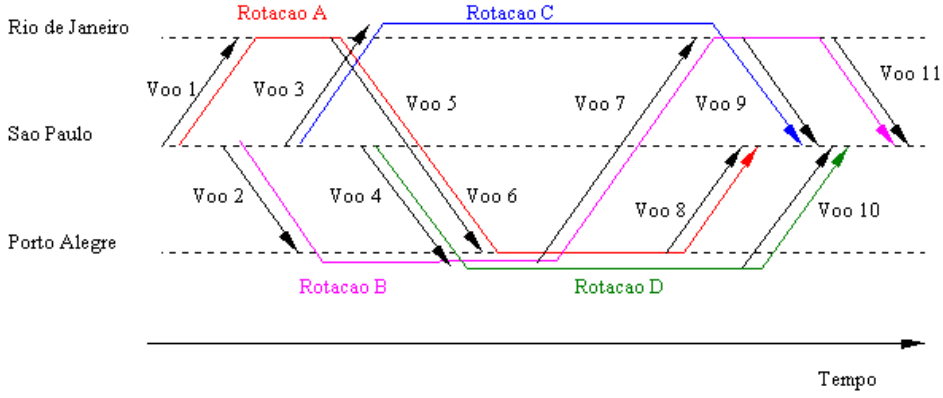


Figura 1.1: Rotações ou "pairings"

Nesta etapa, no entanto, as rotações ainda não estão alocadas aos tripulantes, sendo apenas relacionadas a uma determinada base. Uma rotação pode envolver trechos de vôos de diferentes trilhos, ou seja, realizados por diferentes aeronaves. Neste caso, um tempo de solo maior é requerido para que a tripulação realize a troca de equipamento.

A rotação deve atender a diversas restrições impostas pela regulamentação do aeronauta, tais como jornada máxima de trabalho contínuo, máximo de horas voadas por dia, períodos mínimos de repouso entre as jornadas, período máximo longe da base domiciliar, etc; bem como ainda a possíveis normas ou recomendações da empresa. Normalmente as rotações duram de um a cinco dias, devido ao período máximo permitido entre as folgas sociais [1] dos tripulantes. Uma rotação pode incluir também os denominados "deadheads", trechos de vôos nos quais o tripulante é transportado como passageiro, seja para realizar vôos a partir de uma outra localidade, seja para retornar à sua base no final da jornada. Rotações longas não são desejáveis porque, além de incorrerem em maiores custos com pernoites, dificultam o tratamento de problemas na fase de execução da programação, quando imprevistos impedem a realização de uma determinada rotação e os vôos contidos na mesma devem ser realocados em outras rotações.

Nesta etapa o trabalho consiste em estabelecer um conjunto de rotações que cubra todos o trechos de vôos operados pela empresa, de forma a minimizar uma função de custo adotada. O custo envolvido na operação das rotações compreeende diversos fatores, podendo ser expresso em termos de: utilização dos tripulantes, número de pernoites fora da base (despesas com hotel, alimentação, etc), e outros.

Algumas características peculiares do problema

Diversas características do problema de construção de rotações podem variar bastante de país a país ou mesmo entre empresas. Segundo Wedelin [4], entre estas características, as mais relevantes dizem respeito a: categorias de tripulação, tipo de frota, estrutura da malha de vôos, regulamentação, quadro de horários e estrutura de custos da empresa.

- Categoria de tripulação:

  A tripulação é composta de diversas categorias(comandantes, co-pilotos, comissários, etc) que obedecem a diferentes regulamentações, segundo a empresa ou o país que a sedia. Consequentemente, algumas importantes restrições, como por exemplo jornada máxima de trabalho, podem sofrer alterações, e em

[1] As folgas ditas sociais devem ter duração de no mínimo dois dias, e são oferecidas no domicílio dos tripulantes

decorrência disso, o problema poderá ter que ser tratado separadamente para cada uma das categorias consideradas.

- Tipo de frota:

  Cada tripulante, dependendo da sua categoria, é normalmente habilitado a voar em aeronaves de apenas um ou alguns dos diversos tipos de frota. O problema difere, portanto, também por tipo de frota. Ou seja, a decomposição aparece em dois níveis: o problema é inicialmente dividido em subproblemas por tipo de aeronave e cada um destes por sua vez é decomposto por categoria de tripulação.

- Estrutura da malha de vôos:

  A estrutura da malha de vôos de cada empresa acarreta impactos bastante relevantes no problema de construção de rotações. A este respeito pode-se observar diferenças significativas entre a América do Norte, a Europa e o Brasil.

  Na América do Norte, o sistema de transporte aéreo apresenta uma característica de forte concentração das rotas, constituindo os chamados sistemas "Hub and Spoke".

  Notadamente após a desregulamentação do setor, iniciada em 1978, as companhias de maior participação no mercado de transporte aéreo americano, procuraram, como forma de enfrentar a crescente concorrência, se "instalar" em alguns aeroportos nos quais estabeleceriam um ponto de transbordo para suas linhas. Iniciava-se um processo de reorganização do sistema, onde a antiga configuração de linhas diretas cedia lugar ao sistema de "Hub and Spoke". Dessa forma, alguns aeroportos com localização estratégica foram escolhidos para funcionarem como os chamados Hubs aeroportuários, onde, em pequenos intervalos de tempo, chegariam e sairiam uma grande quantidade de vôos, possibilitando diversas formas de conexão entre os mesmos, com tempos de espera bastante reduzidos. Este tipo de operação é capaz de proporcionar reduções significativas nos custos das empresas, e foi deste modo que algumas companhias aéreas americanas sobreviveram à abertura do mercado instituída pelo processo de desregulamentação. Em 1998, as participações das companhias

americanas em seus Hubs variava entre 75% a 95%. Ou seja, a grande maioria dos vôos operados nestes aeroportos passaram a ser das empresas que ali se "instalaram".

Como exemplo pode-se citar o Hub da American Airlines em Dallas, onde todo passageiro que aterrissa em um dos períodos de pico do aeroporto, dispõe, dentro de um período de duas horas subsequentes, de um leque de opções de mais de uma centena de diferentes destinos possíveis.

Pode-se concluir, portanto, que a estrutura da malha de vôos nos Estados Unidos é de tal forma que multiplica significativamente o número de diferentes possibilidades de rotações nas empresas que operam em Hubs. A tarefa de construção de rotação para estas companhias tornou-se, desse modo, ainda mais complexo.

Na Europa, tais sistemas ainda não predominam. A malha de vôos apresenta-se pouco concentrada, sendo quase a metade do tráfego aéreo total composto de vôos não regulares. A forte competitividade do sistema ferroviário, em especial dos trens de alto desempenho, aliada às curtas distâncias que separam os grandes centros limitaram o desenvolvimento de um sistema de Hubs nos moldes americanos.

As maiores participações são atualmente da Air France no aeroporto Charles de Gaulle, cerca de 45% da movimentação do aeroporto, e da British Airways no aeroporto de Heathrow, em torno de 55% [18].

No Brasil também ainda não dispomos de um sistema de "Hubs", sendo a configuração da malha composta basicamente de vôos diretos. Embora se possa perceber uma concentração de rotas envolvendo os aeroportos do Rio de Janeiro e São Paulo, esta se dá por uma questão de demanda, e não por terem estes aeroportos sido "escolhidos" como pontos de transbordo.

- Regulamentação:

  O problema de construção de rotações depende essencialmente do conjunto de regras estabelecidas aos tripulantes. Dessa forma, a atuação do órgão regulador da atividade no país, de possíveis acordos coletivos ou mesmo da política da

empresa exerce uma influência determinante no problema

1.1.4 Alocação dos tripulantes

Após a construção das rotações, segue-se a alocação das mesmas aos tripulantes. O objetivo básico nesta fase é designar a cada tripulante um conjunto de tarefas incluindo rotações, folgas, sobreaviso [2], reserva [3], treinamentos, férias, etc, de forma que sejam atendidas todas as restrições da regulamentação. A alocação pode ainda incluir uma minimização do número de tripulantes necessários, devendo, no entanto, preservar uma certa folga devido a possíveis imprevistos durante a execução da escala. É importante também que as rotações sejam atribuídas de forma equilibrada, nos casos onde a remuneração é determinada com base na quilometragem ou horas voadas.

[2] período no qual o tripulante deve estar à disposição da companhia para a realização de algum vôo não planejado

[3] período no qual o tripulante deve permanecer no aeroporto para a eventual realização de um vôo

Capítulo 2

Definição e formulação matemática

O problema de construção de rotações (PCR) consiste na obtenção de um conjunto de rotações de custo mínimo que cobre todos os trechos de vôos operados pela empresa.

A modelagem e solução do PCR é geralmente feita em duas etapas: geração das rotações e obtenção do conjunto de rotações de mínimo custo. Essa separação simplifica bastante a formulação do problema, pois considera as restrições impostas às rotações apenas na primeira fase, eliminando-as da formulação matemática do problema de otimização a ser resolvido, o que o torna significativamente mais simples.

No entanto, ao mesmo tempo que a separação facilita a modelagem do problema, ela cria outras dificuldades.

Para que essa separação seja equivalente a resolver o problema de forma integrada, seria necessário gerar um conjunto contendo todas as rotações viáveis possíveis, o que nem sempre é viável para um problema real. Para se ter uma idéia, em [8], Chu *et al.* citam uma estimativa da ordem de grandeza desse conjunto, realizada através de uma heurística de contagem, durante a geração das rotações para uma malha diária de MD-80 da American Airlines contendo 1000 trechos de vôos. A estimativa foi da ordem de 10^{12} a 10^{15}. Gerar explicitamente todas essas rotações, no curto espaço de tempo normalmente disponível para a obtenção de uma solução, claramente não é uma alternativa viável. Por outro lado, mesmo que fosse, teríamos que resolver, como veremos a seguir, um problema de partição/recobrimento de um porte acima do que é possível tratar atualmente.

Mesmo diante dessa dificuldade, na grande maioria dos trabalhos na literatura,

observa-se a separação da modelagem e solução do problema. Em alguns trabalhos as duas fases são realizadas separadamente, as rotações são geradas todas previamente e em seguida é feita a otimização [3], [8], [22], [26] e [27].

Em outros, as duas fases se alternam, como é o exemplo das abordagens por geração de colunas [1], [9], [10], [11], [16], [25], [33] e [34] onde se inicia a otimização com um determinado número de colunas (rotações) e, durante o processo, novas colunas são geradas e incluídas no modelo.

A estratégia adotada neste trabalho foi trabalhar com um número restrito de rotações e buscar uma solução de boa qualidade dentro desse conjunto previamente gerado. O PCR a ser otimizado será portanto restrito a este conjunto e será denotado aqui PCR restrito global. Mais adiante justificaremos nossa escolha.

A seguir são apresentadas as modelagens mais comumente aplicadas para cada uma das etapas de solução do PCR.

2.1 Formulação da etapa de geração das rotações

Para a fase de geração das rotações, são utilizadas, na maioria dos trabalhos, modelagens em grafos. Os dois tipos de grafos mais comumente adotados são: o grafo de trechos de vôos ("segment timeline network"), e o grafo de jornadas de serviço ("duty timeline network").

No primeiro caso, o grafo é construído a partir dos trechos de vôos. Cada vôo é representado por um nó no grafo, e os arcos conectam todos os pares de vôos entre os quais houver possibilidade de conexão.

O segundo tipo de grafo é construído a partir das jornadas de serviço, que são conjuntos de trechos de vôos a serem realizados durante a jornada de trabalho dos tripulantes. As jornadas de serviço devem obedecer a diversas restrições impostas pela regulamentação, que serão desritas mais adiante. De maneira análoga ao caso anterior, a cada jornada de serviço é associado um nó, e os arcos conectam todos os pares de jornadas entre as quais existe possibilidade de conexão.

Toda jornada de serviço corresponde a um caminho no grafo de trechos de vôos e toda rotação corresponde a um caminho no grafo de jornada de serviços. No entanto nem todos os caminhos nesses grafos podem ser considerados jornadas de serviço ou rotações, devido às restrições impostas a estes últimos.

No caso do grafo de jornadas de serviço, para que a modelagem não exclua nenhuma possibilidade de combinação entre os trechos de vôos, tem-se que gerar todas as possíveis jornadas. Como as jornadas tem pequena duração e ainda têm que satisfazer a diversas restrições, geralmente faz-se a enumeração completa das mesmas. O tamanho do grafo aumenta consideravelmente em relação ao primeiro caso.

Para ilustrar as modelagens descritas, considere o seguinte exemplo de malha de vôos:

Num. vôo	origem/destino	dia	horarios partida/chegada	
1	GIG - CGH	1	06:00	06:55
2	CGH - BSB	1	07:30	08:40
3	GIG - POA	2	08:00	10:15
4	POA - BSB	2	11:00	12:00
5	BSB - GIG	2	14:10	15:00
6	GIG - PLU	3	14:25	15:40
7	GIG - CGH	3	14:50	15:40
8	PLU - SSA	3	16:10	18:00
9	BSB - PLU	3	16:15	17:25
10	PLU - GIG	3	18:10	19:20
11	GIG - REC	4	09:20	12:10
12	REC - SSA	4	12:40	13:45
13	SSA - CGH	4	14:50	17:00
14	SSA - POA	4	15:00	18:50
15	POA - GIG	4	19:40	21:30
16	CGH - GIG	4	19:50	20:30

Tabela 2.1: Malha de vôos com base domiciliar no Rio de Janeiro (GIG)

A figura 2.1 mostra um grafo definido a partir dos trechos de vôos. Este tipo de grafo também pode ser utilizado para uma estratégia de solução integrada do PCR, como apresentado em [5], onde Ball e Roberts desenvolveram um um algoritmo para encontrar caminhos disjuntos em um grafo, que correspondem à uma solução para o problema de particionamento. No entanto esta abordagem funciona bem apenas para problemas de pequeno porte.

Na figura 2.2 é apresentado um grafo de jornadas de serviço. Observe que um mesmo trecho de vôo pode aparecer em mais de uma jornada de serviço. Neste caso portanto, não se deve "visitar" todos os nós como no grafo de trechos de vôos.

A modelagem escolhida neste trabalho foi a de grafos de jornadas de serviço. Essa modelagem simplifica o processo de geração dividindo o algoritmo em duas etapas

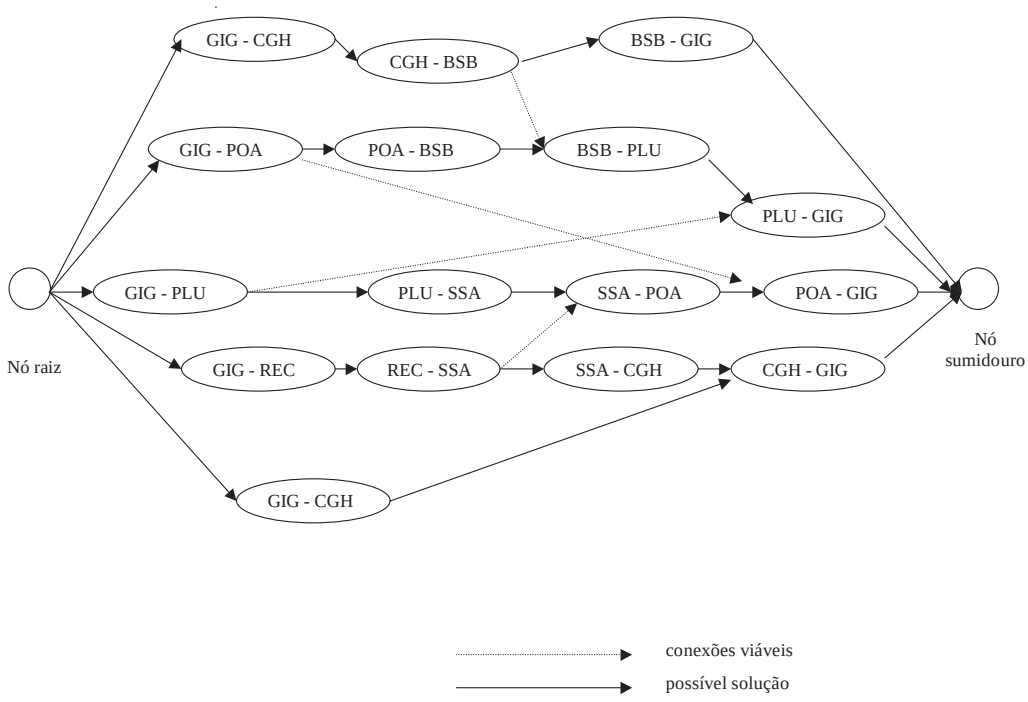


Figura 2.1: Grafo de trechos de vôos

distintas: geração das jornadas e geração das rotações. Dessa forma as restrições podem ser tratadas separadamente. A dificuldade na utilização deste tipo de grafo é que, para malhas grandes, se forem enumeradas todas as jornadas de serviço viáveis, o número de arcos pode chegar à ordem de 10^{10}, como apresentado mais adiante no capítulo que descreve o algoritmo utilizado nesta fase.

Esta característica dificulta a aplicação de algoritmos como os de caminhos mínimos, utilizados comumente nas abordagens por geração de colunas, onde os caminhos mínimos correspondem às rotações que apresentam menores custos reduzidos.

Na maioria dos trabalhos são utilizados algoritmos para encontrar caminhos em grafos. Mais recentemente Klabjan *et al.* [22] apresentaram uma heurística baseada na meta-heurística GRASP [30] para a geração das rotações, que contém princípios semelhantes aos da abordagem desenvolvida no presente trabalho.

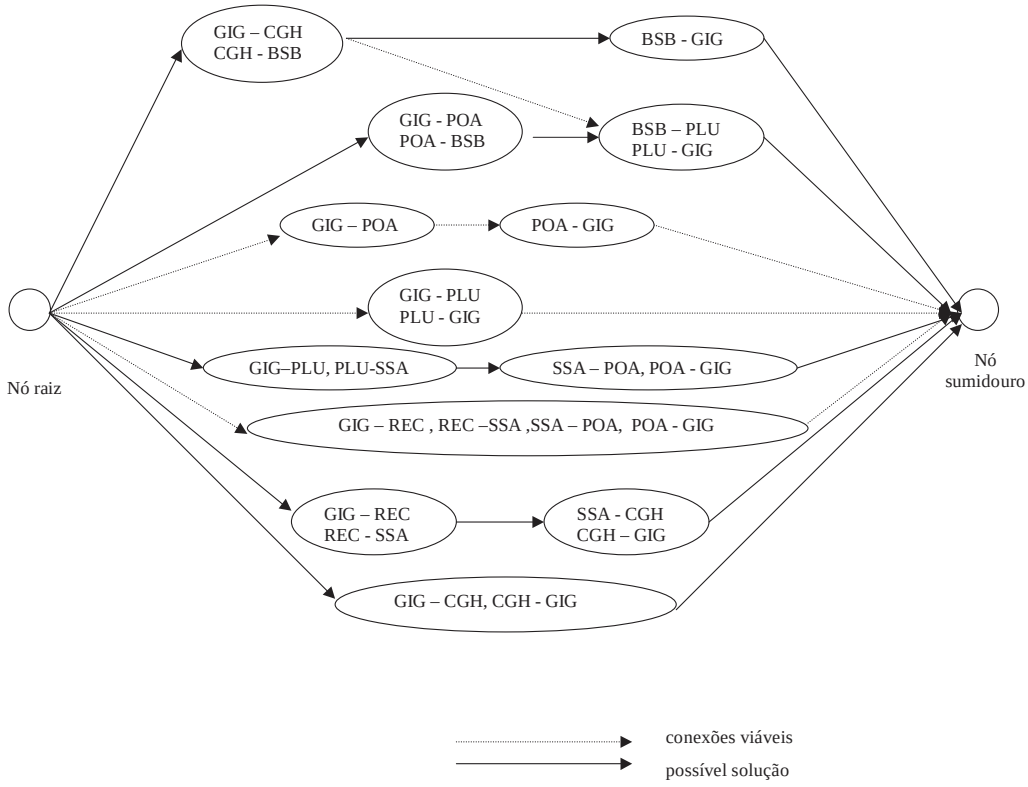


Figura 2.2: Grafo de jornadas de serviço

2.2 Formulação da etapa de otimização

Na fase de otimização, os modelos mais utilizados na literatura são: o particionamento de conjuntos [2], [11], [13], [17], [22], [26], [28] e [31], e o recobrimento de conjuntos [25], [36], onde neste último pode-se considerar a idéia de "deadheading" ou deslocamento de tripulantes como passageiros, conforme será detalhado mais adiante.

Dada à sua maior flexibilidade, foi escolhido o modelo de recobrimento de conjuntos generalizado para o método de solução proposto neste trabalho, o qual pode ser descrito da seguinte forma:

Seja $T = \{t_j : j \in J\}, J = \{1, 2, ..., n\}$, o conjunto de todas as rotações viáveis para o problema. Tais rotações apresentam custos $\{c_j : j \in J\}$, e devem cobrir o conjunto de trechos de vôo $F = \{f_i : i \in I\}, I = \{1, 2, ..., m\}$. Os vôos cobertos por

uma rotação $t_j \in T$ são denotados por $F_j = \{f_i : i \in I_j\}$, onde $I_j \subseteq I$ é o conjunto dos índices destes vôos. De forma análoga, as rotações que contêm o vôo $f_i \in T$, são denotadas por $T_i = \{t_j : j \in J_i\}$, onde $J_i \subseteq J$ é o conjunto dos índices destas rotações.

Rotações se originam em um conjunto $B = \{b_k : k \in K\}, K = \{1, 2, ..., r\}$, de bases. A base associada à rotação $t_j \in T$ é identificada por $b_{k_j} \in B$, e o tempo requerido para implementação desta rotação é denotado por h_{k_j}. Tripulantes da base b_k devem trabalhar, durante o horizonte de tempo considerado, um número total de horas dentro do intervalo $[\delta_k^l, \delta_k^u]$. O conjunto de rotações que têm por base b_k é denotado por $t^k = \left\{t_j : j \in J^k\right\}$, onde $J^k \subseteq J$ é o conjunto dos índices destas rotações.

Um modelo de recobrimento generalizado para o PCR é definido, a partir de um conjunto $\{x_j : j \in J\}$ de variáveis binárias 0-1 que controlam a inclusão $(x_j = 1)$ ou não $(x_j = 0)$ da rotação $t_j \in P$ em uma solução.

O modelo é descrito por:

$$\text{Minimizar} \quad z = \sum_{j \in J} c_j x_j \tag{1}$$

sujeito a:

$$\sum_{j \in J_i} x_j \geq 1, \qquad i \in I \tag{2}$$

$$\delta_k^l \leq \sum_{j \in J^k} h_{kj} x_j \leq \delta_k^u, \quad k \in K. \tag{3}$$

$$x_j \in \{0, 1\}, \qquad j \in J. \tag{4}$$

Cada linha do primeiro grupo de restrições representa um vôo operado pela empresa, e deve fazer parte de pelo menos uma das rotações da solução. No caso em que um vôo está contido em mais de uma rotação, um dos tripulantes realizará o vôo e os demais serão transportados como passageiros, operação denominada na literatura como "deadheading".

O segundo grupo de restrições refere-se à disponibilidade de tripulantes nas diferentes bases domiciliares consideradas no problema. O limite inferior adotado pode ser diferente de zero no caso em que a companhia deseje que não mais que um certo número de tripulantes em cada base seja designado a ficar em reserva, ou seja, um

volume mínimo de trabalho deve ser atribuído. O limite superior representa o número máximo de trabalho a ser atribuído a cada base, em função da disponibilidade de tripulantes.

Neste estudo, o modelo (1)-(4) foi adaptado de forma a melhor tratar a questão de "deadheadings". De (2) observamos que $\sum_{j \in J_i} x_j - 1$, para um dado $i \in I$, indica a sobrecobertura do vôo f_i. Se tomarmos d_i como sendo o custo unitário de sobrecobertura deste vôo, a função objetivo:

$$\begin{aligned} \text{Minimizar} \quad & z = \sum_{j \in J} c_j x_j + \sum_{i \in I} d_i \Big(\sum_{j \in J_i} x_j - 1 \Big) = \\ & = \sum_{j \in J} \Big(c_j + \sum_{i \in I_j} d_i \Big) x_j - \sum_{i \in I} d_i \end{aligned}$$

é suficientemente adequada para tratar "deadheadings" em vôos da companhia aérea onde os algoritmos aqui propostos serão testados.

Um modelo de partição generalizado para o PCR é obtido substituindo-se (2) por uma restrição de igualdade:

$$\text{Minimizar} \quad z = \sum_{j \in J} c_j x_j \qquad (5)$$

sujeito a:

$$\sum_{j \in J_i} x_j = 1, \qquad i \in I \qquad (6)$$

$$\delta_k^l \leq \sum_{j \in J^k} h_{kj} x_j \leq \delta_k^u, \quad k \in K. \qquad (7)$$

$$x_j \in \{0, 1\}, \qquad j \in J. \qquad (8)$$

sendo este modelo o mais utilizado na literatura.

Neste trabalho, no entanto, utilizamos o modelo (1)-(4), por ser mais fácil encontrar soluções viáveis para um problema de recobrimento e também pela facilidade, já mencionada, no tratamento da questão de "deadheadings".

Para uma melhor compreensão do modelo, considere o exemplo prático apresentado na tabela 2.1. Considerando-se apenas três das rotacões viáveis a partir do grafo 2.2, e apenas uma base associada ao problema (GIG). O modelo resultante seria como descrito em 2.2.

No que se refere à determinação da função de custos, observa-se, na maioria dos trabalhos nesta área, uma certa dificuldade na obtenção de uma expressão representativa dos mesmos, sendo adotada, em grande parte dos casos, uma função linear

$$
\begin{array}{lllllll}
\text{Minimizar} & z = c_1 x_1 + & c_2 x_2 & + c_3 x_3 & & \\
\text{sujeito a:} & & & & & \\
\text{(GIG-CGH)} & x_1 & + x_2 & & \geq 1 & \\
\text{(CGH-BSB)} & x_1 & + x_2 & & \geq 1 & \\
\text{(GIG-POA)} & & & x_3 & \geq 1 & \\
\text{(POA-BSB)} & & & x_3 & \geq 1 & \\
\text{(BSB-GIG)} & x_1 & & & \geq 1 & \\
\text{(BSB-PLU)} & & x_2 & + x_3 & \geq 1 & \\
\text{(PLU-GIG)} & & x_2 & + x_3 & \geq 1 &
\end{array}
$$

$$x_1,\, x_2,\, x_3 \geq 0$$

aproximativa. O custo de cada coluna depende mais fortemente dos intervalos entre os vôos, do que diretamente dos mesmos, já que os dias inativos, pernoites em hotéis ou períodos de inatividade durante a jornada, representam as parcelas de maior custo em uma rotação. Também deve ser considerada a forma de remuneração da tripulação, a qual varia de acordo com os contratos de trabalho adotados e a regulamentação vigente. Algumas companhias pagam um mínimo garantido de horas por dia, ou por mês; outras pagam em função do trabalho realizado, por quilometragem ou horas voadas. Além destes aspectos, a função objetivo do problema também pode variar de acordo com a estratégia e as necessidades da empresa.

Um dos desafios para a resolução do PCR utilizando um destes modelos é encontrar um método para gerar um número não explosivamente grande de colunas e que ainda assim contenha soluções de boa qualidade.

O outro desafio é conseguir obter soluções exatas (ou aproximadas, de boa qualidade) para as instâncias de recobrimento/particionamento generalizado resultante. Na literatura, as maiores instâncias resolvidas com garantia de otimalidade para modelos de partição envolviam não mais de 5 milhões de rotações explicitamente geradas, a partir de cerca de 1000 vôos.

Os problemas de recobrimento (PR) e particionamento (PP) são classificados

como NP-Árduos [21], ou seja, se enquadram em uma classe de problemas para os quais não se tem garantia da obtenção de soluções ótimas em tempo polinomial. Encontrar soluções ótimas para instâncias de grande porte destes problemas constitui, portanto, um desafio para algoritmos exatos. Os algoritmos desenvolvidos para este tipo de problema baseiam-se, em geral, em técnicas de enumeração implícita e podem envolver uma combinação de técnicas heurísticas, relaxações lineares, relaxação lagrangeana, planos de cortes, etc.

Capítulo 3

Métodos de solução existentes

Dada a dificuldade de se garantir otimalidade para um problema de construção de rotações não trivial, formulado como um PR ou PP (generalizados ou não), estritamente falando, abordagens baseadas na resolução destes modelos podem ser consideradas como heurísticas.

Desta forma, pode-se observar na literatura, um grupo maior de trabalhos, baseados em técnicas exatas mas que incorporam, na sua maioria, heurísticas, seja na fase de geração de rotações, seja no processo de busca da solução, e, em menor quantidade, alguns trabalhos onde foram desenvolvidas heurísticas específicas para o problema.

3.0.1 Solução exata de problemas de pequeno porte

Em alguns trabalhos desenvolvidos nesta área, quando o número de vôos a cobrir não é muito grande ou a duração das rotações é curta, foram consideradas todas as possíveis combinações viáveis de vôos para a geração das rotações. Para problemas deste tipo, a abordagem exata para os modelos de particionamento e recobrimento mostrou ser bastante eficaz.

Em [26], Marsten *et al.* apresentam um algoritmo que gera todas as possíveis rotações, começando por agrupar trechos de vôos em jornadas de serviço e em seguida enumerando todas as possíveis combinações de jornadas de serviço que resultam em rotações viáveis. O processo considera heurísticas para redução do número de linhas do problema através do agrupamento de alguns vôos.

Após a determinação de todas as rotações, o problema é modelado na forma de um particionamento, onde as linhas correspondem aos grupos de vôos construídos

na fase anterior. São considerados também alguns vôos extras que, obviamente, não fazem parte das restrições do particionamento. Para a solução do problema é aplicado o método "Branch-and-Bound" [14], com limites inferiores obtidos por relaxação linear. Essa abordagem de solução foi utilizada pela Flying Tiger e Pacific Southest Airways.

Em [27], Marsten e Shepardson apresentam uma comparação entre o método apresentado em [26] e um outro semelhante aplicado a uma instância da Continental Airlines, que também utiliza o método "Branch-and-Bound", só que com limites inferiores obtidos por relaxação lagrangeana e método do subgradiente. Para as quatro instâncias testadas o método que utiliza relaxação lagrangeana mostrou-se mais rápido e ambos obtiveram a solução ótima.

3.0.2 Algoritmos de otimização por subconjuntos dos vôos

Em diversos outros trabalhos, foram utilizados métodos exatos para a resolução do PR ou PP sem, no entanto, garantir que toda rotação viável seja candidata a entrar na solução. Uma das abordagens encontradas na literatura utiliza os métodos exatos para promover melhorias em uma solução inicial viável, otimizando subproblemas envolvendo subconjuntos dos vôos. Neste grupo destaca-se o algoritmo desenvolvido por Rubin [31].

O algoritmo de Rubin

A idéia básica do algoritmo de Rubin consiste em tentar melhorar uma dada solução viável para o PCR através de melhoras sistemáticas em "pequenas partes" da mesma.

Assuma que se tem em mãos um conjunto de rotações que constitui uma solução viável para o PCR. Assuma ainda que é selecionado, de alguma forma, um "pequeno" subconjunto destas rotações (em [31] são selecionadas apenas 3). Obviamente, as rotações escolhidas cobrem um certo subconjunto dos vôos considerados. Para este subconjunto de vôos, todas as rotações viáveis devem ser, então, explicitamente geradas, dando origem assim, a um "pequeno" subproblema de partição generalizada. Note que gerar todas essas rotações é possível diante do "pequeno" subconjunto de trechos de vôos considerado.

O subproblema é então resolvido de forma exata e, a partir da solução obtida, duas alternativas se apresentam.

Na primeira delas, quando o custo da solução do subproblema é menor que a soma dos custos das rotações que o geraram, a solução viável do PCR deve ser modificada, com a introdução das novas rotações e a remoção das antigas.

Alternativamente, quando não conseguimos melhorar a solução do PCR, nada é feito.

Diante de qualquer uma das duas alternativas, o procedimento é levado adiante, com a escolha de um novo subconjunto de rotações dentre as que compõem a solução do PCR atualmente em mãos. O processo é repetido até que um dado critério de parada seja satisfeito.

Como descrito acima, os passos fundamentais do algoritmo de Rubin são, em cada iteração, escolher um subconjunto conveniente de rotações e resolver um problema de partição generalizada formulado a partir dos vôos cobertos por aquelas rotações. Adiando a exposição de critérios para a escolha do subconjunto de rotações, vamos assumir, por enquanto, que estas rotações já foram escolhidas.

Seja $S \subseteq P$ um conjunto de rotações que formam uma solução viável para o PCR (como formulado em (5)-(8)), e denote por $S^o \subseteq S$ um subconjunto, convenientemente escolhido das rotações em S.

Os vôos cobertos por rotações em S^o constituem o conjunto $F^o \subseteq F$, enquanto $P^o \subseteq P$ denota o conjunto de todas as rotações viáveis constituídas unicamente de vôos em F^o.

Denote por $\overline{\delta}_k^l$, para cada $k \in K$, a diferença entre δ_k^l e a soma das horas trabalhadas por tripulantes baseados em b_k em rotações de $S \setminus S^o$. De maneira similar, defina $\overline{\delta}_k^u$.

Seguindo a notação introduzida anteriormente, considere os conjuntos de índices J^o, I^o, J_i^o, $i \in I^o$, I_j^o, $j \in J^o$ e J_o^k, $k \in K$ (em analogia, respectivamente, a J, I, J_i, $i \in I$, I_j, $j \in J$ e J^k, $k \in K$.

De mesma forma, considere também os conjuntos F_j^o, $j \in J^o$, de vôos e P_i^o, $i \in I^o$ de rotações.

Introduzindo-se variáveis binárias 0-1 $\{y_j : j \in J^o\}$, uma formulação para o subproblema de construção de rotações é dado por:

$$\text{Minimizar} \quad z^o = \sum_{j \in J^o} c_j y_j \qquad (9)$$

sujeito a:

$$\sum_{j \in J_i^o} y_j = 1, \qquad i \in I^o \qquad (10)$$

$$\overline{\delta}_k^l \leq \sum_{j \in J_o^k} h_{kj} y_j \leq \overline{\delta}_k^u, \quad k \in K. \qquad (11)$$

$$y_j \in \{0, 1\}, \qquad j \in J^o. \qquad (12)$$

Denote por $\overline{S}^o$ o conjunto de rotações associadas à solução ótima de (9)-(12). Caso o custo total da solução $(S \setminus S^o) \bigcup \overline{S}^o$, que é claramente viável para o PCR, seja inferior ao custo de S, atualize S como $S := (S \setminus S^o) \bigcup \overline{S}^o$.

O algoritmo de Rubin foi utilizado como base para o desenvolvimento do método de solução aqui proposto. Voltaremos a mencioná-lo mais adiante na descrição do algoritmo desenvolvido.

Melhorias do algoritmo de Rubin

O algoritmo de Rubin serviu de base para um dos primeiros sistemas comerciais desenvolvidos para alocação de tripulação em companhias aéreas, o sistema TPACS ("Trip Pairing and Crew Scheduling"), desenvolvido pela IBM no início dos anos 70. Desde então, ele vem sofrendo diversas modificações, sendo utilizado por várias empresas aéreas de expressão no setor, e tornou-se alvo de estudos de relevância dentro da área.

O sistema TRIP, desenvolvido na American Airlines, utilizou o método de Rubin, modificando inicialmente apenas o número de rotações selecionadas para a geração dos subproblemas, cujo limite passou de 3 para 10 rotações [13].

Em [2], Anbil *et al.* apontaram as principais limitações do algoritmo e os avanços obtidos até então no sentido de superá-las, onde destaca-se como ponto de maior impacto a possibilidade de resolver subproblemas de maior porte, como forma de "aproximar" a solução obtida ao ótimo global.

Uma outra sugestão de melhora do Algoritmo de Rubin foi proposta em 1993 por Graves *et al.* [15] em um sistema desenvolvido para a United Airlines. A primeira fase, de geração de rotações, consiste da obtenção de um conjunto disjunto de rotações. Para isso o sistema utiliza algumas heurísticas e considera, se necessário, vôos dummy. O gerador utiliza uma busca em um grafo onde são representadas

todas as possíveis conexões entre vôos. As rotações são obtidas basicamente de duas formas: enumeração crescente, onde as rotações são formadas partindo-se do primeiro vôo em sentido crescente de tempo, e enumeração interior, onde parte-se de um trecho de vôo "semente" e a rotação é formada a partir deste vôo, buscando-se as conexões necessárias para completar a rotação em ambos os sentidos.

Após obtido o particionamento inicial, é aplicado o método de minimização local de Rubin (2-OPT ou 3-OPT, onde "2" e "3" representam aqui o número de rotações utilizadas para gerar subproblemas), com a diferença de que as colunas anteriores não são retiradas do problema, não sendo feita assim uma substituição e sim uma adição de colunas. Em seguida é feita mais uma busca local, desta vez em subproblemas de maior porte (n-OPT, $n > 3$), tal como em [13], onde o número de subproblemas tal como o valor de n é definido pelo usuário, nesta fase, no entanto, não são geradas todas as possíveis rotações. Para a solução dos subproblemas foi adotado um modelo de particionamento que permite a relaxação das restrições de cobrimento dos vôos, bem como as de quantidade de tripulantes nas bases, com uma determinada penalidade. O método de solução adotado inclui adição de cortes e ordenamento escalonado das colunas ("block echelon ordering").

Heurísticas para o processo de seleção de subproblemas

Housos e Elmroth [19] desenvolveram uma abordagem heurística para a seleção dos subproblemas, incluindo um preprocessamento que busca reduzir o tamanho do problema através do agrupamento de trechos de vôos e ainda, se possível, definir eventuais pontos de troca de aeronaves.

Inicialmente, para cada dia do período a ser planejado, são examinadas todas as possíveis combinações de atividades durante a jornada, bem como todas as possíveis conexões com os dias anterior e posterior (do que difere da maioria das outras abordagens onde são enumeradas apenas as "jornadas de serviço"). Para cada dia são obtidas, através da solução de um problema de recobrimento, as cadeias de atividades mais produtivas que cobrem todos os trechos de vôos daquele dia. As conexões entre os dias vão sendo fixadas à medida em que é solucionado cada subproblema.

Ao final desta fase tem-se uma solução inicial, considerando-se, no entanto, uma relaxação que admite a formação de rotações incompletas. A partir desta solução

inicial, um processo iterativo semelhante é realizado, só que desta vez considerando-se as restrições de viabilidade para as rotações. São fixados todos os dias da semana com os resultados obtidos da solução inicial, com a exceção de apenas um, para o qual é resolvido novamente o subproblema de recobrimento. O processo é repetido para todos os dias da semana até a convergência de uma solução para o problema semanal. A solução para um horizonte maior de planejamento é obtida, de forma análoga, a partir da solução semanal. O método utiliza o algoritmo desenvolvido por Wedelin [36] para a solução dos problemas de recobrimento.

3.0.3 Algoritmos de otimização por subproblemas

Anbil, Tanga e Johnson [3] apresentaram uma modificação do sistema TRIP resultado de um trabalho em conjunto entre a American Airlines e a IBM, com o objetivo de dar um caráter mais global ao método de solução até então utilizado. O algoritmo desenvolvido é bastante referenciado em trabalhos posteriores e foi denotado por abordagem SPRINT. Esta abordagem também serviu como base para o método de solução proposto neste trabalho e será descrita mais formalmente na apresentação do algoritmo desenvolvido.

Na abordagem SPRINT, é gerado inicialmente um grande número de rotações, no caso da American Airlines, para um problema contendo 837 trechos de vôos, foram gerados 12 milhões de rotações, que após a verificação de duplicidade foram reduzidos à 5.5 milhões. Deste conjunto é selecionado aleatoriamente um subconjunto de 5000 colunas. Para este subconjunto é obtida uma solução ótima para a relaxação linear do problema de particionamento correspondente e aplicado o método primal simplex. As variáveis duais obtidas são, então, utilizadas para selecionar, entre o total de 5.5 milhões de colunas, as 5.000 de menor custo reduzido, que constituirão o novo subproblema a ser resolvido, utilizando como base inicial a base ótima do problema anterior.

O processo é repetido até que não reste nenhuma coluna com custo reduzido negativo, o que resulta em um limite inferior para o problema descrito pelas 5,5 milhões de rotações.

Podemos imaginar que, se for atingido um ponto degenerado, dois subproblemas consecutivos poderiam apresentar soluções ótimas de mesmo valor para diferentes

bases. Nesse ponto o algoritmo poderia ciclar. Supomos, portanto, que a demonstração de convergência deste algoritmo utiliza a hipótese de não degenerescência. No entanto, conforme observado em [3], o problema de construção de rotações pode ser altamente degenerado. Por este motivo, no método desenvolvido neste trabalho, foi incluída uma perturbação no "rhs" das restrições como forma de tratar a degenerescência.

Em [8], é apresentado um algoritmo que utiliza o mesmo princípio da abordagem SPRINT, no entanto com outro critério de parada: quando dois subproblemas consecutivos apresentarem soluções ótimas com valores aproximadamente iguais. Este critério evita a ciclagem que ocorre em pontos degenerados, no entanto, se ainda existirem colunas com custos reduzidos negativos, não se pode provar a otimalidade da solução obtida.

Utilizamos no algoritmo aqui proposto, como veremos mais adiante, o princípio da abordagem SPRINT, porém sem tratar uma possível degenerescência do problema. Evitamos a ciclagem com o mesmo critério de parada utilizado em [8], mesmo que ainda persistam colunas com custos reduzidos negativos. No entanto, como inserimos esta abordagem em um algoritmo Primal-Dual, temos um gap de dualidade associado às soluções obtidas.

A figura 3.1 ilustra a composição dos subproblemas na abordagem SPRINT.

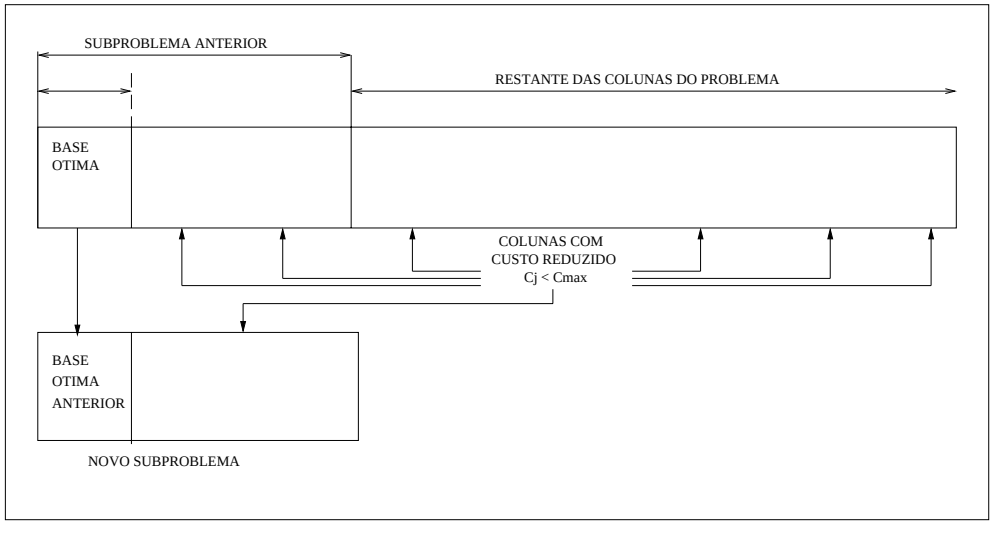


Figura 3.1: Composição dos subproblemas na abordagem SPRINT

Após finalizada a etapa de obtenção do limite inferior, foi realizada a busca de uma solução inteira.

Neste ponto, o método não demonstrou fazer bom uso de todo o esforço realizado na etapa anterior. A relaxação linear foi utilizada apenas como base para uma heurística que, de acordo com os valores obtidos para as variáveis, fixa sucessores aos trechos de vôos até que todos os vôos apresentem um único sucessor [32].

Dando continuidade a esta abordagem, e ainda utilizando heurísticas para a obtenção de uma solução inteira como etapa final do algoritmo, Chu *et al.* [8] apresentaram um algoritmo heurístico onde o problema de particionamento é modelado em um grafo e cada rotação é definido por um circuito (onde um arco fictício liga o último ao primeiro trecho de vôo). A estratégia de branching utilizada segue o mesmo princípio da heurística desenvolvida por Ryan e Falkner em [32].

Ainda utilizando a abordagem de otimização por subproblemas, Klabjan, Johnson e Nemhauser [22] apresentaram um novo algoritmo de solução para o problema, também a partir do modelo de particionamento. Uma vez que o algoritmo apresentado serviu de base para a concepção do algoritmo desenvolvido neste trabalho, apresentaremos o mesmo com mais detalhes.

A metodologia proposta envolve um algoritmo gerador de rotações baseado na meta-heurística GRASP [30], em processamento paralelo, combinando aleatoriedade a um critério guloso referente aos tempos de conexão. Após a geração das rotações, a relaxação linear do problema de particionamento resultante é resolvida através do algoritmo primal-dual por subproblemas paralelo [24], até a sua "quase" otimalidade, e por fim é utilizada uma heurística de redução de colunas e o problema final é submetido a um software comercial para a obtenção de soluções inteiras.

O algoritmo realiza várias iterações compreendendo geração de colunas e obtenção da relaxação linear do problema resultante, mantendo-se sempre as colunas associadas às bases ótimas obtidas (o que garante que o valor da função objetivo não cresça em nenhuma iteração). O procedimento é interrompido quando o decréscimo de valor da função objetivo cai abaixo de um determinado limite.

Após esta fase é iniciado o procedimento de seleção das colunas para a obtenção de soluções inteiras. Se tiverem sido realizados k iterações na fase anterior, são selecionadas 3k milhões de colunas com os menores valores de custos reduzidos e uma nova etapa de relaxação linear é executada para o problema resultante. A etapa seguinte consiste na remoção de colunas duplicadas, através de um algoritmo paralelo.

Em seguida uma nova etapa de seleção de colunas é realizada, onde são selecionadas todas as que apresentarem custo reduzido inferior a um determinado valor. Após esta seleção, uma etapa de redução de colunas é realizada, utilizando uma heurística com base nos valores obtidos para as variáveis na solução da relaxação linear. Esta heurística fixa pares de vôos consecutivos presentes nas rotações que tiveram valores iguais a 1 e também na de valor mais próximo a 1. Fixar pares de vôos significa eliminar do problema todas as rotações que apresentarem um dos vôos isoladamente. Mais uma vez a relaxação linear do problema resultante é resolvida e novamente é aplicada a heurística de redução de colunas. O procedimento é repetido até que o problema atinja o tamanho desejado, no caso apresentado esse tamanho foi fixado em 100 mil colunas. A heurística reduz também o número de linhas do problema à medida que fixa os pares de vôos. Por fim o problema reduzido é submetido a um software comercial para a obtenção de soluções inteiras, adicionado de regras de ramificação específicas para o problema, que ramifica de acordo com a sucessão ou não de pares de vôos, denotado na literatura de "follow-on branching". Cabe destacar que esta regra é valida, no entanto, apenas para o modelo de particionamento e não poderia ser aplicada para o modelo que adotamos aqui.

O algoritmo desenvolvido foi testado em instâncias com 342, 449 e 654 trechos de vôos para os quais os resultados foram avaliados em termos de qualidade das soluções obtidas.

Detalharemos um pouco mais a heurística de geração de colunas utilizada já que teve como base a meta-heurística GRASP, também utilizada no gerador desenvolvido neste trabalho.

No caso tratado neste trabalho, foram geradas inicialmente entre 50 a 75 milhões de rotações. Para isto foram utilizados dois "clusters" de máquinas, com um total de 160 processadores, em 16 máquinas Quad Pentium com 256Mb de memória RAM cada, e 48 máquinas Dual Pentium com 512Mb de memória cada. Cabe destacar que não foi realizada, no entanto, nenhuma verificação para eliminação de rotações duplicadas. Veremos mais adiante o método desenvolvido para realizar esta verificação já nesta fase inicial de geração, o que poupa bastante memória de armazenamento e evita que rotações duplicadas sejam incluídas no problemas a serem resolvidos.

A geração das rotações foi realizada utilizando-se um método que adota como

parâmetro para avaliação da qualidade de uma rotação, o tempo de conexão entre os vôos. A geração foi realizada em duas etapas: primeiramente foram construídas jornadas de serviço diárias a partir dos trechos de vôo, e em seguida foram formadas as rotações a partir das jornadas de serviço construídas.

Para a geração das jornadas de serviço, foi construído um grafo orientado no tempo, com dois nós e um arco para cada segmento de vôo, os nós representando as partidas e as chegadas dos vôos, e um arco interligando estes nós, representando o vôo. Entre o nó de chegada de um vôo i e o nó de partida de um vôo j, existe um arco ij se o aeroporto de chegada do vôo i for o mesmo que o de partida do vôo j e se o tempo de conexão t_{ij} entre os vôos for menor que o máximo permitido pela regulamentação, denominado $maxSit$.

Para todos os arcos de conexão entre vôos, foram estabelecidas as probabilidades de escolha dos mesmos em função dos tempos de conexão $p_{ij} = f(t_{ij})$. As jornadas de serviço foram formadas a partir de uma busca em profundidade no grafo, conectando-se os vôos segundo os valores de p_{ij}, e verificando-se a viabilidade da jornada após cada conexão estabelecida.

Para algumas conexões foram fixadas probabilidades iguais a um, ou seja, os vôos são pré-conectados devido às características da malha. No caso tratado foram pré-conectados os vôos operados pela mesma aeronave e para os quais o tempo de solo entre os mesmos não é suficiente para que se efetue uma troca de tripulação, e também aqueles que se conectam em aeroportos do tipo "spoke", onde as conexões são suficientemente esparsas a ponto de se poder determinar de antemão quais vôos seguirão com a mesma tripulação. Para o restante dos casos, foi definida uma função de probabilidade para a qual conexões mais longas têm menor probabilidade de serem selecionadas. A função de probabilidade adotada foi:

$p_{ij} = exp(-\zeta(t_{ij} - minSit)^2))$

onde ζ é um parâmetro através do qual é atribuído um peso ao tempo de conexão e $minSit$ é o tempo mínimo permitido. O valor de ζ também deve satifazer um número esperado E de conexões a serem estabelecidas em cada nó do grafo, ou seja:

$g(\zeta) = \sum_{j}^{X} exp(-\zeta(t_{ij} - minSit)^2) = E$

Introduzindo-se uma variável $\xi = exp(-\zeta)$, a solução da equação acima é a raiz de um polinômio e pode ser calculada através do método de Newton, o que foi feito

para cada nó do grafo, uma vez que o número de conexões não é demasiado grande.

Após calculados os valores de p_{ij}, foi realizada a busca no grafo e enumeradas as jornadas de serviço a partir das quais serão geradas as rotações.

O processo de geração das rotações seguiu o mesmo princípio da geração das jornadas de serviço, estabelecendo-se conexões a partir de uma função de probabilidade. No entanto, devido à diferença de magnitude do número de possíveis conexões entre um grafo de jornadas de serviço e o de trechos de vôos, não foi possível proceder da mesma forma que na etapa de enumeração das jornadas.

Seja S_j o conjunto de todas as jornadas de serviço que iniciam com o trecho de vôo j e denote por a_j o tempo de início destas jornadas, se queremos expandir uma rotação que termina em uma jornada d, procuraremos entre todas as jornadas com a_j que satisfaçam os tempos mínimo e máximo permitidos a partir do horario de chegada do último vôo em d e que partam do seu aeroporto de destino.

Para contornar a dificuldade de se calcular os valores de ζ para todas as possíveis conexões entre jornadas, é estabelecido, aleatoriamente, um intervalo de tempo de n minutos, denominado de tamanho do passo, de tal forma que, para expandir uma rotação cuja última jornada foi selecionada de um conjunto S_j, serão consideradas as jornadas pertencentes ao primeiro conjunto S_k, com tempo de início $a_k > a_j + n$.

Para a determinação do tamanho do passo foi utilizada a distribuição normal $n \sim N(\mu, \sigma^2)$, onde a média μ é uma função do tempo de conexão entre um conjunto S_j e a última jornada selecionada para a rotação, e a variância σ é determinada durante o processo.

Uma vez fixado o conjunto S_k, parte-se para a seleção da jornada dentro deste conjunto. O método de escolha da jornada depende de uma função de probabilidade definida como:

$p = exp(-\tau(t_k - minRest)^2)$

Note que a função de probabilidade fornece o mesmo valor para qualquer jornada pertencente ao conjunto S_k. Utilizando-se um parâmetro $pMethod$, foi estabelecido que, se $p > pMethod$, seleciona-se aleatoriamente uma jornada em S_k, e se $p \leq pMethod$, utiliza-se uma distribuição geométrica procedendo-se da seguinte maneira: seleciona-se um número u a partir de uma distribuição uniforme [0,1] e calcula-se $\tilde{u} = \lfloor lnu/ln(1-p) \rfloor$, que é geometricamente distribuído com parâmetro p e indica

a posição da jornada a ser escolhida. No que se refere ao tratamento dos custos das rotações geradas, foram adotados os seguintes parâmetros:

- o custo das jornadas de serviço é o máximo entre: o tempo de vôo, uma fração do tempo total decorrido e um mínimo garantido de pagamento;
- o custo de uma rotação é o máximo entre: a soma dos custos das jornadas de serviço, uma fração do tempo decorrido fora da base e o pagamento mínimo garantido vezes o número de dias contidos na rotação;
- o custo excedente ou "pay-and-credit" de uma rotação é o custo da rotação menos o tempo de vôo;
- o "flight time credit" (FTC) de uma rotação é o custo excedente vezes 100 dividido pelo tempo de vôo.

Durante o processo de formação das rotações são podadas as ramificações que conduziriam a uma rotação com FTC superior a um dado valor K, o que é feito calculando-se um limite inferior para o FTC de uma rotação parcial, dada pela expressão:

$$\frac{\sum_{i=1} kdc_{d_i} + (maxDuties - k).maxFly}{\sum_{i=1} kfl_{d_i} + (maxDuties - k).maxFly} \geq K + 1$$

onde $0 < K < 1$.

Cabe destacar no entanto que não se pode saber "a priori" se rotações de alto custo poderiam fazer parte de boas soluções. Estas podas podem, portanto, eliminar rotações que estariam contidas em soluções de custo total inferior às que seriam obtidas com este procedimento. Por este motivo, no gerador aqui desenvolvido não fizemos uso de procedimentos deste tipo.

3.0.4 Algoritmos do tipo "Branch-and-Cut"

Hoffman e Padberg [17] apresentaram uma abordagem de solução utilizando o método "branch-and-cut" [29] para a resolução exata de problemas de particionamento de grande porte.

Inicialmente, os autores avaliaram, para instâncias de pequeno porte, a qualidade da solução obtida através do Algoritmo de Rubin. Em particular, em um dos testes

envolvendo 145 trechos de vôos, foram geradas todas as rotações viáveis (cerca de 1 milhão), e a solução ótima obtida proporcionou uma melhoria de 0.5% no custo total das rotações, em relação à abordagem baseada no método de Rubin. Os autores concluiram que, diante daquela diferença para uma instância de pequeno porte, ganhos maiores deveriam ser possíveis para instâncias de maior porte. A conclusão parece obviamente correta. No entanto, mesmo um sofisticado algoritmo de solução exata, como aquele proposto em [17], seria incapaz de resolver instâncias de grande porte. O potencial de ganhos seria assim reduzido pela necessidade prática de se restringir o conjunto de rotações a trabalhar.

O método desenvolvido por Hoffman e Padberg utiliza uma heurística para a obtenção de soluções primais (i.e. soluções inteiras viáveis) e adiciona cortes poliedrais para reforçar os limites duais obtidos por relaxação linear.

3.0.5 Algoritmos de geração de colunas

Minoux *et al.* [25] apresentaram um novo método de resolução para o problema contínuo de construção de rotações considerando implicitamente todas as rotações viáveis. Trata-se do método de geração de colunas aplicado à relaxação linear do problema de recobrimento.

Considere que em uma etapa t do algoritmo, um subconjunto $J_t \subset J = \{1, 2, ..., K\}$ do conjunto de todas as rotações viáveis do problema tenha sido gerado. A cada rotação é associado um custo medido em unidade de tempo, composto de duas parcelas: TAT ("Total Absence Time"), tempo decorrido entre a partida do primeiro trecho de vôo da rotação e a chegada do último; e RT ("Rest Time"), que é representado por uma função não negativa de TAT.

É resolvido o problema restrito às colunas em J_t:

$$
\begin{aligned}
&\text{Minimizar} && \sum_{k \in J_t} c_k x_k \\
&\text{sujeito a:} && \\
& && \sum_{k \in J_t} R(k) x_k \geq 1, \\
& && x_k \geq 0 \qquad\qquad k \in J_t
\end{aligned}
$$

Onde c_k é o custo associado à rotação k; x_k é uma variável binária que assume valor igual a 1 se a rotação k estiver presente na solução, ou igual a zero caso contrário; R(k) é um vetor 0-1 associado à rotação k, com componentes $R_i(k)$, $i = \{1, ..., NumVoos\}$, definido da seguinte forma: $R_i(k) = 1$, se o vôo i está contido na rotação k; e $R_i(k) = 0$, caso contrário.

As variáveis duais $\pi_i, i = \{1, ..., NumVoos\}$ associadas à solução deste problema são então utilizadas em um problema "gerador de colunas" para a geração da rotação $k_o \in J$ de menor custo reduzido:

$$\begin{aligned} \bar{c}_{k_o} &= c_{k_o} - \sum_{i=1}^{NumVoos} \pi_i R_i(k_o) \\ &= \min_{k=1,...,K} \left\{ c_k - \sum_{i=1}^{NumVoos} \pi_i R_i(k) \right\} \end{aligned}$$

Inicialmente são enumeradas todas as jornadas de serviço viáveis. Associando-se a cada jornada gerada um nó, é contruído um grafo auxiliar simplificado G, contendo todas as rotações viáveis possíveis a partir destas jornadas, no entanto reduzindo as restrições de viabilidade das rotações a apenas uma condição: o intervalo entre duas jornadas de serviço é de no mínimo 12 horas. Em seguida, a partir de G, é construído um grafo G', considerando desta vez todas as restrições de viabilidade existentes.

A cada nó j em G', que representa uma jornada de serviço f_j, é associado um comprimento $l_j = -\sum_{i \in f_j} \pi_i$, correspondente à soma de todos os valores duais associados aos vôos contidos na jornada f_j. Uma vez que o grafo G' é acíclico, não existe problema em se considerar comprimentos negativos.

Considerando que o custo de cada rotação depende apenas do primeiro e do último trecho de vôo, os custos reduzidos são computados da seguinte maneira: a cada jornada f_i tomada como origem para a rotação, adiciona-se, ao comprimento de cada uma das demais jornadas f_j que poderiam constituir uma jornada de término, o custo da rotação que se inicia em f_i e termina em f_j. A coluna k_o é obtida então através da resolução do algoritmo de Dijkstra para obtenção do caminho mínimo no grafo G'. Em seguida, se a coluna k_o obtida apresentar um valor não negativo para o custo reduzido, pode-se afirmar que a solução obtida para o problema 3.0.5,

é também solução do problema envolvendo todas as rotações em J. Se, no entanto, o custo reduzido da coluna k_o for negativo esta é incorporada ao problema 3.0.5 e uma nova solução é encontrada produzindo um novo vetor π de variáveis duais. Este procedimento é repetido até que inexistam rotações viáveis em G' com custo reduzido negativo

Os primeiros experimentos utilizando essa abordagem em [25] apontaram que a maior parte do tempo computacional para execução do algoritmo era gasto na obtenção das colunas de mínimo custo reduzido. Em vista disso, procurou-se minimizar o número total de chamadas ao problema gerador através de uma abordagem denotada "Multiple Pricing", que permite que várias colunas com custos reduzidos negativos sejam incorporadas de uma só vez ao problema reduzido. Uma vez que para obter a coluna de menor custo reduzido é necessário obter, para cada par (f_i, f_j), a rotação de menor custo reduzido que se inicia em f_i e termina em f_j, várias colunas com custo reduzido negativo são encontradas a cada chamada do gerador sem nenhum custo computacional adicional.

Diferentes abordagens de 'Multiple Pricing" podem ser adotadas incorporando: todas as colunas com custos reduzidos negativos encontradas; ou apenas as que apresentam custos reduzidos abaixo de um determinado valor θ; ou ainda as p colunas de menores valores de custos reduzidos. Em [25], a abordagem que apresentou melhores resultados em redução de tempo do algoritmo foi a primeira, apontando que o problema gerador é realmente a fase mais crítica em termos de custo computacional.

É importante ressaltar que as instâncias utilizadas para teste em [25] variaram entre 20 a 329 trechos de vôos, envolvendo entre 38 e 1113 possíveis jornadas de serviço viáveis, respectivamente.

Gerar colunas através de algoritmos de caminhos mínimos para instâncias de grande porte, com dezenas de milhares de vôos, envolveria grafos excessivamente grandes, como veremos mais adiante, que praticamente inviabilizariam a aplicação deste tipo de abordagem.

Também é importante destacar que, somente é possível garantir a otimalidade do método proposto, quando (como assumido em [25]) o custo de uma rotação depende apenas dos trechos de vôo que a mesma cobre.

No caso tratado no presente trabalho são consideradas outras parcelas de custo,

além do tempo decorrido fora da base dos tripulantes, que não poderiam ser computados apenas a partir dos trechos iniciais e finais das rotações. O custo de pernoites em hotéis, por exemplo, varia bastante com a cidade. Não poderíamos saber *a priori*, apenas a partir dos trechos de vôos extremos, em quais cidades a tripulação iria pernoitar durante a rotação. Em alguns casos poderíamos incluir este custo no nó correspondente à jornada que contém o pernoite, mas existem casos em que os pernoites podem acontecer entre dois dias inativos, ou seja, não estariam associados a um nó específico. Não poderíamos também, utilizando esta abordagem, aplicar penalidades a dias inativos. Dependendo da estrutura de custos da empresa uma modelagem que aproxima o custo da rotação de forma a possibilitar a sua obtenção apenas a partir dos trechos de vôos extremos, pode ser razoável ou não.

Concluímos que, para a realidade brasileira e mais especificamente para o caso tratado neste trabalho, a abordagem de geração de colunas não é a mais adequada.

Vários outros trabalhos deram continuidade à abordagem de geração de colunas para construção de rotações. Desaulniers *et al.* [10] apresentaram uma formulação para o gerador de colunas como um problema de multifluxo em uma rede com recursos restritos, com variáveis inteiras, onde cada fluxo corresponde a uma tripulação e os recursos representam alguns parâmetros que restringem a viabilidade das rotações.

Para cada um dos tripulantes considerados no problema, foi atribuído um grafo onde os nós representam localidades no tempo, e os arcos representam possíveis atividades, tais como um vôo, uma troca de avião, um período de descanso, um vôo extra, etc. Cada grafo apresenta um nó origem e um nó sumidouro, que representam, respectivamente, o início e o fim da rotação a ser atribuído àquela tripulação. Em cada grafo deve ser estabelecido um fluxo unitário que corresponde à rotação a ser realizado pela mesma.

As restrições do problema foram consideradas em dois níveis: global, que se referem à viabilidade do conjunto de rotações; e internas, que devem ser obedecidas em cada uma das rotações. Sendo estas últimas representadas na forma de recursos restritos. O problema foi decomposto seguindo o princípio de decomposição de Dantzig-Wolfe, onde os subproblemas foram definidos como problemas de caminhos mínimos com recursos restritos para cada uma das tripulações, cada caminho cor-

respondendo a uma rotação. No problema mestre foram consideradas as restrições de viabilidade da solução a nível global e as referentes à cobertura dos vôos, e ainda a garantia de ser atribuído apenas uma rotação a cada tripulação.

A relaxação linear do problema mestre é resolvida, inicialmente para um número limitado de variáveis de caminho (rotações). Após obtido o ótimo linear, os duais correspondentes são utilizados nos subproblemas gerados, resultando em novas colunas que serão adicionadas ao problema mestre.

O processo se repete até que não seja mais encontrada nenhuma nova coluna com custo reduzido negativo. É obtido, portanto, o ótimo linear do problema, considerando-se implicitamente todas as variáveis de caminho. Em seguida é iniciado o procedimento de Branch-and-Bound. A ramificação é feita considerando, para uma determinada rotação que apresentou valor fracionário na solução anterior, o fluxo igual a 1 em um dos novos ramos, e nulo no outro. Para cada nó do Branch-and-Bound novas colunas são geradas até a obtenção do ótimo linear, e o processo continua até ser encontrada a solução inteira.

Para três das onze instâncias resolvidas fornecidas pela Air France, foram obtidas soluções inteiras já no primeiro nó, antes de ser iniciado o processo de ramificação. Nos problemas restantes o número máximo de ramificações realizadas foi de 26 (para um problema com 566 trechos de vôo e 2 bases). O método proporcionou melhorias significativas, quantificadas em termos de tempo fora da base e tempo de repouso requerido subsequente às rotações, em relação à solução adotada pela empresa.

Este método serviu como base para o sistema ALTITUDE, desenvolvido pela Ad Opt Technologies em cooperação com o GERAD, um centro de pesquisas no Canadá. O sistema foi implementado em uma empresa de vôos "charter" e chegou a proporcionar reduções de custo de entre 8 a 12% já no primeiro ano de operação [11].

Vance *et al.* [34] apresentaram uma nova formulação para o problema utilizando também como abordagem de solução o método de geração de colunas. O problema foi decomposto em duas etapas. Na primeira etapa é obtido um conjunto de jornadas de serviço que cobrem todos os trechos de vôos, e na segunda são construídas as rotações a partir deste conjunto. Cada vôo deve ser coberto exatamente por apenas uma jornada de serviço e cada jornada de serviço por uma rotação.

A decomposição foi feita seguindo o princípio de Dantzig-Wolfe, onde cada subproblema foi definido como um problema de particionamento, e cada coluna gerada representa todo um conjunto de jornadas de serviço que particionam os vôos. O problema mestre deve selecionar apenas um destes conjuntos, e a partir deste determinar os rotações componentes da solução. Após a tentativa desta primeira abordagem, que se apresentou muito lenta, algumas possíveis alterações dos conjuntos foram permitidas para dar mais flexibilidade ao modelo.

Os limites inferiores obtidos da relaxação linear deste modelo foram melhores se comparados ao modelo tradicional de particionamento, no entanto, a formulação tornou o problema mais difícil de ser resolvido.

Gustafsson apresenta em [16] uma abordagem heurística de geração de colunas para o problema de construção de rotações utilizada pela *Carmem Systems*, empresa que atualmente desenvolve softwares na área de scheduling para diversas companhias aéreas na Europa.

O método utiliza, para a solução do problema mestre, uma heurística com base na relaxação lagrangeana, que encontra soluções inteiras para o problema de recobrimento e fornece valores para as variáveis duais utilizados nos subproblemas geradores de rotações. O tempo de solução para esta abordagem é, no entanto, bem maior que o da solução da relaxação linear através do simplex. O critério de parada é bem mais difícil de ser estabelecido e os duais produzidos sofrem uma variação muito grande, resultando em chamadas desnecessárias ao gerador de colunas e consequentemente em rotações desinteressantes sendo adicionados ao problema.

A fim de proporcionar uma melhoria neste sentido, Gustafsson propõe ainda uma abordagem híbrida para a solução do problema através de uma combinação entre a heurística adotada pela empresa e o método do subgradiente, o qual é ativado para a obtenção dos duais quando os fornecidos pela heurística estiverem variando além de uma determinada faixa. A nova abordagem produziu soluções de melhor qualidade e mais rapidamente se comparada ao método tradicional de solução da relaxação linear do problema mestre através do simplex.

Anbil, Forrest e Pulleyblank apresentaram em [1] uma abordagem de geração de colunas utilizando o algoritmo do volume na resolução das relaxações lineares. O algoritmo do volume foi introduzido por Barahona e Anbil em 1997, desenvolvido

partir do método do subgradiente, e obtém soluções primais viáveis para a relaxação linear do problema. Para a solução do problema de programação inteira foi apresentado um algoritmo envolvendo: a abordagem SPRINT [3], incluindo geração de colunas através de um algoritmo de caminhos mínimos em um grafo de jornadas de serviço, como em [25]; e redução do número de linhas através de fixação de pares de vôos com base nas soluções das relaxações lineares obtidas, até que o problema atinja um tamanho que permita a aplicação de métodos exatos para obtenção de uma solução inteira. O algoritmo proposto foi testado em problemas da *US Airways* e *Southwest Airlines* com malhas de 2504, 2991 e 4810 vôos, no entanto não foram publicados, no artigo referenciado, os resultados obtidos.

3.0.6 Decomposição por Relaxação Lagrangeana

Beasley e Cao [7] desenvolveram um algoritmo do tipo "Branch-and-Bound", onde, para cada iteração do método do subgradiente, rotações são geradas através de programação dinâmica.

Para um dado conjunto de multiplicadores de Lagrange associados, uma vez relaxadas as restrições associadas, o subproblema lagrangeano resultante pode ser formulado como: encontrar um número determinado de caminhos disjuntos em um grafo que passem por todos os nós a um custo mínimo. Os nós representam as tarefas a serem realizadas (no caso do problema de rotas aéreas seriam os vôos), e os arcos representam as transições entre as tarefas (que poderiam corresponder a deadheads, períodos de repouso, etc). O grafo é orientado no tempo e são representados somente os arcos que correspondem a transições viáveis. Não são considerados, no entanto, alguns aspectos de viabilidade e de custos presentes no problema de construção de rotações.

O algoritmo foi testado em problemas fictícios e alcançou a solução ótima em instâncias de tamanho relativamente grande.

3.0.7 Algoritmos heurísticos específicos para o problema

Ball e Roberts [5] desenvolveram uma abordagem de solução baseada num algoritmo para encontrar caminhos disjuntos em um grafo, onde os caminhos correspondem a rotações. Após obtido um conjunto de rotações correspondentes à uma

partição do grafo (solução viável inicial), um algoritmo para melhoria das rotações é aplicado utilizando o método de emparelhamento ("matching") até que não seja possível mais nenhuma melhoria. Para finalizar são incorporados os deadheads necessários.

Wark *et al.* [35] apresentaram um método semelhante baseado na resolução de uma sequência de problemas de emparelhamento ("repeated matching"). Inicialmente é considerada a solução onde cada rotação é composta de um único vôo, e a partir daí é aplicado repetidas vezes um algoritmo de emparelhamento de custo mínimo até que não mais sejam obtidas reduções de custo.

Este último método foi aplicado em problemas reais onde são admitidas rotações com tripulação adicional, ou seja, incluindo um terceiro piloto, e obteve melhores resultados que a abordagem tradicional, onde é utilizado o modelo de particionamento e a alocação do tripulante adicional é realizada separadamente, após resolvido o problema para uma tripulação simples.

Capítulo 4

Justificativas para a abordagem proposta

O enfoque dado à metodologia de solução do PCR desenvolvida neste trabalho foi motivado por alguns aspectos observados tanto na revisão bibliográfica realizada como na experiência prática obtida na Rio Sul linhas aéreas, onde foi feito o estudo de caso.

A primeira questão observada foi relativa ao horizonte de planejamento. A maioria dos trabalhos propostos na literatura nesta área são aplicáveis a horizontes de planejamento de um dia ou uma semana, denotados na literatura, respectivamente, por "daily problem" e "weekly problem" [5], [10], [26] e [33]. No primeiro caso, considera-se que os trechos de vôos se repetem diariamente, e no segundo caso, a cada semana. No entanto, para a configuração da malha de vôos da companhia mencionada, estas aproximações não eram muito boas, devido a um número considerável de vôos irregulares. Surgiu portanto o interesse em desenvolver uma metodologia aplicável a um horizonte de planejamento de aproximadamente um mês, referenciado na literatura por "fully dated problem".

Seria necessário, portanto, desenvolver uma metodologia que pudesse ser aplicável a uma malha consideravelmente maior. Tomamos como base, para desenvolvimento do método, um caso com uma malha composta por 11210 trechos de vôos, referentes a um período de 34 dias, e envolvendo 48 aeroportos, sendo que 6 destes são bases domiciliares de tripulantes. O problema resultante teria, portanto, considerando o modelo apresentado anteriormente, 11210 linhas relativas ao cobrimento dos vôos e mais 204 (6 bases x 34 dias) restrições diárias de disponibilidade de tripulação nas bases, totalizando 11414 linhas.

A maioria dos trabalhos na literatura trabalham com malhas inferiores a mil trechos de vôos: 34 a 210 vôos [9]; 329 vôos [25]; 837 vôos [3]; 825 vôos [17]; até 500 vôos [28] [6]; até 654 vôos [22]; e até 959 vôos [23].

Em [5] são apresentados resultados para uma heurística aplicada a problemas de até 1058 vôos. Em [33] são apresentados resultados para problemas com 2000 trechos de vôos.

Observou-se também, durante o estudo de caso, que os trechos de vôos eram alocados às bases antes de iniciar a solução do PCR, como forma de simplificar o processo de geração das rotações. É evidente que não se pode saber "a priori" qual a divisão dos trechos entre as bases que possibilitaria a construção das rotações de mínimo custo. Seria bem mais inteligente deixar que o processo de otimização escolhesse esta divisão. No entanto, para isso, seria necessário trabalhar com a malha inteira para cada uma das bases, o que muliplica pelo número de bases a quantidade de jornadas de serviço geradas, como veremos mais adiante na descrição do algoritmo de geração das rotações.

Tornou-se, portanto, ainda maior, a necessidade de uma metodologia capaz de trabalhar com um grafo de grande porte, o que excluiu a opção pelos algoritmos de geração de colunas, como também veremos adiante, com mais detalhes.

Capítulo 5

Geração das rotações

O algoritmo de geração das rotações foi desenvolvido com dois intuitos principais: construir rotações que levem a soluções de "boa qualidade" para o PCR, e gerar um número grande de rotações distintas.

No que se refere à qualidade da solução, foram considerados os seguintes aspectos:

Como a estratégia de solução para o PCR adotada aqui foi a de encontrar uma solução de boa qualidade a partir de um subconjunto de todas as possíveis rotações viáveis, é fundamental que este subconjunto seja "bem" escolhido. Não seria coerente dispender o grande esforço requerido na solução de um PR de grande porte se este não incluisse rotações que possibilitassem soluções de boa qualidade.

Durante o processo de desenvolvimento do algoritmo, observou-se que soluções de baixo custo apresentavam algumas rotações de alto custo, necessárias para cobrir alguns vôos que não se conectavam bem com o restante da malha. Muito pouco provavelmente estas rotações seriam geradas em um algoritmo que analisasse apenas a qualidade de cada rotação, ou mesmo que incluisse outros critérios mas referentes apenas à rotação, sem "enxergar" a malha, e consequentemente esta solução não seria alcançada mesmo se fosse obtido o ótimo exato do PR resultante.

A estratégia adotada para gerar rotações que possam compor soluções de boa qualidade foi utilizar a meta-heurística GRASP para construir várias soluções diferentes a cada iteração buscando atender a critérios de qualidade estabelecidos. Dessa forma, as rotações construídas são as que possibilitam diversas combinações de soluções e o processo de construção das mesmas visa diminuir o custo total da solução.

Quanto à geração de uma grande quantidade de rotações distintas, os principais

problemas encontrados durante a implementação foram: elevado tempo de processamento e necessidade de grande quantidade de memória de armazenamento.

A estratégia adotada para tentar minimizar estes problemas foi paralelizar o processamento. A tarefa de construção das rotações propriamente dita foi atribuída a diversos escravos já que se tratam de procedimentos independentes; e o armazenamento e eliminação das rotações duplicadas foram atribuídos ao processo mestre.

Dessa forma, os escravos não necessitam de grande quantidade de memória para armazenamento das rotações, aumentando sensivelmente a velocidade de execução do algoritmo; e ao mestre cabe apenas um algoritmo de armazenamento estruturado das rotações, o que diminuiu bastante a necessidade de memória, uma vez que não são incluídas as rotações repetidas.

Antes de detalhar o algoritmo, faremos uma breve descrição da meta-heurística GRASP, e apresentaremos algumas considerações no que se refere aos critérios de viabilidade e de avaliação da qualidade de uma rotação.

5.1 A meta-heurística GRASP e sua aplicação em problemas de otimização combinatória

A meta-heurística GRASP ("Greedy Randomized Adaptive Search Procedures") é uma técnica iterativa, onde em cada iteração é construída uma solução viável para o problema. As iterações consistem basicamente de duas fases: construção de uma solução e uma busca local onde tenta-se a melhoria da solução obtida. Ao final de um determinado número de iterações, ou após ser atingido algum critério de qualidade definido, toma-se a melhor solução construída como solução final do problema.

O processo de construção de cada solução é realizado adicionando-se, passo a passo, os elementos componentes da solução, segundo uma função de avaliação adaptativa, de forma gulosa e aleatória.

Primeiramente é definida uma função de avaliação para os elementos a partir dos quais será construída a solução. A função deve representar o benefício (míope) de adição do referido elemento na solução. A função é adaptativa na medida em que os benefícios associados aos elementos sofrem atualizações a cada iteração, refletindo os impactos causados pela escolha do elemento anterior.

Para cada etapa do processo construtivo é feita uma lista de elementos candida-

tos (CL- "Candidate List"), ordenados segundo a função de avaliação definida. No processo de escolha do elemento a ser inserido na solução entra o caráter aleatório do GRASP, não é escolhido necessariamente o elemento no topo da lista, mas dentro de uma determinada faixa de maior benefício (RCL - "Restricted Candidate List").

A RCL é determinada fixando-se um coeficiente correspondente ao percentual de elementos de maior benefício da lista de candidatos que se queira considerar para a escolha. Quanto mais este coeficiente se aproxima de 0, mais gulosa é a heurística, quanto mais próximo de 1, mais aleatória. Esta aleatoriedade permite que sejam construídas diversas soluções diferentes, sem, no entanto, comprometer a atuação inteligente da função de avaliação.

Após a seleção do novo elemento segue-se a atualização da função de avaliação e o processo é repetido até que seja construída uma solução viável para o problema. Ao final de cada iteração recomenda-se que se faça uma busca local com o intuito de identificar alguma solução de melhor qualidade na vizinhança da solução obtida.

Encontra-se na literatura implementações da meta-heurística GRASP cobrindo uma gama variada de aplicações em problemas industriais, como scheduling e localização, além de outros envolvendo lógica ou que possam ser modelados através de grafos ou problemas de recobrimento de conjuntos. Em [12] encontra-se a aplicação do GRASP em problemas de transporte aéreo representados pelo modelo de recobrimento.

A meta-heurística GRASP tem-se mostrado bastante eficiente em problemas práticos de otimização combinatória, e pode ser facilmente implementada em processadores paralelos [30].

5.2 Condições de viabilidade adotadas para as rotações

O conceito de viabilidade de uma rotação adotado neste trabalho refere-se às restrições impostas pela regulamentação do aeronauta ou determinações dos sindicatos no domínio da companhia onde foi feito o estudo de caso, bem como regras internas ou critérios de qualidade adotados pela empresa. No que se refere à jornada de trabalho, as restrições consideradas foram:

- O limite máximo de 11 horas diárias de trabalho, contadas desde o momento de apresentação do tripulante no aeroporto para início da jornada até o momento de sua liberação, admitindo uma folga de 1 hora em relação à jornada máxima definida pela regulamentação;
- O máximo de cinco pousos contidos na jornada;
- O intervalo mínimo referente aos tempos de "briefing" e "debriefing" entre dois vôos consecutivos de 30 minutos para vôos com a mesma aeronave e de 60 minutos nos casos onde há troca de aeronave;

Em relação à rotação, foi adotado:

- O período máximo de 5 dias;
- O tempo mínimo de 12 horas de repouso entre duas jornadas de trabalho, acrescentado de 30 minutos para desligamento dos motores e 30 minutos para a apresentação do tripulante antes do início da jornada;
- Os trechos inicial e final da rotação partindo e retornando respectivamente da mesma localidade, que constitui a base domiciliar da tripulação que o realizará;
- O máximo de um dia de inatividade na rotação;

Adotou-se um horizonte de planejamento de aproximadamente um mês, por ser este o mais adequado ao caso estudado.

Conforme já mencionado a rotação deve partir e retornar à base domiciliar da tripulação a ela designada. Tal restrição limita o número de rotações a ser atribuído a cada base em função da tripulação disponível nas mesmas. Uma das formas de tratar esta restrição seria fazer o planejamento separadamente para cada base, considerando para cada uma delas os trechos de vôos já em operação a partir das mesmas, o que garantiria a disponibilidade de tripulantes suficientes para as rotações a serem construídos. Neste caso as restrições de base não seriam consideradas e o problema pode ser formulado exatamente como um problema de recobrimento ou particionamento, conforme foi feito em [5],[25].

No entanto, conforme ressaltado anteriormente, manter os trechos de vôos fixos às bases durante o processo de otimização pode impedir que sejam obtidas melhores soluções.

O algoritmo desenvolvido considera, no processo de construção das rotações, a possibilidade de atribuição das mesmas a qualquer uma das bases operadas pela empresa, ficando a possibilidade de incluir ou não as restrições de base na etapa de solução do problema de recobrimento, já que é possível a identificação da base relacionada a cada rotação construída.

Neste trabalho foram consideradas as restrições de base na etapa de otimização. No entanto uma abordagem que não as considera pode ser útil como uma ferramenta de simulação, no sentido de possibilitar uma redistribuição da carga de trabalho entre as bases operadas pela empresa. Desse modo pode-se observar os impactos em termos de custos que a redistribuição proporcionaria. Decisões como transferências de pessoal entre bases, ou mesmo eliminação de uma base ou implementação de outra nova teriam bastante suporte com esta abordagem.

5.3 Critérios de qualidade de uma rotação e função de custo adotada

Os critérios de qualidade para avaliação de uma rotação podem variar muito dependendo da empresa e sobretudo das características da malha de vôos operada pela mesma. De uma forma geral, uma solução satisfatória deve apresentar um baixo custo para a empresa sem no entanto comprometer o nível de satisfação dos tripulantes.

O critérios de qualidade considerados no presente trabalho foram os sugeridos pela empresa na qual foi feita a aplicação do algoritmo e envolvem basicamente:

- Período de duração das rotações: uma boa solução deve conter rotações de curta duração, pois além de minimizar os custos com pernoites fora da base, as rotações mais curtas facilitam o processo conseguinte de alocação efetiva de rotações aos tripulantes. Rotações curtas beneficiam a fase de operação, diminuindo os efeitos em cascata das modificações de contingência. Para o caso estudado foi dada preferência às rotações de até três dias.

- Presença de dias inativos: foi admitido o máximo de um dia de inatividade na rotação, somente quando não há a possibilidade de atribuição de quaisquer outros vôos naquele dia, nem de retorno do tripulante para a sua base domiciliar.

- Presença de vôos extras ("deadheads"): é dada preferência a vôos extras, que tragam o piloto de volta à sua base como passageiro no final do dia (ou o levem para o início da sua rotação, no início do dia), em detrimento a pernoites fora da base.

A função de avaliação adotada para a heurística considera os critérios de qualidade acima mencionados através de fortes penalizações às rotações que os violarem sem, no entanto, eliminá-las. É importante lembrar que, como os custos e penalizações adotados seguiram critérios sugeridos pela empresa, a função de avaliação adotada na heurística deve, portanto, ser modificada para cada caso tratado, refletindo a realidade do problema que se deseja resolver. A função incorpora também informações correspondentes a algumas parcelas de custos incorrentes nas rotações. Foi admitido o custo de uma rotação como o somatório dos custos de cada jornada de serviço. O custo de uma jornada de serviço foi calculado a partir das seguintes parcelas:

1. Aproveitamento do tripulante nas 11 horas de jornada diária: foi estabelecida uma penalização para as horas não voadas dentro da jornada.

   $\alpha * (t_d - \sum_{i \in I_d} t_i)$

   onde:

   α =custo do minuto de trabalho da tripulação;

   t_d =duração da jornada de serviço;

   t_i =duração do trecho de vôo i, incluindo os tempos de "briefing" e "debriefing";

   I_d =conjunto dos vôos contidos na jornada de serviço d;

2. Custos de pernoite: os custos referentes a pernoites fora da base foram incluídos nos jornadas de serviço cujos vôos iniciais ou finais não começam ou terminam, respectivamente, na base domiciliar do tripulante e representam o

custo com hotéis, alimentação e traslados. O custo de pernoite é função da cidade (H_c).

3. Custo referente a um dia de inatividade: o custo de uma jornada de serviço inativa foi considerado como o custo de 24 horas de trabalho da tripulação (C_{INAT}))

4. Custos referentes a vôos extras:;

   (a) Nos casos em que o vôo é realizado pela própria companhia foi adotado um custo equivalente a três vezes o tempo de trabalho do tripulante, ou seja, voar como passageiro representa o mesmo custo de ficar inativo durante a jornada por um período igual a três vezes o tempo de vôo do trecho em questão:

   $\alpha * \sum_{i \in E_d} 3 * t_i$

   onde:

   E_d = conjunto de vôos extras contidos na jornada d.

   (b) Quando é utilizado um "extra dummy", ou seja, um vôo de outra companhia, é adotada uma penalização equivalente a uma jornada de serviço inativa (C_{DUM})): $n * C_{DUM}$

   onde:

   n= número de "extras dummy" contidos na jornada;

O custo total da jornada de serviço foi, portanto, representado da seguinte maneira:

$$c_d = \alpha * [t_d - \sum_{i \in I_d} t_i + \sum_{i \in E_d} 3 * t_i] + H_c + n * C_{DUM}$$

5.4 Descrição do algoritmo

O algoritmo desenvolvido para a geração das rotações tem como base a metaheurística GRASP [30] e foi implementado em processamento paralelo devido ao alto custo computacional para a geração de um conjunto de rotações de grande porte.

A paralelização do algoritmo foi realizada através do MPI (Message Passing Interface), no modelo mestre/escravo.

Ao processo mestre foram atribuídas as tarefas de distribuir as sementes do GRASP entre os escravos; receber e armazenar cada rotação construída, ordenando-as segundo um critério de custo a fim de identificar e eliminar as rotações duplicadas.

Aos processos escravos foi atribuída a construção das rotações utilizando a metaheurística GRASP, com o objetivo de obter soluções que atendam a critérios de qualidade estabelecidos.

Descreveremos primeiramente o procedimento de construção das rotações propriamente dito, executado pelos processos escravos, em seguida apresentaremos o algoritmo executado pelo processo mestre e finalmente apresentaremos a metodologia geral de geração em paralelo.

5.4.1 Procedimento de construção das rotações

O procedimento para construção das rotações foi dividido em duas etapas: construção das jornadas de serviço e formação das rotações a partir das mesmas.

Algoritmo de construção das jornadas de serviço

Devido ao curto período de duração e às fortes restrições impostas às jornadas de serviço, foi utilizado um algoritmo de enumeração completa das mesmas, tal como foi feito em [10] e [25], e também em diversos outros trabalhos.

Seja $D = \{d_s : s \in S\}, S = \{1, 2, ..., q\}$, o conjunto de todas as as possíveis jornadas de serviço viáveis do problema. Utilizando a notação introduzida anteriormente, as jornadas de serviço são associadas a um conjunto $B = \{b_k : k \in K\}, K = \{1, 2, ..., r\}$, de bases. A base associada à jornada de serviço $d_s \in D$ é identificada por $b_{k_s} \in B$. O conjunto de jornadas de serviço associadas à base b_k é denotado por $d^k = \left\{d_s : s \in S^k\right\}$, onde $S^k \subseteq S$ é o conjunto dos índices destas jornadas.

De maneira análoga, denotamos o conjunto dos dias contidos no horizonte de planejamento $H = \{h_u : u \in U\}, U = \{1, 2, ..., v\}$. O conjunto de jornadas de serviço contidas no dia h_u é denotado por $d^u = \{d_s : s \in S^u\}$, onde $S^u \subseteq S$ é o conjunto dos índices destas jornadas.

Considerando os dois índices definidos denotamos $d^{ku} = \left\{d_s : s \in S^{ku}\right\}$, onde

$S^{ku} = S^k \bigcap S^u$.

Finalmente, o conjunto d^{ku} foi classificado segundo a posição das jornadas na rotação, resultando nos seguintes subconjuntos: di^{ku}, referente às jornadas de início da rotação, ou seja, que partem da base; dm^{ku}, associado às jornadas no meio da rotação; dt^{ku}, referente às jornadas de término da rotação, ou seja, que voltam para a base; e finalmente df^{ku}, constituído pelas jornadas que partem e retornam à base.

Considerou-se, conforme já destacado, que todos os trechos de vôos e consequentemente todas as jornadas de serviço construídas poderiam estar associadas a qualquer uma das bases, ficando a cargo do processo de otimização escolher a melhor configuração em termos de custo.

Dessa forma cada um dos conjuntos $d^k = \left\{ d_s : s \in S^k \right\}$, $k \in K$, contém todos os trechos de vôos da malha, sendo a única diferença entre estes os custos associados às jornadas.

Por exemplo, para um problema com 2 bases, b_1 e b_2, pode-se ter uma sequência de trechos de vôos para os quais seja possível construir: uma jornada do tipo df^1, atribuindo os trechos à base b_1; ou uma outra jornada dm^2, atribuindo os mesmos à base b_2. No entanto os custos associados às duas jornadas seriam diferentes. Como não sabemos *a priori* qual das duas estaria contida na melhor solução, optamos pela geração de todas as possibilidades.

Após a geração dos conjuntos acima descritos e cálculo dos custos associados, como descrito na seção 5.3, prosseguimos com a construção das rotações.

Algoritmo de construção das rotações

Adotando-se a notação introduzida anteriormente, onde F representa o conjunto de trechos de vôos a cobrir, denotou-se por $F_n \subset F$ o conjunto de trechos de vôos em aberto em cada iteração n do algoritmo. Denotou-se por t, a duração máxima, em dias, considerada para as rotações e por o_u o número de ordem do dia h_u na rotação, $o_u = 1, ..., t$.

A partir dos conjuntos de jornadas de serviços gerados, utilizou-se o seguinte procedimento para a geração das rotações:

Algoritmo (*CONSTRUÇÃO DE ROTAÇÕES*)

Passo 0 *(Início do algoritmo)* Inicialize a iteração $n = 0$.

Passo 1 *(Início das iterações)* Inicialize o índice da rotação $j_n = 0$, e faça $F_n = F$.

Passo 2 *(Escolha do dia de início)* Selecione aleatoriamente um dia de início $u \in U$ e faça $o_u = 1$.

Passo 3 *(Construção da lista de candidatos)* Se $o_u = 1$, ordene pelo custo todas as jornadas de serviço $d_s \in (di^u \bigcup df^u)$. Se $1 < o_u < t$, ordene as rotações $d_s \in (dm^{ku} \bigcup dt^{ku})$, que mantenham a viabilidade da rotação. Se $o_u = t$, ordene as rotações $d_s \in (dt^{ku})$, também verificando as condições de viabilidade. Denotou-se cd^u o conjunto ordenado resultante para os dois casos. Se $cd^u \neq \emptyset$ vá para o Passo 5.

Passo 4 *(Backtrackings)* Se $o_u = 1$, volte ao Passo 2, caso contrário faça $h_u = h_u - 1$, e vá para o Passo 6.

Passo 5 *(Construção da lista restrita de candidatos)* É estabelecida a lista restrita $rcd^u \subset cd^u$, que contém um percentual p das jornadas em cd^u que apresentam os menores valores de custo.

Passo 6 *(Escolha aleatória)* É escolhida aleatoriamente uma jornada de serviço $\bar{d}_s \in rcd^u$. Uma vez feita a escolha, se $o_u = 1$, faz-se $k = ks$, associando a rotação em construção à base b_{ks}.

Passo 7 *(Teste de fechamento da rotação)* Se $\bar{d}_s \in (di^{ku} \bigcup dm^{ku})$, a rotação ainda está em aberto, faça $h_u = h_u + 1$, $o_u = o_u + 1$, e vá para o Passo 3. Caso contrário, se $\bar{d}_s \in (dt^{ku} \bigcup df^{ku})$, a rotação já foi fechada, atualize o conjunto de trechos de vôos a cobrir: $F_n = F_n - F_j$ e envie ao processo mestre a rotação t_j.

Passo 8 *(Teste de fechamento de uma solução viável)* Se $F_n \neq \emptyset$, faça $j_n = j_n + 1$, e vá para o Passo 2, caso contrário, faça $n = n + 1$, se $n \leq MaxIt$, vá para o Passo 1.

No Passo 5, foi utilizado um percentual p variável ao longo das iterações, com $0,3 \leq p \leq 1$.

No Passo 7, se após a atualização, $hu \notin H$, faz-se $h_u = h_1$, ou seja, foi considerado um horizonte de planejamento circular: as rotações que têm início no final do período de planejamento podem se extender ao início do mesmo.

No Passo 6, cada vez que uma jornada de serviço é selecionada, seus trechos de vôos são marcados como já cobertos, porém as demais jornadas de serviço que contêm estes vôos não são retiradas do conjunto de candidatos, já que foi admitida a possibilidade de realização de vôos extras. É feita, no entanto, uma atualização de custos, fazendo com que todos os trechos que já foram cobertos passem a apresentar um custo de vôo extra nas demais jornadas de serviço candidatas.

Em relação ao Passo 4, cabe ressaltar que, de acordo com a dimensão do problema foram definidos limites para o número máximo de "backtrackings" realizados. Após atingidos estes limites, algumas relaxações foram feitas no que se refere aos critérios de qualidade adotados para a construção das listas de candidatos, como por exemplo, número máximo de extras por jornada e presença de dia inativo. No entanto, não é feita nenhuma relaxação que comprometa a viabilidade da rotação.

O algoritmo é iniciado adotando-se uma duração máxima t de três dias para as rotações, como um critério de qualidade a fim de favorecer à formação de rotações mais curtas que o permitido. No entanto, no caso em que são atingidos os limites de "backtrackings", se após feitas as relaxações dos critérios de qualidade, restarem vôos a serem cobertos, é acionada uma rotina que amplia a duração das rotações para cinco dias e realiza a construção das mesmas de forma guiada a encontrar os vôos restantes. Nesta fase a construção das rotações não mais é realizada necessariamente a partir dos seus dias iniciais (escolhidos aleatoriamente), mas sim nos dias que contiverem algum trecho de vôo que ainda não foi coberto. Se, por exemplo, algum vôo só estiver presente em jornadas de serviço de retorno à base, é feita a escolha a partir destas jornadas de serviço (seguindo também um critério de custo), e a construção da rotação segue na direção que se fizer necessária. Para esta fase não foi adotado nenhum movimento de backtracking.

Para os casos onde não são encontradas as jornadas de serviço de fechamento das rotações (que conectam as jornadas de serviço intermediárias com a base no início e no fim da rotação), é admitido, nestes casos, um vôo dummy que fará as conexões necessárias, o que poderia corresponder na prática a um deslocamento do tripulante

como passageiro em um vôo operado por outra companhia.

Finalmente, no Passo 8, quando $F_n = \emptyset$, temos todos os vôos cobertos, ou seja, a cada iteração, tem-se uma solução viável do problema.

Note que o algoritmo descrito corresponde a um procedimento heurístico de encontrar caminhos em um grafo de jornadas de serviços. No entanto, não são gerados todos os arcos possíveis. Escolhe-se inicialmente um nó, correspondente à jornada de serviço inicial, e somente são gerados os arcos que conectam este nó a uma próxima jornada viável. O mesmo é feito quando é escolhida a segunda jornada e assim sucessivamente. Ou seja, não são exploradas todas as possibilidades de caminhos, como é feito em algoritmos exatos de caminhos mínimos.

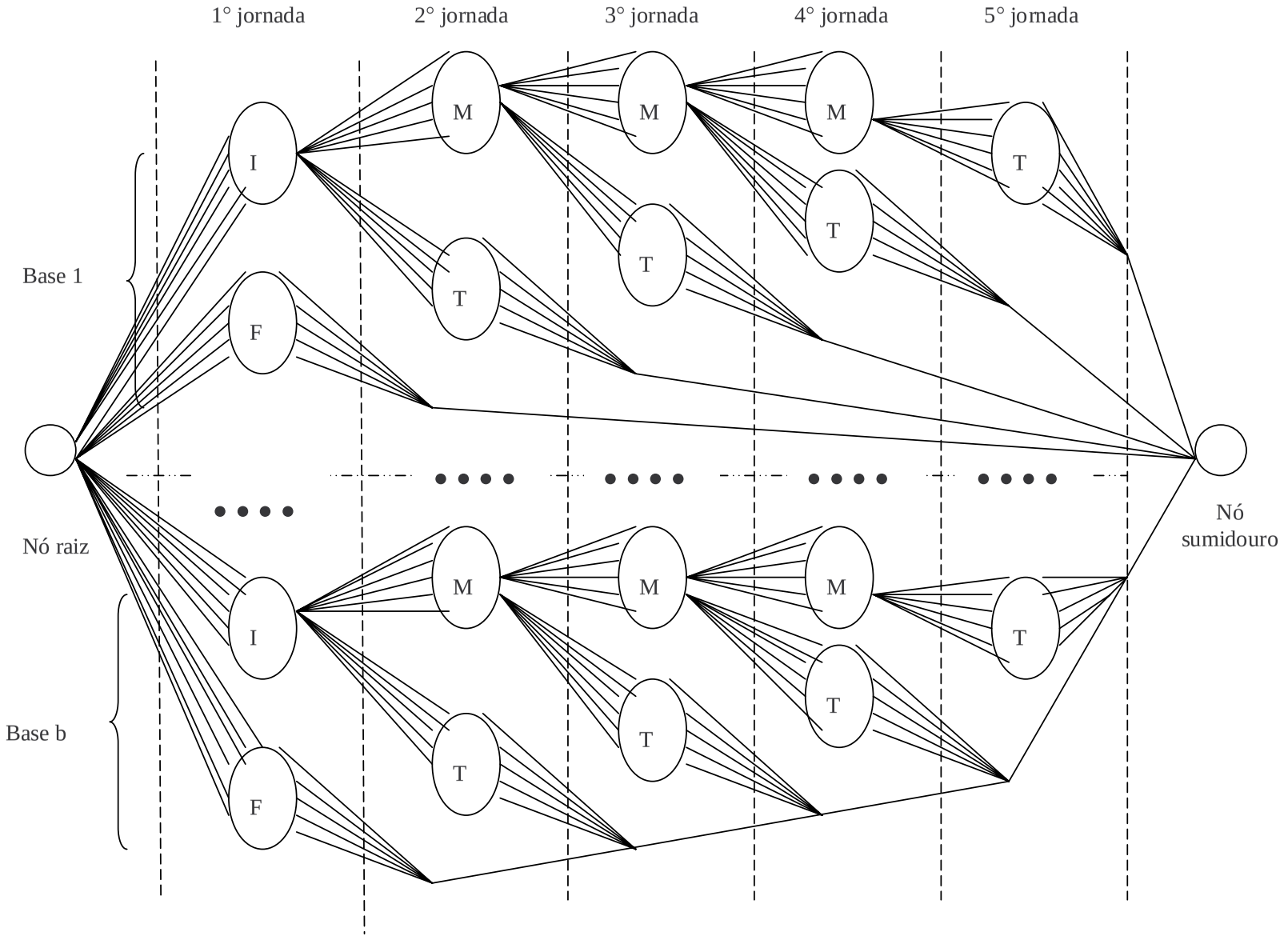
No entanto, para o porte do problema que se propôs resolver, a alternativa de um algoritmo exato, que permitiria a utilização de uma abordagem de geração de colunas, é praticamente inviável considerando os recursos computacionais disponíveis, como veremos a seguir.

Considerando a duração máxima de cinco dias das rotações, o grafo para o caso estudado, denotado $G(N, A)$ seria composto por cinco "camadas" de nós, entre um nó fonte, de onde partem os arcos para as jornadas do início das rotações, e um nó sumidouro, que representa o fim das mesmas, conforme ilustra a figura 5.4.1

Convém destacar que, para cada base b_k, tem-se um subgrafo com todas as jornadas de serviço viáveis para esta base. Inicialmente, quando é feita a escolha do primeiro nó (primeira jornada), considera-se o total de nós na primeira camada para todas as bases, mas uma vez feita a primeira escolha, prossegue-se no subgrafo correspondente à base associada à jornada escolhida.

A tabela 5.1 apresenta o número de nós e arcos do grafo. Considerando o índice k correspondente ao subgrafo asociado à base b_k, foi denotado N_i^k o número total de nós na camada i (referente a i-ésima jornada de serviço da rotação), A_{ij}^k, o número de arcos entre as camadas i e j, A_f^k, o número de arcos que saem do nó fonte e A_s^k, o número de arcos que chegam no nó sumidouro.

Cabe destacar que os números de arcos apresentados representam as possibilidades de conexão entre os nós, antes de serem verificadas as condições de viabilidade destas conexões. Após a verificação destas condições, o número de arcos seria significativamente reduzido. Para a modelagem do problema de geração das rotações


1° jornada
2° jornada
3° jornada
4° jornada
5° jornada
Base 1
Base b
Nó raiz
Nó sumidouro
I
F
M
T

em questão, teríamos que gerar um grafo $\bar{G}(N, \bar{A})$, onde $\bar{A}$ corresponderia aos arcos representando todas as conexões viáveis entre as jornadas, construídos a partir da verificação de viabilidade dos arcos em A. No entanto, diante do porte do grafo G, pode-se constatar que, para este caso, seria praticamente inviável este tipo de abordagem.

Optou-se, portanto pela metodologia apresentada, onde, conforme já destacado, são gerados apenas os arcos necessários para dar seguimento a um caminho escolhido de forma heurística.

5.4.2 O procedimento de comparação e armazenamento das rotações

Antes de detalhar o algoritmo, faremos uma breve descrição da árvore AVL.

A árvore AVL foi apresentada por Adelson-Velskii e Landis em 1962. Trata-se de uma árvore de busca binária que é balanceada em relação à altura das sub-árvores. A grande contribuição na concepção da árvore AVL foi a diminuição da complexidade dos algoritmos de busca e inserção. Para uma árvore de n nós, pode-se realizar uma busca com complexidade O(log(n)).

Pode-se definir uma árvore balanceada por altura da seguinte forma:

Toda árvore binária vazia é balanceada por altura, e seja uma árvore binária A, não vazia, e sejam A_e e A_d suas sub-árvores da esquerda e direita, então A é balanceada por altura se:

- A_e e A_d são balanceadas por altura;
- $\mid h_e - h_d \mid \leq 1$, onde h_e e h_d são as alturas de A_e e A_d respectivamente.

O algoritmo implementado para armazenamento das rotações inclui procedimentos de busca, comparação das rotações e inserção balanceada, e foi concebido com o objetivo de tornar mais rápida a eliminação das rotações duplicadas.

A estrutura foi constituída de dois componentes básicos: uma lista duplamente encadeada ζ de vetores representando as rotações ordenadas pelo custo, e uma árvore AVL, onde cada nó N_i contém três componentes:

- um valor de custo C_i;

- dois ponteiros P_i^c e P_i^f apontando respectivamente para o começo e o fim da região da lista encadeada que contém as rotações que apresentam o custo C_i

A cada rotação recebida apresentando um custo C_t, o processo mestre busca na árvore um nó que apresente este valor de custo. Como a árvore é mantida sempre balanceada, a complexidade da busca é sempre O(log(n)).

Se não for encontrado nenhum nó, isto significa que a rotação em questão não é duplicada. O nó representando o custo da rotação é inserido de forma balanceada na árvore.

Após a inserção do nó, a rotação deve ser inserida na lista de modo que a mesma permaneça ordenada pelo custo. Para isso utilizamos os ponteiros de delimitação de trechos P_i^c e P_i^f contidos na estruturas do nós adjacentes ao nó inserido. Por exemplo, se a rotação a ser inserida tem um custo C_t, e $C_a < C_t < C_b$, onde C_a e C_b são custos de rotações contidas na lista, a inserção é feita entre os trechos de custo C_a e C_b,utilizando o ponteiro de fim de trecho do nó a. O mesmo procedimento é feito quando $C_t > C_a$, e C_a é o maior custo das rotações na lista, neste caso a rotação é inserida no final da lista. No caso em que $C_t < C_b$, e C_b é o menor custo da lista, utiliza-se o ponteiro de início de trecho do nó b e insere-se a rotação no início da lista.

A figura 5.1 ilustra o procedimento descrito:

Se for encontrado um nó de mesmo custo, é realizada uma busca na região da lista compreendida entre P_i^c e P_i^f. Se a rotação for idêntica a alguma das rotações já armazenadas a mesma é descartada, caso contrário, faz-se a inserção da mesma na região da lista em questão. O procedimento é repetido até que seja armazenado o número desejado de rotações $NumRot$.

É interessante notar que se conseguiu, através do uso da árvore AVL, reduzir a complexidade de pior caso de busca e inserção ordenada na lista ζ, de O(n) para O(log(n)). Como o objetivo foi desenvolver um algoritmo aplicável a problemas de grande porte, essa diferença é bastante significativa.

Para o caso estudado, o maior conjunto gerado foi de aproximadamente 11 milhões de rotações. O número de operações necessárias para a inserção ordenada de um elemento na lista, quando esta atinge a ordem de 11 milhões de elementos, seria, no pior caso, 11 milhões. Com o uso da árvore AVL, o número de operações cai para

16. Dessa forma conseguiu-se tornar viável a geração de conjuntos de grande porte de rotações distintas.

O procedimento de armazenamento das rotações pode ser descrito da seguinte forma:

Algoritmo (*COMPARAÇÃO E ARMAZENAMENTO*)

Passo 0 *(Início do algoritmo)* Inicialize o índice da rotação $j = 0$ e a árvore AVL, inicialmente vazia.

Passo 1 *(Recebimento da rotação)* Receba uma rotação t_j, de custo C_{t_j}, e busque na árvore AVL um nó N_i com $C_i = C_{t_j}$. Se existir o nó N_i , vá para o Passo 6.

Passo 2 *(Inserção do Nó)* Insira o nó N_i na árvore AVL. Tome NP_i, o nó pai do nó N_i, e ${P^c}_{(NP_i)}$ e ${P^f}_{(NP_i)}$, os respectivos ponteiros que indicam a posição do trecho da lista ζ com rotações de custo C_{NP_i}.

Passo 3 *(Inserção da rotação - caso 1)* Se $C_i < C_{NP_i}$, insira a rotação t_j na lista ζ antes do elemento apontado por ${P^c}_{(NP_i)}$. Faça ${P_i}^c$ e ${P_i}^f$ apontarem para a rotação t_j. Vá para o Passo 5.

Passo 4 *(Inserção da rotação- caso 2)* Se $C_i > C_{NP_i}$, insira a rotação t_j na lista ζ depois do elemento apontado por ${P^f}_{(NP_i)}$. Faça ${P_i}^c$ e ${P_i}^f$ apontarem para a rotação t_j.

Passo 5 *(Balanceamento da AVL)* Verifique se a árvore AVL está balanceada. Se não estiver, proceda com as operações de balanceamento. Vá para o Passo 7.

Passo 6 *(Verificação e Inserção da rotação)* Verifique no trecho da lista ζ compreendida entre os elementos apontados pelos ponteiros ${P_i}^c$ e ${P_i}^f$ se existe alguma rotação idêntica a t_j. Se não existir, insira a rotação t_j na lista ζ depois do elemento apontado por ${P_i}^c$, caso ${P_i}^c = {P_i}^f$, faça ${P_i}^f$ apontar para o elemento t_j.

Passo 7 Faça $j = j + 1$. Se $j \leq NumRot$ vá para o Passo 1.

5.4.3 O algorimo paralelo de geração das rotações

A paralelização do algoritmo foi realizada através do MPI, no modelo mestre/escravo.

Seja $Pr = \{pr_i : i \in I\}, I = \{1, 2, ..., numproc\}$, o conjunto dos processos a serem executados. Atribuímos ao processo mestre o identificador $rank_i = 0$, e aos processos escravos $rank_i > 0$.

A seguir descrevemos o procedimento adotado.

Algoritmo (*GERADOR PARALELO*)

Passo 0 *(Mestre)* Se $rank_i = 0$, gere o vetor $\lambda \in \Re^{|I|-1}$ de sementes aleatórias, envie λ_k a todo processo $pr_k \in Pr$, com $rank_k > 0$ e inicialize a execução do algoritmo *COMPARAÇÃO E ARMAZENAMENTO*.

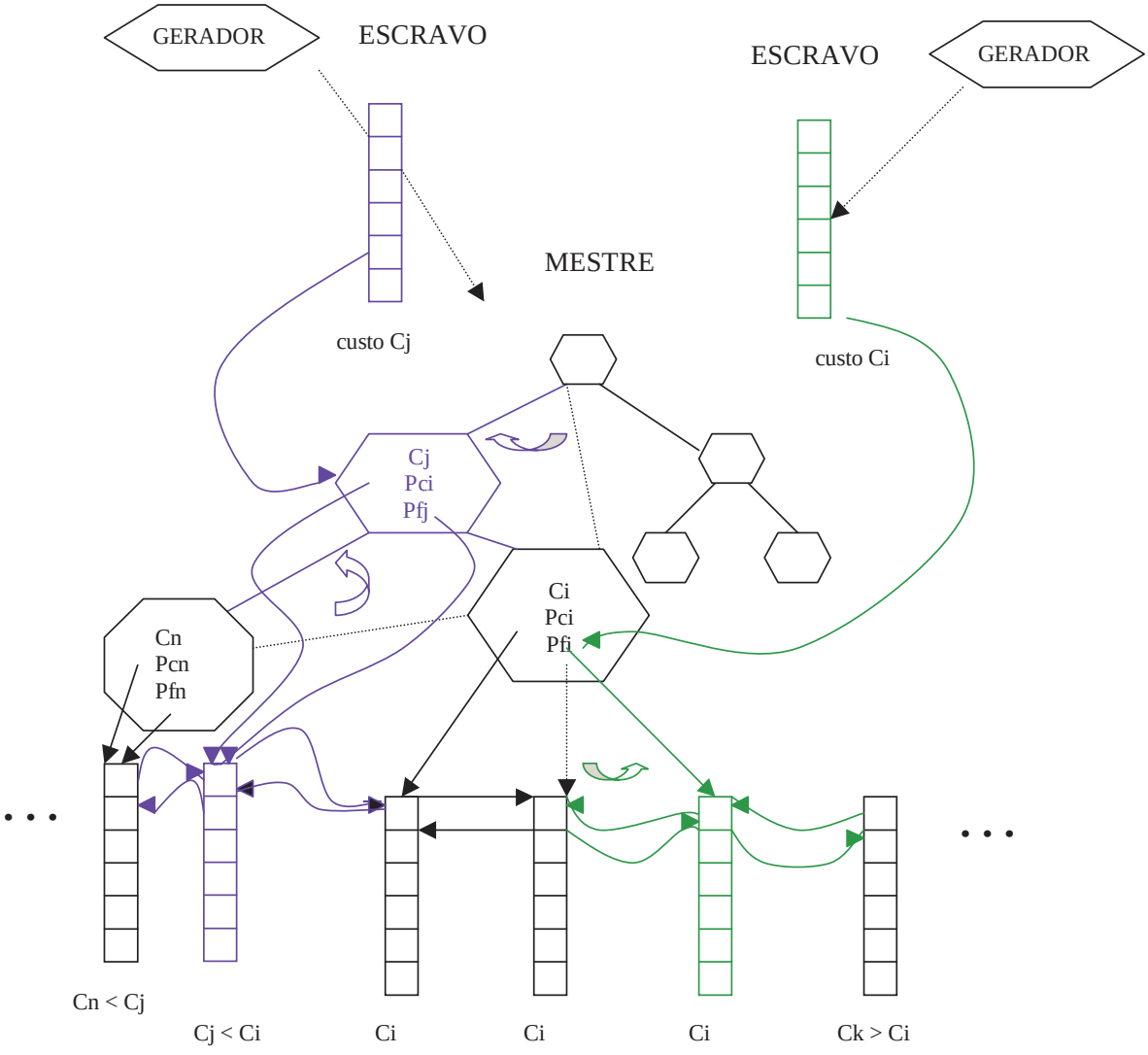
Passo 1 *(Escravo)* Se $rank_i > 0$, receba a semente λ_i e inicialize a execução do algoritmo *CONSTRUÇÃO DE ROTAÇÕES*.

A geração de rotações em paralelo foi utilizada em [22], onde são citados outros trabalhos utilizando também esta abordagem.

Ao mesmo tempo que proporcionou uma diminuição significativa do tempo de processamento, a paralelização do algoritmo possibilitou uma redução da memória de armazenamento descartando as rotações duplicadas no momento em que são geradas.

Bases	Jornadas de serviços		Número de nós nas camadas		Número de arcos entre as camadas	
base b_1	di^1	15375	N_1^1	19144	A_f^1	19144
	dm^1	115136	N_2^1	125500	A_{12}^1	1929562500
	dt^1	10364	N_3^1	125500	A_{23}^1	$1,445 \cdot 10^{10}$
	df^1	3769	N_4^1	125500	A_{34}^1	$1,445 \cdot 10^{10}$
			N_5^1	10364	A_{45}^1	1193269504
					A_s^1	45225
base b_2	di^2	16678	N_1^2	18885	A_f^2	18885
	dm^2	112750	N_2^2	125759	A_{12}^2	2097408602
	dt^2	13009	N_3^2	125759	A_{23}^2	$1,4179 \cdot 10^{10}$
	df^2	2207	N_4^2	125759	A_{34}^2	$1,4179 \cdot 10^{10}$
			N_5^2	13009	A_{45}^2	1466764750
					A_s^2	54243
base b_3	di^3	11244	N_1^3	12037	A_f^3	12037
	dm^3	124225	N_2^3	132607	A_{12}^3	1491033108
	dt^3	8382	N_3^3	132607	A_{23}^3	$1,6473 \cdot 10^{10}$
	df^3	793	N_4^3	132607	A_{34}^3	$1,6473 \cdot 10^{10}$
			N_5^3	8382	A_{45}^3	1041253950
					A_s^3	34321
base b_4	di^4	9170	N_1^4	10076	A_f^4	10076
	dm^4	124261	N_2^4	134568	A_{12}^4	1233988560
	dt^4	10307	N_3^4	134568	A_{23}^4	$1,6722 \cdot 10^{10}$
	df^4	906	N_4^4	134568	A_{34}^4	$1,6722 \cdot 10^{10}$
			N_5^4	10307	A_{45}^4	1280758127
					A_s^4	42134
base b_5	di^5	16287	N_1^5	17782	A_f^5	17782
	dm^5	116515	N_2^5	126862	A_{12}^5	2066201394
	dt^5	10347	N_3^5	126862	A_{23}^5	$1,4781 \cdot 10^{10}$
	df^5	1495	N_4^5	126862	A_{34}^5	$1,4781 \cdot 10^{10}$
			N_5^5	10347	A_{45}^5	1205580705
					A_s^5	42883
base b_6	di^6	3745	N_1^6	5107	A_f^6	5107
	dm^6	136903	N_2^6	139537	A_{12}^6	522566065
	dt^6	2634	N_3^6	139537	A_{23}^6	$1,9103 \cdot 10^{10}$
	df^6	1362	N_4^6	139537	A_{34}^6	$1,9103 \cdot 10^{10}$
			N_5^6	2634	A_{45}^6	360602502
					A_s^6	11898

Tabela 5.1: Número de jornadas de serviço, nós e arcos do grafo G


GERADOR
ESCRAVO
ESCRAVO
GERADOR
MESTRE
custo Cj
custo Ci
Cj
Pci
Pfj
Ci
Pci
Pfi
Cn
Pcn
Pfn
Cn < Cj
Cj < Ci
Ci
Ci
Ci
Ck > Ci

Capítulo 6

Otimização das rotações

Uma vez gerado o conjunto de rotações com o qual se vai trabalhar, é proposto um algoritmo de solução aproximada para o PCR resultante.

Antes de descrevermos o algoritmo proposto introduziremos a notação utilizada na sua descrição.

Seja P o subconjunto gerado de colunas tal que $P \subseteq T$, onde T é o conjunto de todas a rotações viáveis. O modelo para o PCR definido na página 16 composto pelo subconjunto de colunas em P pode ser escrito através da seguinte notação $\min\left\{c^T x \mid Ax \geq b, x \in \{0,1\}^n\right\}$, onde $m = m_i + m_k$ representa o número de linhas da matriz A, m_i representa o número de linhas associadas às restrições de cobertura dos vôos e m_k representa o número de linhas associadas às restrições de base. Denotamos (RL) a relaxação linear deste problema definida por $\min\left\{c^T x \mid Ax \geq b, x \in \Re^n\right\}$.

Para a solução do problema descrito, foi utilizada uma combinação de algoritmos. Apresentaremos cada um deles separadamente e em seguida descreveremos o algoritmo final.

6.1 Obtenção do limite inferior

Para a obtenção do limite inferior para o PCR foram utilizados como base: a abordagem SPRINT [3], o método PRIMAL-DUAL por subproblemas [20], e ainda foi utilizada uma heurística para a obtenção de uma solução dual viável inicial. Descreveremos cada um deles e mais adiante apresentaremos como os mesmos foram inseridos na abordagem de solução desenvolvida.

6.1.1 Algoritmo por Subproblemas - SPRINT

Este algoritmo tem por finalidade encontrar a solução ótima para o problema (RL). Ressalta-se que em [3], Anbil *et al.* utilizam um valor constante para n_k.

Algoritmo (*SPRINT*)

Passo 0 Inicialize $k = 0$. Selecione aleatoriamente um conjunto A_k com n_k colunas da matriz A.

Passo 1 Seja P_k um subproblema de (RL) formado pela matriz A_k. Seja

$$\bar{x}_k = \text{argmin}\left\{ {c_k}^T x_k \mid A_k x_k \geq b, x_k \in \Re^{n_k} \right\},$$

μ_k seu dual ótimo associado e B_k a respectiva base primal viável associada a $\bar{x}_k$. Esta base também é primal viável para (RL) pois $x = [\bar{x}_k^T \quad 0]^T \in \{Ax \geq b, x \in \Re^n\}$.

Passo 2 Seja N o conjunto de colunas não básicas de (RL) relativas a B_k. Se $c_N - \mu_k N \geq 0$ então o vetor $x = [\bar{x}_k^T \quad 0]^T$ será uma solução ótima de (RL). PARE.

Passo 3 Faça $k = k+1$. Selecione um conjunto S_k com $n_k - m$ colunas da matriz N, onde m é o número de colunas da matriz B_{k-1}. Este conjunto de colunas é escolhido tomando-se as $n_k - m$ colunas de N com os menores valores de $c_N - \mu_{k-1} N$. A nova matriz A_k será então $[B_{k-1} \quad S_k]$. Vá para o passo 1.

6.1.2 Algoritmo Primal-Dual por Subproblemas

O algoritmo Primal-Dual por subproblemas é uma modificação do algoritmo Primal-Dual simplex original que incorpora a abordagem SPRINT de solução por subproblemas. Este algoritmo foi proposto por Hu e Johnson em [20], onde são apresentados resultados computacionais para problemas de grande porte de Construção de Rotações.

Assim como no método Primal-Dual original, parte-se de uma solução dual viável inicial, na qual são realizadas sucessivas melhoras a cada iteração, sem que a mesma perca a viabilidade. No entanto, a cada iteração são adicionados blocos de colunas

ao problema primal, como veremos a seguir, enquanto que no Primal-Dual original adiciona-se apenas uma coluna por vez. Segundo Hu e Johnson, para problemas de grande porte esta abordagem funciona significativamente melhor.

Pode-se descrever o referido algoritmo da seguinte forma:

Algoritmo (*Primal-Dual por Subproblemas*)

Passo 0 Inicialize $k = 0$. Encontre um vetor π dual viável para (RL). Selecione um conjunto A_k com n_k colunas da matriz A.

Passo 1 Seja P_k um subproblema de (RL) formado pela matriz A_k. Seja

$$\bar{x}_k = \text{argmin}\left\{ {c_k}^T x_k \mid A_k x_k \geq b, x_k \in \Re^{n_k} \right\},$$

ρ_k seu dual ótimo associado e B_k a respectiva base primal viável associada a $\bar{x}_k$. Esta base também é primal viável para (RL) pois $x = [\bar{x}_k^T \quad 0]^T \in \{Ax \geq b, x \in \Re^n\}$.

Passo 2 Seja $\pi^T b$ o valor da função dual de (RL) no ponto π e ${c_k}^T \bar{x}_k$ o valor da função primal no ponto $x = [\bar{x}_k^T \quad 0]^T$. Se $\pi^T b = {c_k}^T \bar{x}_k$, então o vetor x e o vetor π são soluções ótimas de (RL). PARE.

Passo 3 Seja N o conjunto de colunas não básicas de (RL) relativas a B_k. Se $c_N - \rho_k N \geq 0$ então o vetor $x = [\bar{x}_k^T \quad 0]^T$ será uma solução ótima de (RL) e o vetor ρ_k uma solução ótima dual viável de (RL). PARE.

Passo 4 Fase de melhoria do vetor dual viável atual π. Encontre um escalar $\alpha \in \{\alpha \in \Re \mid 0 \leq \alpha \leq 1\}$ tal que o vetor $\tilde{\pi}$ formado pela combinação convexa entre π e ρ_k, $\tilde{\pi} = (1-\alpha)\pi + \alpha\rho_k$, seja dual viável para (RL), ou seja, $\tilde{\pi}A \leq c$ e $b^T\tilde{\pi}$ seja máximo. Faça $\pi = \tilde{\pi}$.

Passo 5 Faça $k = k+1$. Selecione um conjunto S_k com $n_k - m$ colunas da matriz N, onde m é o número de colunas da matriz B_{k-1}. Este conjunto de colunas é escolhido tomando-se as $n_k - m$ colunas de N com os menores valores de $c_N - \pi_{k-1}N$. A nova matriz A_k será então $[B_{k-1} \quad S_k]$. Vá para o passo 1.

Um algoritmo paralelo primal-dual por subproblemas foi proposto por Klabjan, Johnson e Nemhauser [24] e foi utilizado em [22] para encontrar a relaxação linear

de problemas de Construção de Rotações em instancias com o número de colunas em torno de 50 milhões e com um número de linhas inferior a 1000. A seguir esclareceremos o Passo 4 do algoritmo referente a fase de melhoria da solução dual.

Cálculo do passo α

Na etapa referente ao cálculo do passo α, dois pontos são fundamentais, o novo vetor dual obtido $\tilde{\pi}$ deve ser dual viável para (RL), ou seja $\tilde{\pi}A \leq c$, e o seu valor deve ser o melhor possível na função objetivo do problema dual $\tilde{\pi}^T b$. Dessa forma podemos concluir que, lembrando que $\bar{c}_\mu$ é o vetor de custos reduzidos associados ao vetor dual μ ($\bar{c}_\mu = c - \mu A$):

$$\left((1-\alpha)\pi + \alpha\rho_k\right) A \leq c \Rightarrow \pi A - \alpha\pi A + \alpha\rho_k A \leq c \Rightarrow \alpha(\rho_k A - \pi A) \leq c - \pi A \Rightarrow$$

$$\alpha\left[(c - \bar{c}_{\rho_k}) - (c - \bar{c}_\pi)\right] \leq c - \pi A \Rightarrow \alpha\left(\bar{c}_\pi - \bar{c}_{\rho_k}\right) \leq \bar{c}_\pi \quad \text{(I)}$$

Tendo-se obtido a relação (I), podemos proceder com o cálculo de α. Como o vetor dual ρ_k está associado a uma base B_k primal viável, sabemos pelo teorema da dualidade fraca que ${\rho_k}^T b \geq \pi^T b$. Sendo assim o maior valor que $\tilde{\pi}^T b$ poderia assumir seria dado pelo valor máximo de α pois:

$$\tilde{\pi}^T b = \left((1-\alpha)\pi + \alpha\rho_k\right)^T b = \pi^T b + \alpha({\rho_k}^T b - \pi^T b) \quad \text{(II)}$$

De (I) e (II) concluimos que o valor ótimo de α seria:

$$\alpha = \max\left(0, \frac{\bar{c}_\pi}{(\bar{c}_\pi - \bar{c}_{\rho_k})}\right)$$

6.1.3 Heurística para a obtenção de uma solução dual viável

Nesta seção descreveremos o método utilizado para se determinar uma solução dual viável inicial para (RL) que será utilizada no algoritmo Primal-Dual por Subproblemas.

Seja (DRL) o problema dual de (RL) definido por $\max\left\{b^T\mu \mid \mu A \leq c, \mu \in \Re^m\right\}$. A heurística apresentada a seguir busca construir uma solução viável para (DRL) de boa qualidade.

Denote por μ_A o conjunto das componentes μ_i do vetor μ que, se aumentadas de valor, tornam inviável alguma restrição de (DRL). Seja também μ^k o conjunto das componentes μ_i do vetor μ com coeficientes não nulos na k-ésima restrição de (DRL) e suas partições ${\mu^k}_a = \mu_A \cap \mu^k$ e ${\mu^k}_n = \mu^k - {\mu^k}_a$.

Algoritmo (*Heurística - Dual Inicial*)

Passo 0 Inicialize $\mu_A = \emptyset$. Faça $\mu_i = 0$ para todo $i \in \{1, ..., m\}$ e $k = 1$;

Passo 1 Se $\sum_{\mu_i \in {\mu^k}_a} \mu_i \leq c_k$, atribua valores às componentes μ_i, $\mu_i \in {\mu^k}_n$ tais que se verifique a relação $\sum_{\mu_i \in {\mu^k}_n} \mu_i = c_k - \sum_{\mu_i \in {\mu^k}_a} \mu_i$ e atualize $\mu_A = \mu_A \cup {\mu^k}_n$. Vá para o Passo 3;

Passo 2 Se $\sum_{\mu_i \in {\mu^k}_a} \mu_i > c_k$, atribua valores as componentes μ_i tais que se verifique a relação $\sum_{\mu_i \in {\mu^k}_a} \mu_i = c_k$ e atualize $\mu_A = \mu_A \cup {\mu^k}_n$;

Passo 3 Se $k < m$, faça $k = k + 1$ e vá para o Passo 1.

Passo 4 PARE. Retorne μ.

Observe que no passo 1, o valor em "excesso" $c_k - \sum_{\mu_i \in {\mu^k}_a} \mu_i$ é repartido igualmente entre as componentes μ_i, $\mu_i \in {\mu^k}_n$.

No passo 2, nenhuma das componentes μ_i, $\mu_i \in {\mu^k}_a$ podem ter seus valores aumentados, pois poderiam tornar inviáveis outras restrições já verificadas.

6.2 Obtenção de soluções inteiras

Para a obtenção de soluções inteiras de "boa qualidade" para o PCR, foi desenvolvido o algoritmo de Rubin Restrito, baseado no método de Rubin [31], descrito na seção 3.0.2. Antes de descrevermos o algoritmo de Rubin Restrito, apresentaremos de uma maneira mais formal o algoritmo de Rubin original, para que fique mais clara a modificação realizada.

Algoritmo (*Algoritmo de Rubin*)

Passo 0 Seja x_{int} uma solução inteira do problema (PCR) e A_{int} a submatriz da matriz A correspondente às colunas associadas a esta solução inteira. Inicialize $p = 0$;

Passo 1 Selecione aleatoriamente uma submatriz $A_p \subseteq A_{int}$ com um número $NCol$ de colunas. Seja x_p o vetor contendo as componentes de x_{int} associadas a A_p e $ValA_p$ o valor da parcela da função objetivo do problema (PCR) correspondente às colunas em A_p. Seja $m_p \leq m_i$ o número de restrições de recobrimento dos vôos do (PCR) que ficam inviáveis após a retirada do conjunto de colunas A_p;

Passo 2 Gere TODAS as n_p possíveis rotações viáveis para o conjunto de vôos representados pelas m_p linhas;

Passo 3 Defina $R_p = \min\left\{{c_p}^T x \mid D_p x \geq b_p, x \in \{0,1\}^{n_p}\right\}$ o (PCR) resultante das n_p rotações geradas que compõem a matriz D_p, com $m_p + m_k$ linhas;

Passo 4 Seja então $\tilde{x}_p$ a solução deste problema, $\tilde{D}_p$ as colunas da matriz D_p referentes a esta solução inteira e $ValR_p = {c_p}^T \tilde{x}_p$ o seu valor na função objetivo;

Passo 5 Se $ValR_p < ValA_p$ então faça $A_{int} = [A_{int}/A_p \;\; \tilde{D}_p]$ e $x_{int} = [x_{int}/x_p \;\; \tilde{x}_p]$;

Passo 6 Faça $p = p + 1$. Se $p \leq MaxInt$, vá para o Passo 1;

Passo 7 PARE. Retorne x_{int}.

Conforme destacado em [2], uma das principais limitações do método de Rubin está na necessidade prática de se trabalhar com subproblemas associados ao recobrimento de um número reduzido de vôos (m_p pequeno), uma vez que se propõe gerar todas as rotações viáveis associadas a estes vôos. Esta limitação aumenta o risco de ficarmos restritos a mínimos locais de baixa qualidade.

Uma forma de minimizar esta limitação é possibilitar a solução de subproblemas com um número maior de vôos. No entanto, neste caso não seria viável a geração de todas as possibilidades de rotações. Foi neste sentido que foi concebido o algoritmo de Rubin Restrito, descrito a seguir.

6.2.1 Algoritmo de Rubin Restrito

O algoritmo de Rubin restrito difere do algoritmo original basicamente na fase de composição dos subproblemas, Passo 2, onde, ao invés de serem geradas todas as possíveis rotações que cobrem os vôos desalocados, são selecionadas colunas de um conjunto previamente gerado.

A idéia consiste em limitar o tamanho dos subproblemas. Trata-se, portanto, de uma variação do método de Rubin, restrito a um conjunto considerado representativo de colunas (geradas, por exemplo, com descrito no capítulo 5).

Esta nova abordagem do método de Rubin permite trabalhar com um número maior de vôos a cada iteração com a garantia de que o número total de possíveis rotações para o subproblema a ser resolvido não ultrapassará um determinado limite (número de colunas do conjunto gerado). Esta possibilidade de se trabalhar com subconjuntos de maior porte proporcionou, assumindo que o conjunto de rotações previamente gerado é de "boa qualidade", um caráter mais global ao algoritmo.

Vale também ressaltar que o algoritmo de Rubin restrito pode ser aplicado a qualquer problema de partição/recobrimento, constituindo assim uma nova heurística para esses problemas.

O algoritmo de Rubin Restrito pode ser descrito da seguinte forma:

Algoritmo (*Algoritmo de Rubin Restrito*)

Passo 0 Seja x_{int} uma solução inteira do problema (PCR) e A_{int} a submatriz da matriz A correspondente às colunas associadas a esta solução inteira. Inicialize $p = 0$;

Passo 1 Selecione aleatoriamente uma submatriz $A_p \subseteq A_{int}$ com um número $NCol$ de colunas. Seja x_p o vetor contendo as componentes de x_{int} associadas a A_p e $ValA_p$ o valor da parcela da função objetivo do problema (PCR) correspondente às colunas em A_p. Seja $m_p \leq m_i$ o número de restrições de recobrimento dos vôos do (PCR) que ficam inviáveis após a retirada do conjunto de colunas A_p;

Passo 2 Selecione no conjunto P todas as rotações que contenham os vôos representados pelas m_p linhas;

Passo 3 Defina $R_p = \min\left\{{c_p}^T x \mid D_p x \geq b_p, x \in \{0,1\}^{n_p}\right\}$, o (PCR) resultante das n_p rotações selecionadas que compõem a matriz D_p, com $m_p + m_k$ linhas;

Passo 4 Seja então $\tilde{x}_p$ a solução deste problema considerando-se uma tolerância TolBB, $\tilde{D}_p$ as colunas da matriz D_p referentes a esta solução inteira e $ValR_p = {c_p}^T \tilde{x}_p$ o seu valor na função objetivo;

Passo 5 Se $ValR_p < ValA_p$ então faça $A_{int} = [A_{int}/A_p \;\; \tilde{D}_p]$ e $x_{int} = [x_{int}/x_p \;\; \tilde{x}_p]$;

Passo 6 Faça $p = p + 1$. Se $p \leq MaxInt$, vá para o Passo 1;

Passo 7 PARE. Retorne $\hat{x}_{int} = [(x_{int})^T \;\; 0]^T$.

6.3 Algoritmo Híbrido

Com base nos algoritmos apresentados, concebeu-se um algoritmo híbrido para a solução do PCR. Este algoritmo corresponde a união do algoritmo Primal-Dual por Subproblemas, inicializado a partir de uma solução dual gerada através da heurística apresentada, e da heurística Rubin Restrito desenvolvida. Estes algoritmos foram inseridos em um esquema de otimização global para o problema PCR que será descrito mais adiante.

Um ponto importante a ser citado é que foi feita uma pequena modificação no algoritmo Primal-Dual por Subproblemas para contornar os casos em que não existe melhoria do ponto dual viável atual, ou seja, $\alpha = 0$. No esquema de otimização global adotado, as seguintes variáveis são definidas:

1. GapRel - Variável que guarda o valor do Gap Relativo entre a função objetivo primal e dual, ou seja, para uma dada iteração k tem-se:
$$\text{GapRel} = \frac{{c_k}^T \bar{x}_k - \pi^T b}{{c_k}^T \bar{x}_k}$$

2. GapInt - Variável que guarda o Gap Relativo entre a melhor solução inteira encontrada até o momento, $\bar{x}_{int}$, e o valor do limite superior para a relaxação linear do problema, ou seja, $\bar{x}_k$. Sendo assim, para uma dada iteração k tem-se:
$$\text{GapInt} = \frac{c^T \bar{x}_{int} - {c_k}^T \bar{x}_k}{{c_k}^T \bar{x}_k}$$

3. TolRel - Valor da tolerância estabelecida para a relaxação linear. Nos exemplos estudados foi adotada uma tolerância de 0.03;

4. TolInt - Tolerância adotada para o valor de GapInt. Quando ao longo das iterações do algoritmo, GapInt > TolInt, então um esquema de melhoramento da solução inteira atual, $\bar{x}_{int}$, é utilizado no intuito de se obter um novo limite superior para o PCR de tal forma que GapInt $\leq$ TolInt. Nos exemplos estudados foi adotada uma tolerância de 0.03;

5. TolBB - Valor da tolerância estabelecida para o gap do algoritmo de Branch-and-Bound utilizado na resolução dos subproblemas inteiros e no algoritmo de Rubin Restrito. Este gap é calculado tomando-se a diferença relativa entre a melhor solução inteira encontrada até o momento e a melhor relaxação linear dos nós abertos. Nos exemplos estudados foi adotada uma tolerância de 0.03;

6. NumColsInt - Número de colunas adicionadas à base do subproblema anterior, no caso em que o subproblema será resolvido de forma inteira;

O Algoritmo Híbrido apresentado a seguir trabalha resolvendo uma sequencia finita de subproblemas lineares e inteiros e fazendo uso, ao longo das iterações, do Algoritmo de Rubin Restrito para melhorar os limites superiores na resolução do PCR.

Algoritmo (*Algoritmo Híbrido*)

Passo 0 *(Inicialização do algoritmo)* Inicialize $k = 0$. Encontre um vetor π dual viável para (RL) utilizando o algoritmo (*Heurística - Dual Inicial*). Selecione um conjunto A_k com n_k colunas da matriz A segundo um critério heurístico que será detalhado mais adiante. Inicialize o valor do limite superior (LimSup) para o PCR com $+\infty$.

Passo 1 *(Composição dos subproblemas)* Seja P_k um subproblema de (PCR) formado pela matriz A_k:

$$P_k = \min \left\{ {c_k}^T x_k \mid A_k x_k \geq b, x_k \in \{0,1\}^{n_k} \right\}.$$

Se $k > 0$ e n_k = NumColsInt, então adicione à matriz A_k as colunas da matriz A_{int} referentes a melhor solução inteira, $\bar{x}_{int}$, encontrada até o momento, e ainda colunas selecionadas seguindo-se o mesmo critério heurístico utilizado no Passo 0, cuja matriz resultante denotaremos A_{heur}. Atualize $A_k = [A_k \;\; A_{int} \;\; A_{heur}]$ e $n_k = n_k + n_{int} + n_{heur}$.

Passo 2 *(Obtenção da relaxação linear dos subproblemas)* Seja $\bar{x}_k$ a solução da relaxação linear de P_k:

$$\bar{x}_k = \text{argmin} \left\{ {c_k}^T x_k \mid A_k x_k \geq b, x_k \in \Re^{n_k} \right\}.$$

Seja ρ_k seu dual ótimo associado e B_k a respectiva base primal viável associada a $\bar{x}_k$. Esta base também é primal viável para (RL) pois $x = [\bar{x}_k^T \;\; 0]^T \in \{Ax \geq b, x \in \Re^n\}$.

Passo 3 *(Decisão pela resolução inteira dos subproblemas)* Seja ${c_k}^T \bar{x}_k$ o valor do limite superior de (RL), se GapInt $<$ TolInt ou se NÃO foram adicionadas as matrizes A_{int} e A_{heur} no Passo 1, vá para o Passo 6.

Passo 4 *(Adição de busca local e resolução do modelo inteiro)* Adicione ao problema P_k uma restrição do tipo (${c_k}^T x_k \leq$ LimSup) e uma restrição de busca local do tipo (n-opt). Seja então $\bar{x}_k^{int}$ a solução deste problema considerando-se uma tolerância TolBB.

Passo 5 *(Execução do Rubin Restrito para melhoria da solução inteira)* Inicialize o algoritmo de Rubin Restrito a partir das colunas da matriz A referentes a solução $\bar{x}_k^{int}$. Seja então $\hat{x}_{int}$ a nova solução inteira obtida pelo algoritmo. Se $c^T \hat{x}_{int} < {c_k}^T \bar{x}_k^{int}$, faça $\bar{x}_{int} = \hat{x}_{int}$ e LimSup $= c^T \hat{x}_{int}$, senão faça $\bar{x}_{int} = [(\bar{x}_k^{int})^T \;\; 0]^T$ e LimSup $= {c_k}^T \bar{x}_k^{int}$.

Passo 6 *(Teste de parada do Primal-Dual)* Seja ${c_k}^T \bar{x}_k$ o limite superior e $\pi^T b$ o limite inferior para o problema (RL). Se GapRel $<$ TolRel, vá para o Passo 12.

Passo 7 *(Teste de otimalidade para a relaxação linear do PCR)* Seja N o conjunto de colunas não básicas de (RL) relativas a B_k. Se $c_N - \rho_k N \geq 0$ então o vetor $x = [\bar{x}_k^T \;\; 0]^T$ será uma solução ótima de (RL) e o vetor ρ_k uma solução ótima dual viável de (RL), vá para o Passo 12.

Passo 8 *(Fase de melhoria do vetor dual viável atual π)* Encontre um escalar $\alpha_k \in \{\alpha \in \Re \mid 0 \leq \alpha \leq 1\}$ tal que o vetor $\tilde{\pi}$ formado pela combinação convexa entre π e ρ_k, $\tilde{\pi} = (1 - \alpha_k)\pi + \alpha_k \rho_k$, seja dual viável para (RL), ou seja, $\tilde{\pi} A \leq c$ e $b^T \tilde{\pi}$ seja máximo. Faça $\pi = \tilde{\pi}$.

Passo 9 *(Fase de melhoria do primal - Abordagem SPRINT)* Se $\alpha_k = 0$ alterne para utilização das variáveis duais segundo a abordagem SPRINT, ou seja, faça $\mu = \rho_k$. Senão faça $\mu = \pi$.

Passo 10 *(Critério de parada da abordagem SPRINT)* Se $k \geq 1$, $\alpha_{k-1} = 0$ e $\alpha_k = 0$,

verifique se ${c_k}^T \bar{x}_k = {c_{k-1}}^T \bar{x}_{k-1}$, em caso afirmativo vá para o Passo 12.

Passo 11 *(Seleção de colunas de menores custos reduzidos)* Faça $k = k+1$. Selecione um conjunto S_k com $n_k - m$ colunas da matriz N, onde m é o número de colunas da matriz B_{k-1}. Este conjunto de colunas é escolhido tomando-se as $n_k - m$ colunas de N com os menores valores de $c_N - \mu_{k-1}N$. A nova matriz A_k será $[B_{k-1} \;\; S_k]$. Vá para o Passo 1.

Passo 12 *(FIM)* PARE. Retorne a melhor solução inteira encontrada, $\bar{x}_{int}$, o seu respectivo valor na função objetivo, $c^T \bar{x}_{int}$, e os valores de GapRel e GapInt.

A convergência do algoritmo apresentado é clara, visto que a cada iteração k tem-se um problema primal contendo as colunas associadas à base ótima B_{k-1} do subproblema anterior, ou seja, para $k \geq 1$ tem-se a garantia que ${c_k}^T \bar{x}_k \leq {c_{k-1}}^T \bar{x}_{k-1}$. Além disso a cada iteração k é gerado um vetor dual viável, π, para (RL) com um valor na função objetivo do dual que satisfaz ${\pi_k}^T b \geq {\pi_{k-1}}^T b$.

A seguir são apresentados alguns detalhes dos passos do algoritmo descrito acima.

No passo 0, as colunas da matriz A_k são selecionadas segundo uma heurística de balanceamento da cobertura dos vôos. Uma vez que o conjunto P foi gerado como descrito no capítulo 5, temos uma lista das rotações em P em ordem crescente de custo. Selecionamos nesta fase as n_k primeiras colunas desta lista que cobrem todos os vôos ao menos um determinado número de vezes, que denotaremos $NumCoberturas$.

No passo 11, o número n_k de colunas a serem adicionadas à base do subproblema anterior pode variar entre as iterações. Adotamos, dois valores distintos: $n_k = NumColsInt$, onde $NumColsInt$ é um valor não excessivamente grande de forma a não prejudicar o desempenho do algoritmo Branch-and-Bound; e $n_k = NumColsRL$, com $NumColsRL > NumColsInt$, adotado com o intuito de melhorar apenas o limite superior da relaxação linear do PCR. Alternamos os dois valores, e submetemos os subproblemas à resolução de forma inteira apenas quando $n_k = NumColsInt$.

No passo 1, é importante ressaltar que para $k > 0$, e se $n_k =$ NumColsInt, ou seja, se o subproblema será submetido à resolução inteira, são adicionadas ao problema P_k as M colunas da matriz A_{int} referentes a melhor solução inteira encontrada na

iteração $k-1$ para que a mesma possa ser melhorada no Passo 4 através de uma restrição de busca local (n-opt) do tipo: $\sum_{j \in A_{int}} x_j \geq M - n$. Esta restrição garante que, ao menos, $M-n$ colunas da matriz A_{int} sejam mantidas na nova solução inteira a ser encontrada. Adotou-se o valor de n variável ao longo das iterações, decrescendo à medida que aumenta a dificuldade de solução do (RL) correspondente. Este critério foi estabelecido de forma empírica e buscou evitar um tempo excessivo de execução do branch-and-bound, observado nos subproblemas para os quais foi necessário um número grande de iterações do simplex para a obtenção da relaxação linear.

Ainda no passo 1, se $n_k = \text{NumColsInt}$, foram adicionadas ainda colunas selecionadas a partir da heurística de balanceamento dos vôos, só que desta vez, a lista percorrida para seleção das colunas segue uma ordem crescente de custos reduzidos. Optou-se pela adição destas colunas após várias tentativas sem sucesso do algoritmo Branch-and-Bound. Na maioria dos casos testados, observamos que foram estas colunas que possibilitaram, em vários subproblemas, a obtenção de soluções inteiras no tempo limite fornecido ao algoritmo. Em outros trabalhos, também foram utilizadas colunas adicionais para a resolução dos subproblemas de forma inteira, como em [22], onde estas foram selecionadas de forma aleatória.

No passo 3, faz-se uma verificação da diferença entre o valor na função objetivo da melhor solução inteira, $\bar{x}_{int}$, e o valor na iteração k do limite superior para o (RL). O objetivo desta verificação é evitar que o procedimento de branch-and-bound seja ativado a cada iteração do algoritmo e sim apenas quando o ponto primal viável de uma dada iteração k se "afastar" do ponto $\bar{x}_{int}$ acima de um dado limite e o algoritmo branch-and-bound se fizer necessário para que se recupere o gap estabelecido.

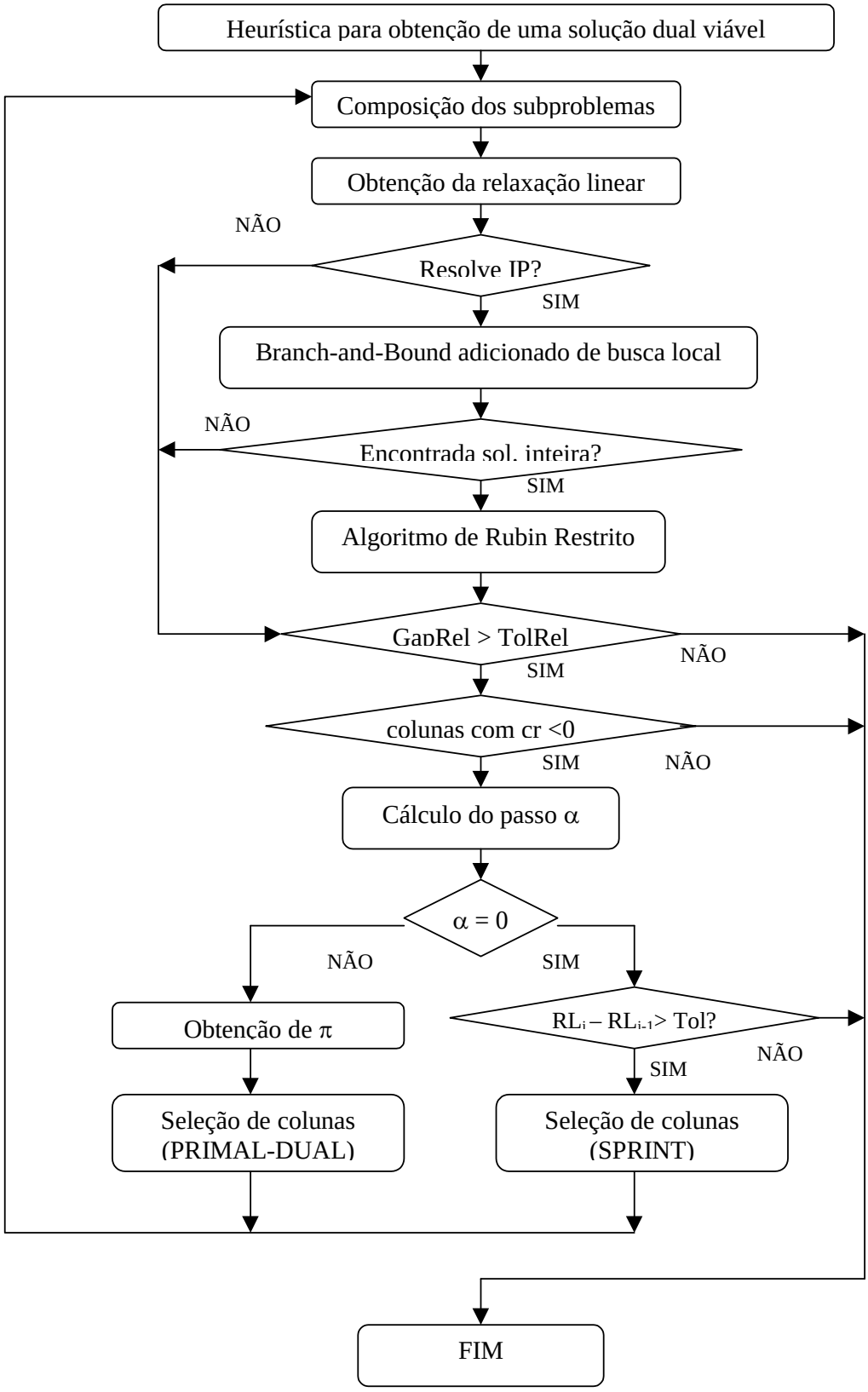
No passo 5, o algoritmo de Rubin Restrito foi utilizado para melhorar a melhor solução inteira obtida até então fazendo uso do restante das colunas em P. Observou-se que, nas primeiras iterações do Rubin Restrito consegue-se frequentemente melhorar a solução inteira, no entanto após um determinado número de iterações, não são mais obtidas melhoras. É estabelecido portanto um número máximo de iterações sem melhoria como critério de parada do algoritmo. É interessante notar que, prosseguindo com o algoritmo, quando fornecemos um outro ponto inteiro ao Rubin Restrito (sempre melhor que o anterior), este volta a obter melhoras, seguindo sempre um mesmo padrão (melhorias nas primeiras iterações), ou seja, con-

seguimos sair de um ótimo local atingido em uma solução anterior para um outro de melhor qualidade.

No passo 9, ao encontrarmos um valor de $\alpha = 0$, ficamos impossibilitados de gerar uma nova solução dual viável a partir de π, sendo assim as colunas com os menores custos reduzidos calculados a partir de π seriam as mesmas da iteração anterior, ou seja, não teríamos melhoria nem no primal nem no dual.

Para contornar este problema fizemos uma modificação no algoritmo primal-dual por subproblemas. No momento em que α assume valor nulo utilizamos o vetor dual inviável ρ_k para calcular os custos reduzidos e selecionar as colunas do novo subproblema, tal como no algoritmo SPRINT. Observamos que, ao obtermos uma nova solução primal viável e um novo vetor dual inviável associado, conseguimos voltar a atualizar o vetor dual viável, π, ou seja, conseguimos um valor de $\alpha > 0$. Quando isto acontece alternamos para a abordagem do algoritmo primal-dual por subproblemas.

A figura 6.1 ilustra o procedimento descrito:


Heurística para obtenção de uma solução dual viável
Composição dos subproblemas
Obtenção da relaxação linear
NÃO
Resolve IP?
SIM
Branch-and-Bound adicionado de busca local
NÃO
Encontrada sol. inteira?
SIM
Algoritmo de Rubin Restrito
GapRel > TolRel
NÃO
SIM
colunas com cr <0
SIM
NÃO
Cálculo do passo α
α = 0
NÃO
SIM
Obtenção de π
RLi – RLi-1> Tol?
NÃO
SIM
Seleção de colunas (PRIMAL-DUAL)
Seleção de colunas (SPRINT)
FIM

Capítulo 7

Resultados Computacionais

A malha utilizada para testar os algoritmos propostos foi cedida pela Rio-Sul Linhas Aéreas, e envolve a programação de vôos domésticos da Varig, Rio-Sul e Nordeste para o período de aproximadamente 1 mês. São 11210 trechos de vôos, conectando 48 aeroportos distintos, sendo que 6 deles são bases domiciliares de tripulantes das empresas. Para cada base tem-se os seguintes números de tripulantes disponíveis: 75, 66, 116, 16, 26 e 42, totalizando 341 tripulantes, para os quais serão designadas as rotações construídas.

Foram fornecidos os seguintes dados de entrada referentes a esta malha: horários de partida e chegada, localidades de origem e destino, e trilhos das aeronaves alocadas a cada trecho de vôo.

Primeiramente apresentaremos resultados referentes ao gerador das rotações e em seguida os referentes ao algoritmo de otimização.

7.1 Resultados para o gerador

A fim de avaliar a qualidade das rotações obtidas com o algoritmo gerador proposto, geramos, para a mesma malha, vários conjuntos de rotações. A idéia foi avaliar se o algoritmo é capaz de gerar boas soluções sem a necessidade de gerar um número excessivo de rotações.

Geramos inicialmente o maior conjunto que pudemos com os recursos computacionais disponíveis, estabelecendo como critério de parada do algoritmo o consumo de toda a memória RAM da máquina onde foi executado o processo mestre. Conseguimos gerar um conjunto de aproximadamente 11 milhões de rotações em pouco mais de 1 dia, utilizando quatro máquinas HP Proliant DL 360 G3 com 2 CPU´s

Xeon 3.06 e 4GB de memória cada.

Em seguida, a fim de avaliar o desempenho do gerador, produzimos conjuntos menores com: 100 e 500 mil; e 1, 3, 5, 7 e 9 milhões de rotações. Os conjuntos de 100 mil a 1 milhão de rotações foram gerados em uma máquina apenas, simulando processamento paralelo. Foi utilizado um Pentium III, 800MHz, com 1Gb de memória RAM.

Avaliamos a qualidade das colunas geradas através dos resultados obtidos quando submetemos cada um dos problemas resultantes ao algoritmo Primal-Dual por Subproblemas.

A tabela 7.1 apresenta os resultados obtidos:

Tabela 7.1: Resultados Computacionais para o Gerador de Rotações

NumCols (1000 colunas)	Solução Dual	Solução Primal
100	10.136.270,19	10.557.515,04
500	10.058.352,89	10.357.690,82
1.000	9.955.124,14	10.328.300,74
3.000	9.989.371,01	10.293.220,10
5.000	9.996.331,57	10.284.423,18
7.000	9.951.654,60	10.252.156,21
9.000	memória esgotada	
11.000	memória esgotada	

Obtivemos uma redução de apenas 3% nos valores das soluções primais e 2% nos valores das soluções duais, o que indica que os menores conjuntos gerados já contém rotações "atraentes". Possivelmente a diferença nos valores das soluções inteiras obtidas a partir destes conjuntos não seria significativa, o que comprovamos na fase de otimização.

7.2 Resultados para o otimizador

Os conjuntos obtidos com o gerador foram utilizados para testar o algoritmo de otimização proposto. Para todas as instância fornecemos os seguintes parâmetros, estabelecidos de forma empírica:

- Número de colunas adicionadas à base anterior dos subproblemas, no caso em que os mesmos serão submetidos apenas à resolução das suas relaxações

lineares: $NumColsRL = 40.000$;

- Número de colunas adicionadas à base anterior dos subproblemas, no caso em que os mesmos serão submetidos à resolução do problema inteiro: $NumColsInt = 15.000$;
- Parâmetro $n - opt = 10$ da busca local;
- Número máximo de iterações sem melhoria do Rubin Restrito: $MaxIntRub = 30$.

A tabela 7.2 apresenta os resultados obtidos para o conjunto de 500 mil rotações.

Observamos que para este problema a convergência do método Primal-Dual por subproblemas foi muito lenta. O número total de colunas dos subproblemas foi maior que o número de colunas do problema.

Obtivemos bons resultados nas buscas por soluções inteiras, apenas em dois subproblemas não conseguimos sucesso com o Branch-and-Bound, e também em duas chamadas ao Rubin Restrito não obtivemos melhoras.

Tomando-se o gap relativo entre a melhor solução inteira encontrada (10679682) e a melhor solução contínua primal (10357690,82), igual a 3,15%, e somando-se ao gap do Primal-Dual (2,89%), obtivemos um gap total de aproximadamente 6% entre a solução inteira obtida e o limite inferior dual. O tempo total de execução do algoritmo foi de aproximadamente 28,5 horas.

A tabela 7.3 apresenta os resultados obtidos para o conjunto de 1 milhão.

Neste caso obtivemos um gap do Primal-Dual um pouco maior que o esperado (acima de 3%). Não conseguimos voltar à fase de melhoria do dual $\alpha > 0$ após duas iterações (13 e 14) com a abordagem SPRINT, nas quais também não conseguimos baixar a solução primal.

Quanto às soluções inteiras, apenas no quarto subproblema, não obtivemos sucesso com o Branch-and-Bound, no entanto, observamos que para os subproblemas seguintes, com a adição das restrições de busca local, conseguimos encontrar soluções inteiras já nos primeiros nós.

O Rubin Restrito apresentou um melhor desempenho nas iterações iniciais, mas conseguiu melhoras, ainda que pequenas, em quase todas as iterações em que foi chamado.

O gap entre a melhor solução inteira encontrada e a melhor solução contínua primal ficou em 4,06%, e somando-se ao gap do Primal-Dual, obtivemos um gap total de 7,67%. O tempo total de execução do algoritmo foi de aproximadamente 15,3 horas.

A tabela 7.4 apresenta os resultados obtidos para o conjunto de 3 milhões.

Neste caso não tivemos problemas com a convergência do Primal-Dual, o algoritmo atingiu um gap (2,95%) inferior ao estabelecido como critério de parada (3%).

Conseguimos melhorias das soluções inteiras em todas as chamadas ao Rubin Restrito, sendo a primeira a mais significativa; e obtivemos sucesso em todas, exceto uma, chamadas ao Branch-and-Bound.

Atingimos finalmente um gap de 8,72% entre a melhor solução inteira e o limite inferior dual. O tempo total de execução do algoritmo foi de aproximadamente 17 horas.

A tabela 7.5 apresenta os resultados obtidos para o conjunto de 5 milhões.

Neste caso obtivemos sucesso em todas as chamadas ao Branch-and-Bound e em todas, exceto na iteração 12, ao Rubin Restrito. Também observamos neste caso um melhor desempenho do Rubin Restrito nas primeiras iterações.

Acionamos a busca local já no quarto subproblema, e talvez por isso tenhamos restringido demais a busca e a partir deste ponto não conseguimos melhorias maiores na qualidade das soluções inteiras.

Obtivemos para este caso um gap de 9,91% entre a melhor solução inteira e o limite inferior dual. O tempo total de execução do algoritmo foi de aproximadamente 14,1 horas.

A tabela 7.6 apresenta os resultados obtidos para o conjunto de 7 milhões.

Neste caso obtivemos sucesso em todas as chamadas ao Branch-and-Bound e apesar do Rubin Restrito não ter conseguido melhoras em 4 chamadas, obtivemos o melhor gap entre a solução inteira e a melhor solução contínua primal: 2,8%.

O gap total entre a melhor solução inteira e o limite inferior dual ficou em 5,73%. O tempo total de execução do algoritmo foi de aproximadamente 14,5 horas.

			Relaxação Linear				Branch-and-Bound				Rubin Restrito		
It	NumCols	$F_{obj}Dual$	F_{obj}	T(s)	Gap Primal-Dual	Gap (Int. Anterior)	F_{obj} Int	NumNos	T(s)	busca local	NumIt	F_{obj} Int	T(s)
1	18614	6074305,90	11521401,64	26	0,4728		11530694	75	718	não	91	11351480	960
2	27778	7922127,97	10632854,70	813	0,2549	0,0676	10730466	472	19344	não	45	10716588	382
3	46901	8603801,10	10413798,32	2867	0,1738	0,0291							
4	26175	9206506,55	10384074,18	1858	0,1134	0,0320	-	162	14399	não			
5	47015	9339768,48	10360395,53	5107	0,0985	0,0344							
6	24644	9637361,91	10359409,60	1969	0,0697	0,0345	10708906	2	984	sim	30	-	248
7	47072	9637361,91	10357761,34	5527	0,0695	0,0339							
8	23004	9680516,29	10357692,14	2518	0,0654	0,0339	10704667	6	1836	sim	37	10699652	314
9	47009	9772489,80	10357690,83	5560	0,0565	0,0330							
10	23359	9871807,50	10357690,82	2414	0,0469	0,0330	10694453	3	1345	sim	35	10693340	267
11	47028	9890827,73	10357690,82	5688	0,0451	0,0324							
12	22980	9925884,12	10357690,82	2202	0,0417	0,0324	10688481	2	1324	sim	39	10684227	322
13	46989	9925884,12	10357690,82	4529	0,0417	0,0315							
14	23172	9951814,25	10357690,82	2661	0,0392	0,0315	-	4	7201	sim			
15	47007	9986902,62	10357690,82	5940	0,0358	0,0315							
16	22805	10058352,89	10357690,82	1735	0,0289	0,0315	10679682	3	1383	sim	30	-	257

Tabela 7.2: Resultados Computacionais - Algoritmo Híbrido - 500 mil colunas

			Relaxação Linear				Branch-and-Bound				Rubin Restrito		
It	NumCols	$F_{obj}Dual$	F_{obj}	T(s)	Gap Primal-Dual	Gap (Int. Anterior)	F_{obj} Int	NumNos	T(s)	busca local	NumIt	F_{obj} Int	T(s)
1	19111	5912391,39	11608271,09	10	0,4907		11611518	14	92	não	43	11572017	535
2	25540	7302604,76	10752787,49	343	0,3209	0,0762	10841062	399	7307	não	141	10809007	1198
3	46822	8330532,76	10482310,53	1234	0,2053	0,0312							
4	23979	8853215,40	10404894,14	872	0,1491	0,0388	-	342	14401	não			
5	46944	9169918,62	10346938,23	1877	0,1137	0,0446							
6	22615	9437787,97	10338979,27	1182	0,0872	0,0455	10800623	6	1113	sim	155	10782863	1284
7	46949	9437787,97	10329599,51	2675	0,0863	0,0439							
8	22932	9557926,63	10328694,94	1516	0,0746	0,0440	10776089	1	619	sim	49	10770626	469
9	46969	9677574,05	10328369,95	2810	0,0630	0,0428							
10	21662	9767997,05	10328336,74	1469	0,0543	0,0428	10765581	1	779	sim	96	10759744	836
11	47040	9843931,83	10328301,39	3022	0,0469	0,0418							
12	21306	9912995,88	10328300,74	1491	0,0402	0,0418	10754894	1	1491	sim	30	-	227
13	47032	9955124,14	10328300,74	3051	0,0361	0,0413							
14	21029	9955124,14	10328300,74	1432	0,0361	0,0413	10750672	2	1209	sim	65	10748045	533

Tabela 7.3: Resultados Computacionais - Algoritmo Híbrido - 1 milhão de colunas

			Relaxação Linear				Branch-and-Bound				Rubin Restrito		
It	NumCols	$F_{obj}Dual$	F_{obj}	T(s)	Gap Primal-Dual	Gap (Int. Anterior)	F_{obj} Int	NumNos	T(s)	busca local	NumIt	F_{obj} Int	T(s)
1	19576	5828536,83	11786627,33	5	0,5055		11786744	1	13	não	163	11635261	3961
2	22833	7427806,13	10900890,53	183	0,3186	0,0674	10938053	162	3236	não	34	10936103	633
3	46766	8325206,46	10555062,48	1333	0,2113	0,0361							
4	21448	8892765,71	10441809,17	877	0,1484	0,0473	-	343	14399	não			
5	46825	9094067,49	10339595,33	2732	0,1205	0,0577							
6	20690	9356446,69	10325421,28	1336	0,0938	0,0591	10925910	1	564	sim	43	10923660	884
7	46858	9506446,87	10300756,41	3589	0,0771	0,0605							
8	20208	9670925,81	10298785,99	1542	0,0609	0,0607	10915602	4	2188	sim	33	10914245	663
9	46934	9747548,68	10294060,70	3635	0,0531	0,0602							
10	19762	9800366,69	10293885,84	1504	0,0479	0,0603	10906374	1	717	sim	34	10903855	690
11	46909	9839457,37	10293368,88	4513	0,0441	0,0593							
12	19703	9914048,34	10293348,29	1983	0,0368	0,0593	10897074	7	1373	sim	32	10896518	656
13	46986	9942437,19	10293232,39	4426	0,0341	0,0586							
14	19561	9989371,01	10293220,10	1758	0,0295	0,0586	10890258	1	1155	sim	36	10887983	711

Tabela 7.4: Resultados Computacionais - Algoritmo Híbrido - 3 milhões de colunas

			Relaxação Linear				Branch-and-Bound				Rubin Restrito		
It	NumCols	$F_{obj}Dual$	F_{obj}	T(s)	Gap Primal-Dual	Gap (Int. Anterior)	F_{obj} Int	NumNos	T(s)	busca local	NumIt	F_{obj} Int	T(s)
1	19852	5859773,96	11830651,45	4	0,5047		11831451	6	29	não	139	11781661	2237
2	19618	7309805,96	11090842,02	84	0,3409	0,0623	11116239	118	1840	não	98	11099699	1412
3	46558	8099875,30	10633221,29	1120	0,2382	0,0439							
4	19063	8646250,48	10532350,74	592	0,1791	0,0539	11095674	0	401	sim	154	11075208	2275
5	46770	8992822,99	10365830,29	2569	0,1325	0,0684							
6	18638	9282141,41	10349955,33	910	0,1032	0,0701	11065308	1	478	sim	31	11064648	542
7	46876	9429745,36	10298218,06	3520	0,0843	0,0744							
8	18396	9555085,97	10296402,78	1292	0,0720	0,0746	11055558	1	617	sim	31	11054888	497
9	46873	9669514,36	10286654,77	4061	0,0599	0,0747							
10	18287	9753510,84	10286497,04	1419	0,0518	0,0747	11046395	6	861	sim	74	11043066	1058
11	46971	9820063,90	10284733,00	4317	0,0452	0,0737							
12	18226	9869467,10	10284704,05	1603	0,0404	0,0737	11035386	4	1038	sim	30	-	422
13	46976	9908745,12	10284458,56	4676	0,0365	0,0730							
14	18131	9947423,92	10284452,12	1350	0,0328	0,0730	11027734	1	880	sim	42	11023648	706
15	46987	9969859,15	10284423,83	4946	0,0306	0,0719							
16	18183	9996331,57	10284423,18	1775	0,0280	0,0719	11016414	2	953	sim	46	11015699	702

Tabela 7.5: Resultados Computacionais - Algoritmo Híbrido - 5 milhões de colunas

			Relaxação Linear				Branch-and-Bound				Rubin Restrito		
It	NumCols	$F_{obj}Dual$	F_{obj}	T(s)	Gap Primal-Dual	Gap (Int. Anterior)	F_{obj} Int	NumNos	T(s)	busca local	NumIt	F_{obj} Int	T(s)
1	20206	6000936,81	11939349,29	3	0,4974		11939569	4	20	não	65	11899877	1078
2	19859	7430599,47	11133606,67	53	0,3326	0,0688	11160154	109	1145	não	72	11150148	1185
3	46583	8315269,33	10629417,47	794	0,2177	0,0490							
4	18917	8767555,34	10496873,26	543	0,1647	0,0622	10573978	337	8307	não	30	-	508
5	46733	9036497,16	10328430,73	1422	0,1251	0,0238							
6	18145	9304525,77	10314681,84	727	0,0979	0,0251	10572196	0	283	sim	31	10572101	588
7	46755	9395471,53	10268187,80	2788	0,0850	0,0296							
8	18087	9512479,92	10265656,61	923	0,0734	0,0298	10559838	1	312	sim	30	-	504
9	46853	9512744,72	10256152,23	2663	0,0725	0,0296							
10	18004	9623686,97	10255845,58	1069	0,0616	0,0296	10554530	5	524	sim	44	10552756	797
11	46913	9691524,30	10253494,01	2787	0,0548	0,0291							
12	17886	9764281,10	10253391,55	952	0,0477	0,0292	10547315	1	436	sim	30	-	532
13	46947	9769796,12	10252570,39	3304	0,0471	0,0287							
14	17901	9827358,99	10252564,11	1329	0,0415	0,0287	10544782	6	1349	sim	30	-	559
15	46972	9849717,75	10252319,12	3872	0,0393	0,0285							
16	17845	9893693,75	10252315,74	1178	0,0350	0,0285	10540456	1	548	sim	51	10539666	906
17	46887	9927339,14	10252160,23	3289	0,0317	0,0280							
18	17797	9951654,60	10252156,21	1158	0,0293	0,0280	10536303	11	1983	sim	44	10535662	818

Tabela 7.6: Resultados Computacionais - Algoritmo Híbrido - 7 milhões de colunas

Conclusões e perspectivas

A idéia quanto à contribuição deste trabalho foi tentar resolver um problema real de alocação de tripulações para companhias aéreas, enfocando a realidade brasileira; e ao mesmo tempo desenvolver um algoritmo para solução de problemas de recobrimento de grande porte, possível de ser utilizado em outras aplicações.

Após realizada uma pesquisa envolvendo os métodos de solução existentes, e considerando as características do sistema de transporte aéreo brasileiro, estabelecemos a abordagem de solução a ser seguida: tentar encontrar boas soluções viáveis em um conjunto previamente gerado de rotações. Esta abordagem permite isolar, na fase de geração das rotações, todas as questões específicas ao caso a ser tratado, como função de custo das rotações e restrições da regulamentação. A idéia foi desenvolver um gerador "sob medida" para a realidade brasileira, e trabalhar na segunda fase de uma forma mais geral, desenvolvendo um algoritmo aplicável a qualquer problema que possa ser modelado como um problema de recobrimento de conjuntos.

Considerou-se que, se o gerador fosse capaz de produzir conjuntos de boa qualidade, as soluções obtidas a partir destes seriam satisfatórias, e não haveria necessidade da geração de um conjunto excessivamente grande de rotações, ou de aplicação de técnicas como geração de colunas para incluir novas rotações na fase de otimização. No caso específico tratado, uma abordagem de geração de colunas seria praticamente inviável para os recursos computacionais disponíveis, conforme destacamos no capítulo 5.

Considerando os critérios estabelecidos, a meta-heurística GRASP possibilitou a obtenção de conjuntos de boa qualidade. Inicialmente encontramos, no entanto, uma dificuldade operacional na geração das rotações, devido à grande quantidade de rotações duplicadas geradas em diferentes iterações da heurística. Devido a esta característica, concluímos que, para o problema que se propôs resolver, um algoritmo

eficiente para geração das rotações teria que ser bastante otimizado em termos de memória de armazenamento e tempo de processamento.

A primeira implementação do gerador foi em processamento sequencial. As rotações geradas eram armazenadas, e com uma certa frequência, as mesmas eram ordenadas, utilizando um algoritmo do tipo "quick sort". A cada ordenação, as rotações duplicadas eram removidas. Com esta implementação, conseguimos gerar um máximo de aproximadamente 2 milhões de rotações em um computador Pentium III, com 1Gb de memória RAM e processador de 800MHz, em aproximadamente 2 dias. Observamos que, à medida em que o número de rotações armazenadas aumentava, o processo de geração se tornava mais lento.

Surgiu então a idéia da paralelização e da utilização da estrutura em árvore AVL para armazenamento e remoção das rotações duplicadas. A meta-heurística GRASP possibilitou um esquema de paralelização bastante simples e eficiente. Conseguimos gerar um conjunto de aproximadamente 11 milhões de rotações em pouco mais de 1 dia.

Os resultados obtidos foram avaliados a partir dos limites inferiores encontrados para os problemas resultantes de cada conjunto gerado. Observamos que, os maiores conjuntos gerados não tiveram limites inferiores proporcionalmente menores em relação aos demais. Ou seja, não foi verificado um ganho significativo em termos de redução do limite inferior, para os maiores conjuntos gerados. Isto comprova a eficiência do gerador e indica que possivelmente os menores conjuntos já contenham boas soluções.

Na fase de otimização, concebemos inicialmente dois algoritmos: um de abordagem exata e outro de abordagem heurística. A primeira idéia para o algoritmo exato foi utilizar a abordagem SPRINT, submeter alguns subproblemas à sua resolução de forma inteira, para obtenção de limites superiores; após a obtenção da relaxação linear, reduzir o tamanho do problema através da fixação de variáveis, eliminando todas as colunas com custos reduzidos inferiores ao gap obtido; e finalmente submeter o problema reduzido ao algoritmo Branch-and-Bound. A solução ótima obtida para o problema reduzido seria também uma solução ótima para o problema contendo todas as colunas geradas. Observamos, no entanto, que à medida que se aproximava da relaxação linear, as soluções inteiras ficavam mais escassas, fato observado tam-

bém em [33], conforme mencionamos no capítulo anterior. Observamos também, no que se refere à obtenção da relaxação linear do problema, a limitação do método para o tratamento de problemas degenerados. Em nenhum dos casos testados conseguimos eliminar todas as colunas com custos reduzidos negativos, atingimos sempre pontos onde subproblemas consecutivos apresentavam soluções ótimas com valores iguais, porém com diferentes bases. Não conseguimos em nenhum dos casos "sair" desses pontos nem provar otimalidade. Não obtivemos, portanto, sucesso com esta abordagem em nenhum dos casos testados.

Na tentativa seguinte utilizamos o algoritmo Primal-Dual por subproblemas [20]. Observamos que a convergência para este algoritmo é bem mais rápida. O mesmo decréscimo na função objetivo é verificado na metade de subproblemas resolvidos e, aproximadamente, na metade do tempo. No entanto, em alguns casos, atingimos pontos onde $\alpha = 0$ para gaps de dualidade ainda altos. Finalmente decidimos juntar as duas abordagens, alternando-as de acordo com os valores de α obtidos durante o processo: para $\alpha > 0$, Primal-Dual e para $\alpha = 0$, SPRINT. Obtivemos, na maioria dos casos testados, o mesmo valor para a solução primal final obtida com a abordagem SPRINT, no entanto, dispomos dessa vez de um gap de dualidade associado e consequentemente temos um limite inferior para o problema.

Para a abordagem heurística, implementamos inicialmente o algoritmo Rubin Restrito. Observamos que a qualidade das soluções finais obtidas dependendia da solução inicial fornecida. Em um dado momento as melhorias cessavam, indicando a obtenção de um ótimo local. Seria necessário alterar, de alguma outra forma, a solução corrente, para voltarmos a obter melhorias, saindo do ótimo local alcançado.

Surgiu então a idéia de combinar as duas abordagens. A heurística de Rubin Restrito seria aplicada às soluções inteiras obtidas a partir dos subproblemas gerados no algoritmo exato, e a solução final obtida com a heurística retornaria ao algoritmo exato um novo limite superior para o problema. Dessa forma conseguiríamos sair dos ótimos locais encontrados na heurística para outros pontos, também ótimos locais, mas de melhor qualidade.

Ainda assim não tivemos muito sucesso com o desempenho do Branch-and-Bound. Somente nos primeiros subproblemas, e mais notadamente no primeiro, conseguíamos obter soluções inteiras em tempo razoável. Decidimos limitar o nosso

espaço de busca através da adição de restrições de busca local, que seriam incorporadas a partir de um determinado subproblema. Esta modificação melhorou sensivelmente o desempenho do algoritmo, mas em alguns casos, ainda não encontrávamos soluções inteiras em tempo razoável. Até então utilizávamos, para composição dos subproblemas, apenas as regras de seleção de colunas por custos reduzidos, a menos do primeiro, quando ainda não tínhamos variáveis duais. Para este utilizávamos uma heurística de cobertura dos vôos, buscando possibilitar a presença de soluções inteiras viáveis garantido a cobertura de todos os vôos ao menos um determinado número de vezes. Observamos que em todos os casos eram encontradas soluções inteiras muito rapidamente para os primeiros subproblemas. Experimentamos então incorporar aos demais subproblemas colunas adicionais utilizando a mesma heurística aplicada ao primeiro. Com estas duas modificações passamos a encontrar soluções inteiras em tempo razoável na maioria dos subproblemas em todos os casos testados.

É interessante observar que a abordagem proposta proporciona a obtenção de um grande número de soluções viáveis para o problema, o que é bastante atraente do ponto de vista prático. Sabemos que às vezes, as melhores soluções encontradas em termos de custo podem ser de difícil implementação por motivos que geralmente os modelos não incorporam. Além disso, as primeiras soluções viáveis são obtidas rapidamente e já apresentam uma qualidade "razoável". Concluímos, portanto, que o algoritmo desenvolvido é capaz de encontrar boas soluções para problemas reais de construções de rotações. Além disso, constitui uma contribuição à solução de problemas de recobrimento de conjuntos, sendo capaz de encontrar soluções viáveis e limites inferiores para problemas de grande porte.

Como resultado da experiência obtida com este trabalho, concebemos algumas possíveis vertentes de trabalhos futuros. O algoritmo de Rubin Restrito poderia ser paralelizado da seguinte maneira: um processo mestre guardaria sempre a melhor solução inteira encontrada; e diversos processos escravos resolveriam os subproblemas selecionados a partir desta solução e retornariam os resultados obtidos para a atualização da mesma. Dessa forma, o ganho obtido em termos de tempo seria maior que o proporcional à divisão das tarefas de solução dos subproblemas, uma vez que cada melhoria obtida por um dos processos escravos, seria considerada por

todos os processos restantes.

A paralelização também poderia ser aproveitada na parte exata do algoritmo, para a melhoria do Primal-Dual por subproblemas, como em [22].

Uma outra possibilidade de melhoria para a heurística Rubin Restrito poderia ser explorada, no que se refere à seleção dos subproblemas. Até o presente momento, a literatura aponta a escolha aleatória como a técnica que tem apresentado melhores resultados para a seleção de subproblemas. No entanto, diversas estratégias de escolha já foram formuladas no sentido de melhorar a qualidade das soluções obtidas. Gershkoff, em [13], relatou ter experimentado estratégias que buscam por rotações cujos trechos de vôos possam ser conectados a outros presentes em rotações de alto custo. A técnica mostrou-se ocasionalmente eficiente, permitindo a saída de ótimos locais em suas experiências para problemas com vôos de curta duração, no entanto para malhas compostas por vôos mais longos, a escolha aleatória continuou apresentando os melhores resultados. Também são citadas por Gershkoff, o uso de técnicas de inteligência artificial com base no histórico de sucesso das iterações anteriores. Explorar novas heurísticas para seleção dos subproblemas, que permitam a saída de ótimos locais, poderia proporcionar melhorias significativas no desempenho do algoritmo de Rubin Restrito.

Referências Bibliográficas

[1] Anbil, R., Forrest, J., Pulleyblank, W. "Column Generation and the Airline Crew Pairing Problem". *Documenta Mathematica*, v. Extra Volume ICM III, pp. 677–686, 1998.

[2] Anbil, R., Gelman, E., Patty, B., Tanga, R. "Recent Advances in Crew Pairing Optimization at American Airlines". *Interfaces*, v. 21, pp. 62–74, 1991.

[3] Anbil, R., Tanga, R., Johnson, E. "A Global Approach to Crew Pairing Optimization". *IBM Systems Journal*, v. 31, n. 1, pp. 71–78, 1992.

[4] Andersson, E., Housos, E., Kohl, N., Wedelin, D. *Operations Research in the Airline Industry*, chapter Crew Pairing Optimization, pp. 228–258. Kluwer Academic Publishers, 1999.

[5] Ball, M., Roberts, A. "A Graph Partitioning Approach to Airline Crew Scheduling". *Transportation Science*, v. 19, n. 2, pp. 107–126, 1985.

[6] Beasley, J., Cao, B. "A Tree Search Algorithm for the Crew Scheduling Problem". *European Journal of Operational Research*, v. 94, pp. 517–526, 1996.

[7] Beasley, J., Cao, B. "A Dynamic Programming Based Algorithm for the Crew Scheduling Problem". *Computers Operations Research*, v. 25, n. 7/8, pp. 567–582, 1998.

[8] Chu, H., Gelman, E., Johnson, E. "Solving Large Scale Crew Scheduling Problems". *European Journal of Operational Research*, v. 97, pp. 260–268, 1997.

[9] Crainic, T., Rousseau, J. "The Column generation principle and the airline crew scheduling problem". *INFOR 25*, pp. 136–151, 1987.

[10] Desaulniers, G., Desrosiers, J., Dumas, Y., Marc, S., Rioux, B., Solomon, M., Soumis, F. "Crew Pairing at Air France". *European Journal of Operational Research*, v. 97, pp. 245–259, 1997.

[11] Desrosiers, J., Lasry, A., McInnis, D., Solomon, M., Soumis, F. "Air Transat Uses ALTITUDE to Manage Its Aircraft Routing, Crew Pairing and Work Assignment". *Interfaces*, v. 30, pp. 41–53, 2000.

[12] Feo, T., Bard, J. "Flight Scheduling and Maintenance Base Planning". *Management Science*, v. 35, n. 12, pp. 1415–1432, 1989.

[13] Gershkoff, I. "Optimizing Flight Crew Schedules". *Interfaces*, v. 19, n. 4, pp. 29–43, 1989.

[14] Goldbarg, M., Luna, H. *Otimização Combinatória e Programação Linear*. Editora Campos, 2000.

[15] Graves, G., McBride, R., Gershkoff, I., Anderson, D., Mahidhara, D. "Flight Crew Scheduling". *Management Science*, v. 39, n. 6, pp. 736–745, 1993.

[16] Gustafsson, T. *A Heuristic Approach to Column Generation for Airline Crew Scheduling*. PhD thesis, Chalmers University of Technology and Goteborg University, 1999.

[17] Hoffman, K., Padberg, M. "Solving Airline Crew Scheduling Problems by Branch-and-Cut". *Management Science*, v. 39, n. 6, pp. 657–682, 1993.

[18] Holloway, S. *Straight and Level: Practical Airline Economics*. Ashgate Publishing Company, 1997.

[19] Housos, E., Elmroth, T. "Automatic Optimization of Subproblems in Scheduling Airline Crews". *Interfaces*, v. 27, pp. 68–77, 1997.

[20] Jing Hu, Ellis L. Johnson. "Computational Results with a Primal-Dual Subproblem Simplex Method". *Operations Research Letters*, n. 25, pp. 149–157, 1999.

[21] Karp, R. "Reducibility Among Combinatorial Problems". In *Complexity of Computer Computations*. Plenum Press, 1972.

[22] Klabjan, D., Johnson, E., Nemhauser, G., Gelman, E., Ramaswamy, S. "Solving Large Airline Crew Scheduling Problems: Random Pairing Generation and Strong Branching". *Computational Optimization and Applications*, v. 20, n. 1, pp. 73–91, October 2001.

[23] Klabjan, D., Makri, A. "A New Pricing Scheme for Airline Crew Scheduling". *INFORMS Journal on Computing*, v. 16, pp. 56–67, 2004.

[24] Klabjan, Johnson E., D., Nemhauser, G. "A Parallel Primal-Dual Algorithm". *Operations Research Letters*, v. 27, pp. 47–55, 2000.

[25] Lavoie, S., Minoux, M., Odier, E. "A New Approach for Crew Pairing Problems by Column Generation with an Application to Air Trasnportation". *European Journal of Operational Research*, v. 35, pp. 45–58, 1988.

[26] Marsten, R., Muller, M., Killion, C. "Crew Planning at Flying Tiger: A Successful Application of Integer Programming". *Management Science*, v. 25, n. 12, pp. 1175–1183, 1979.

[27] Marsten, R., Shepardson, F. "Exact Solution of Crew Scheduling Problems Using the Set Partitioning Model: Recent Successful Applications". *Networks*, v. 11, pp. 165–177, 1981.

[28] Mingozzi, A., Boschetti, M., Ricciardelli, S., Bianco, L. "A Set Partitioning Approach to the Crew Scheduling Problem". *Operations Research*, v. 47, n. 6, pp. 873–888, 1999.

[29] Padberg, M. "A Branch-and-Cut Algorithm for the Solution of Large Scale Traveling Salesman Problems". *SIAM*, v. 33, pp. 60–100, 1991.

[30] Resende, M. "Greedy Randomized Adaptive Search Procedures". *Journal of Global Optimization*, v. 6, pp. 109–133, 1995.

[31] Rubin, J. "A Technique for the Solution of Massive Set Covering Problems with Application to Airline Crew Scheduling". *Transportation Science*, v. 7, pp. 34–48, 1973.

[32] Ryan, D., Falkner, J. "A Bus Crew Scheduling System Using a Set Partitioning Model". *Asia Pacific Journal of Operational Research*, n. 4, pp. 39–56, 1987.

[33] Vance, P., Barnhart, C., Gelman, E., Johnson, E., Krishna, A., Mahidhara, D., Nemhauser, G. *A Heuristic Branch-and-Price Approach for the Airline Crew Pairing Problem*. Technical report, Georgia Institute of Technology, 1997. LEC-97-06.

[34] Vance, P., Barnhart, C., Johnson, E., Nemhauser, G. "Airline Crew Scheduling: A New Formulation and Decomposition Algorithm". *Operations Research*, v. 45, n. 2, pp. 188–200, 1997.

[35] Wark, P., Holt, J., Ronnqvist, M., Ryan, D. "Aircrew Schedule Generation Using Repeated Matching". *European Journal of Operational Research*, v. 102, pp. 21–35, 1997.

[36] Wedelin, D. "An Algorithm for Large Scale 0-1 Integer Programming with Application to Airline Crew Scheduling". *Annals of Operations Research*, v. 57, pp. 283–301, 1995.