GERAÇÃO DAS JORNADAS DE TRABALHO DE TRIPULAÇÕES AÉREAS VIA HEURÍSTICA DE REFINAMENTOS SUCESSIVOS

Túlio Ângelo Machado Toffolo
Departamento de Computação, Universidade Federal de Ouro Preto
Campus Universitário, 35400-000 Ouro Preto, MG
tuliotoffolo@yahoo.com.br

Marcone Jamilson Freitas Souza
Departamento de Computação, Universidade Federal de Ouro Preto
Campus Universitário, 35400-000 Ouro Preto, MG
marcone@iceb.ufop.br

Roberto Carlos Vieira Pontes
Optimal Integer and Combinatorial Optimization
Caixa Postal 37846, 22640-970 Rio de Janeiro, RJ
rpontesbr@yahoo.com.br

Gustavo Peixoto Silva
Departamento de Computação, Universidade Federal de Ouro Preto
Campus Universitário, 35400-000 Ouro Preto, MG
gustavo@iceb.ufop.br

Resumo

Este trabalho tem seu enfoque em um dos problemas da programação de tripulações aéreas, denominado Geração de Jornadas de Trabalho, ou Rotações (*pairings*). Tal problema consiste em se obter um conjunto de jornadas a serem distribuídas aos tripulantes de forma que todos os vôos de uma empresa aérea sejam executados com o menor custo possível. Dada a natureza combinatorial do problema, sua solução é geralmente feita em duas etapas. Primeiramente, a partir de cada base da empresa, é gerado um número elevado de jornadas de trabalho, as quais podem variar de um a seis dias. A seguir, uma escala é obtida por meio da resolução de um problema de recobrimento de conjuntos. Este trabalho apresenta uma heurística que resolve o problema de recobrimento por meio de refinamentos sucessivos, envolvendo a geração restrita de coberturas ótimas para pequenas partes do problema. Resultados computacionais validam a metodologia proposta.

Palavras-Chaves: Programação de Tripulações Aéreas, Geração de Jornadas de Trabalho, Recobrimento de Conjuntos.

Abstract

This work deals with one of the Airline Crew Scheduling Problems, so called Crew Pairing Problem. Such problem consists in obtaining a work schedule assigning the set of trips to be assigned to a given airline company crew, in order that all flights are covered at minimum costs. Given the combinatorial nature of the problem, its resolution is usually obtained in two steps. First, from each base of the airline company, a high number of pairings or set of duties is generated, which may last from one to six days. After that, a solution is obtained solving a Set Covering Problem. This work presents a heuristic that solves the problem under continuous refinements, that are related to generation of optimal coverings for small pieces of the problem. Computational results validate the proposed methodology.

Keywords: Airline Crew Scheduling, Crew Pairing, Set Covering.

1. INTRODUÇÃO

Considere um conjunto $I = \{1, ..., m\}$ e um outro $J = \{I_j: j = 1, ..., n\}$, cujos elementos definem subconjuntos de I.. Uma *recobertura* do conjunto I é definida como sendo um subconjunto $J^* = \{ I^*_k: k= 1,...,p\}$ de J que contenha todos os elementos de I. Quando custos $\{c_j : j = 1, ..., n\}$ são associados aos elementos de J, o Problema de Recobrimento (PR) é definido como sendo o problema de encontrar uma recobertura de I de menor custo. PR é um dos problemas mais intensamente estudados em Otimização Inteira e Combinatória.

O problema é da classe NP-difícil, conforme demonstrado em [5], e é utilizado para modelar importantes aplicações em áreas tão diversas quanto, por exemplo, roteamento de veículos e localização de postos de serviço. Em particular, é também utilizado para modelar alguns problemas ligados à obtenção de escalas de tripulantes. Como é comum a problemas NP-difíceis, a solução exata de instâncias de grande porte do PR pode ser extremamente árdua de se obter. Sendo assim, nessas situações, algoritmos de solução aproximada tornam-se uma alternativa de solução muito atraente.

Nesse sentido, muitos algoritmos aproximados foram desenvolvidos nos últimos anos, sendo que alguns destes obtiveram muito sucesso. Exemplos, nesse sentido, são os algoritmos baseados em técnicas de relaxação lagrangeana, como aqueles apresentados por [2] e [3]. Tais algoritmos foram testados em instâncias reais envolvendo recobrimento de larga escala oriundas da companhia ferroviária italiana (*Ferrovie dello Stato*). De interesse, são também alguns algoritmos construtivos, como, por exemplo, a heurística gulosa introduzida por [4]. Essa heurística pode ser sofisticada através da adição de aleatoriedade parcial e da eliminação de redundâncias na solução durante a construção, como proposto em [6]. Podemos citar ainda inúmeras outras abordagens propostas na literatura, como a utilização de programação por restrições, Algoritmos Genéticos, Busca Tabu e *Simulated Annealing*.

Em relação às aplicações relacionadas aos problemas de alocação na aviação, uma contribuição muito elegante e eficaz foi proposta por Rubin em [7]. Nesse trabalho, o autor utilizou o PR para resolver pequenas coberturas referentes à geração de jornadas exaustivas de trabalho para parte dos vôos do problema, que realizadas de forma sucessiva, realizavam melhoras nas soluções intermediárias anteriores.

O presente artigo narra a utilização de um método baseado na abordagem proposta em [7] que é realizada de forma restrita, procurando-se a cobertura ótima para uma parte do problema no próprio conjunto inicial, sem que para isso seja necessária a utilização de um gerador de rotações válidas a cada iteração do algoritmo.

Este trabalho está organizado como segue. A seção seguinte discorre sobre o problema da escala de tripulantes na aviação. A seção 3 apresenta uma descrição do problema de recobrimento aplicado à programação de tripulações aéreas. A seção 4 detalha a metodologia usada para resolvê-lo, a seção 5 apresenta os resultados obtidos e a seção 6 conclui o trabalho.

2. A ESCALA DE TRIPULANTES NA AVIAÇÃO

Na aviação o modelo de recobrimento tem importância muito grande na geração da escala de tripulantes, onde um conjunto de trechos de vôos tem que ser organizado em jornadas (rotações). Essas rotações são agrupamentos de tarefas que iniciam e terminam em uma base de tripulantes e que são futuramente atribuídos a uma tripulação.

Há várias formas de resolver esse problema e talvez a mais usual seja aquela em que a geração é feita em duas fases. Primeiramente, um número muito grande de rotações viáveis obedecendo à Regulamentação dos Aeronautas é gerado; depois, um problema de recobrimento é resolvido, sendo as linhas compostas pelos trechos de vôos a serem cobertos e as colunas pelas rotações geradas. Como as rotações podem durar de um a seis dias na aviação, o conjunto gerado normalmente assume proporções muito grandes.

Para se ter uma idéia, algumas dezenas de milhões de colunas são apenas uma parte pequena da geração que seria possível se a geração de colunas fosse efetuada de forma exaustiva. Por essa razão, a geração do conjunto de rotações é feita de forma representativa, onde a presença dos vôos e das jornadas diárias seja feita de forma balanceada, e o recobrimento seja obtido de forma ótima ou de forma aproximada de boa qualidade. Observa-se que as soluções obtidas desta forma costumam superar em muito os resultados obtidos através dos planejadores das empresas aéreas que o fazem de forma manual.

- Além das restrições normais, o problema de recobrimento para obtenção do conjunto de jornadas a um custo mínimo envolve alguns aspectos singulares. São eles:
- A necessidade de distribuição da quantidade de trabalho de acordo com o efetivo de tripulantes em cada base;
- A necessidade de tornar mínimas as sobre-coberturas das linhas, que acabam representando deslocamento de tripulantes que viajam como passageiros.

Essas peculiaridades tornam o processo de obtenção de uma solução do problema ainda mais difícil, e alguns procedimentos devem ser adotados para que a solução seja de boa qualidade e obtida em um tempo razoável de execução.

3. DESCRIÇÃO DO PROBLEMA

O modelo básico do problema de recobrimento consiste na cobertura de linhas de uma matriz binária de m linhas por n colunas por um sub-conjunto de colunas de custo mínimo, onde uma linha i é coberta por uma coluna j se a posição a_{ij} é igual a 1. Considerando que c_j é o custo de cada coluna j, o problema pode ser formulado de acordo com as equações (1) – (3):

$$\textit{Minimizar} \quad \sum_{j=1}^{n} c_j x_j \tag{1}$$

$$\textit{Sujeito a:} \quad \sum_{j=1}^{n} a_{ij} x_j \geq 1 \qquad \forall\, i = 1,...,m \tag{2}$$

$$x_j \in \{0,1\} \qquad \forall\, j = 1,...,n \tag{3}$$

Como o modelo de recobrimento admite que uma linha seja coberta por mais de uma coluna, e cada linha do problema representa um vôo a ser tripulado, soluções que apresentarem coberturas de linhas acima do valor unitário representarão um vôo extra. Esses vôos constituem um custo adicional para as empresas, pois o tripulante viaja como passageiro e computando horas de efetivo trabalho para todos os efeitos legais. Dessa forma, essa sobre-cobertura deve ser um componente da função de minimização.

Normalmente os custos de uma coluna são associados à quantidade de jornadas de trabalho existente na rotação, somados aos custos de pernoites, diárias de alimentação, e tempo de inatividade embutido. Todos esses custos são expressos em unidades referenciais relativas; dessa forma, cada sobre-cobertura de uma linha terá um valor associado, multiplicando-se um coeficiente para cada unidade de cobertura adicional, que representará o custo de uma viagem de vôo extra.

Outra característica específica desse problema na aviação diz respeito às restrições de base. Independente das características da malha a ser atribuída, as empresas aéreas possuem um contingente diferente de tripulantes em cada cidade que contém uma base. Dessa forma, a

alocação deve respeitar os percentuais existentes, de forma que a quantidade de trabalho mensal seja uniforme para todos eles, até porque a remuneração é feita em função das horas trabalhadas.

Os percentuais atribuídos a cada base devem ter uma folga de dispersão admitida, para que o problema tenha um espaço de solução razoável. Essa dispersão é normalmente estabelecida em torno de 2 a 4%. Assim, como exemplo, se uma empresa aérea tiver três bases de tripulantes nas cidades do Rio de Janeiro, São Paulo e Porto Alegre, e as quantidades existentes forem nas proporções 35%, 45% e 20%, e for admitido um desvio de até 3% para cima ou para baixo na alocação, seriam estabelecidas as seguintes faixas de viabilidade:

$32\% \leq$ Total Trabalho RIO $\leq 38\%$
$42\% \leq$ Total Trabalho SAO $\leq 48\%$
$17\% \leq$ Total Trabalho POA $\leq 23\%$

4. METODOLOGIA

A proposta básica de Rubin em [7] é, partindo de um recobrimento J^* de I, considerar as possibilidades de, eventualmente, obter um recobrimento de melhor qualidade ao substituir alguns dos elementos de J^* por outros elementos diferentes de J. Esses novos elementos são escolhidos por meio da solução de um Subproblema de Recobrimento (SR) (de muito menor porte que o PR original). Isso é feito escolhendo-se, por algum critério, um subconjunto J^{**} de elementos de J^* que obedeça a certos critérios. Em particular, nenhum dos elementos de I cobertos por elementos de J^{**} é coberto por um elemento de $(J^* \backslash J^{**})$. Sendo assim, tem-se que resolver um SR relativo ao recobrimento de m^* elementos de I, que estão a descoberto em $(J^* \backslash J^{**})$. Claramente, se m^* é pequeno, SR é um problema de recobrimento de muito menor porte que o PR original. No caso específico de PR's aplicados a problemas de aviação, m pode ter uma ordem de magnitude de bilhões e, sendo assim, os n elementos de J nunca seriam explicitamente gerados. No entanto, sendo m^* suficientemente pequeno, todos os subconjuntos de J^* cobrindo apenas elementos de I a descoberto pelos elementos de $(J^* \backslash J^{**})$ é, em geral pequeno (da ordem de alguns milhares). Todos esses elementos são explícitamente enumerados apenas no momento de resolver SR. Em caso de melhora, ou seja, se o subconjunto J^{***} associado à solução de SR tiver um custo mais baixo que aquele de J^{**}, J^{**} é substituído por J^{***}. Caso contrário, nada é feito. Tanto em um caso, quanto no outro, o procedimento continua com a escolha de um novo subconjunto J^{***} a partir de J^{**} atualizado.

A variação proposta no algoritmo de Rubin é não realizar uma geração exaustiva para os elementos de I constantes do subconjunto J^{**}, através de um gerador exaustivo de jornadas viáveis, mas sim, procurar elementos no conjunto original J que cubram os m^* elementos de I, para então resolver esse pequeno problema de recobrimento.

4.1. GERAÇÃO DE UMA SOLUÇÃO INICIAL

Para iniciar o algoritmo de refinamento sucessivo é necessária a geração de uma solução base inicial. As implementações realizadas demonstraram que a qualidade dessa solução não necessita ser muito apurada, pois a fase de refinamento sucessivo rapidamente converge para bons resultados, independentemente da qualidade dessa solução.

A geração adotada é baseada na construção realizada através de um método guloso, conforme proposto por [4]. Este consiste basicamente em se adicionar a um conjunto S (denominado conjunto solução), que inicia vazio, a coluna de melhor custo-benefício entre todas existentes, até que todas as linhas estejam cobertas.

Procedimento GerarSoluçãoInicial

1. Seja *C* o conjunto de todas as colunas;
2. Seja *S* o conjunto das colunas pertencentes à solução;
3. $S \leftarrow \emptyset$;
4. $i \leftarrow 0$;
5. enquanto (todas as linhas não forem cobertas com as colunas de *S*) faça
6. Calcular o custo-benefício de cada coluna $c_i \in C$ através do procedimento *CB* (c_i);
7. Seja *k* a coluna com o menor custo-benefício $c_k \mid k \in C$ e $c_k > 0$;
8. $S \leftarrow S \cup \{ k \}$;
9. $C \leftarrow C - \{ k \}$;
10. fim-enquanto;
11. retorne *S;*

Fim GerarSolucaoInicial;

Figura 1: Algoritmo "GerarSolucaoInicial"

A Figura 1 apresenta o algoritmo responsável pela geração de uma solução inicial. O procedimento *CB*(.), aludido neste algoritmo, é responsável pelo cálculo do custo-benefício das colunas e é detalhado a seguir.

4.1.1. Cáculo do Custo-Benefício das Colunas

Basicamente, o custo-benefício de cada coluna, calculado a cada iteração do algoritmo, é representado pela razão do custo da coluna pela quantidade de linhas ainda não cobertas na solução. A essa razão, é adicionado um valor de penalização para a quantidade de linhas já presentes no conjunto de solução que a coluna estiver cobrindo, com o objetivo de reduzir o número de sobre-coberturas. As colunas que forem compostas apenas por linhas já cobertas serão desconsideradas. Para as colunas em que o número de linhas não cobertas for maior do que zero, a fórmula que calcula o custo-benefício será a dada pela equação (4):

$$Custo\ Benefício = \frac{Custo\ da\ Coluna}{Número\ de\ linhas\ não\ cobertas} + (Número\ de\ linhas\ \ já\ cobertas) \cdot k \qquad (4)$$

O coeficiente *k* é utilizado para tornar mais relevante o termo relativo à penalização de sobre-cobertura das linhas.

Para desempenho ideal do algoritmo de refinamento sucessivo (seção 4.2), é necessário que a solução base inicial seja viável com relação às restrições de quantidade de trabalho nas bases. Dessa forma, o algoritmo analisa a cada iteração da construção inicial o desvio proporcionado pela adição da coluna de melhor custo-benefício. Se esta coluna violar as quantidades percentuais, ela é submetida a um sorteio, cuja chance de aceitação é inversamente proporcional à quantidade de violação proporcionada pela sua inclusão.

Ao final da geração, quando todas as linhas estiverem cobertas pelas colunas selecionadas, caso ainda persista um desbalanceamento nas proporções de bases, uma rotina de restauração da viabilidade é executada. Esta rotina consiste em adicionar-se colunas redundantes das bases que se encontrarem abaixo do percentual mínimo de trabalho, até que o equilíbrio desejado seja alcançado. A Figura 2 mostra o pseudocódigo do algoritmo responsável pelo cálculo do valor do custo-benefício de cada coluna.

Procedimento CB
1. Seja *c* o custo referente à coluna *c*;
2. Seja *b* a base referente à coluna *c*;
3. Seja *dist_b* a distribuição de carga de trabalho desejada para a base *b*;
4. Seja *des* o desvio admitido nas porcentagens referentes às restrições de base;
5. Seja *L* o conjunto de todas as linhas cobertas pelas colunas de *S*;
6. Seja *tb* o tempo total "coberto" pela base *b*;
7. Seja *t* o tempo de vôo desta coluna;
8. Seja *ttotal* a soma do tempo de todas as colunas inseridas no conjunto *S*;
9. se (número de linhas da coluna *c* não cobertas em $L = 0$)
10. então retorne (-1);
11. $r \leftarrow (t + tb) \div (t + ttotal)$;
12. se ($r \leq dist\_b + des$)
13. então retorne (*c* / número de linhas da coluna *c* não cobertas em *L*);
14. senão
15. $a \leftarrow$ número inteiro sorteado entre 0 e 100;
16. $b \leftarrow (1 - (r - dist\_b)) \cdot 100$;
17. se ($a < b$)
18. então retorne (*c* / número de linhas da coluna *c* não cobertas em *L*);
19. senão retorne (-1);
20. fim-se;
21. fim-se;

Fim CB;

Figura 2: Algoritmo "CB", que calcula o custo-benefício de cada coluna

4.2. REFINAMENTO RESTRITO SUCESSIVO

Após a geração de uma solução inicial válida de recobrimento, o algoritmo passa a fazer refinamentos sucessivos, retirando algumas colunas da solução e procurando no conjunto total de colunas aquelas que cubram as linhas desalocadas da solução. De posse desse novo conjunto de colunas, que inclui aquelas que foram retiradas da solução, um subproblema de recobrimento desse subconjunto de colunas é resolvido por uma metodologia exata, baseada em programação linear inteira. Se a solução desse subproblema for de custo menor que o do conjunto de colunas desalocadas, o novo conjunto é substituído no conjunto solução. O processo se repete até que um número de iterações ou um transcurso de tempo seja atingido.

As colunas ótimas resultantes da resolução do subproblema de recobrimento, ao serem inseridas no conjunto solução, devem satisfazer ao balanceamento exigido pelas restrições de base. Para cada base constante do problema, deverá ser escrito um conjunto de restrições que garantam a distribuição equilibrada das bases em relação à carga de trabalho.

É interessante observar que entre as linhas trazidas para o subproblema, provavelmente existam algumas que já possuam cobertura no conjunto de colunas que permanecem na solução base. Essas linhas, obviamente, não necessitam ser cobertas na solução do subproblema analisado, e a eliminação delas no conjunto a substituir acaba reduzindo gradativamente o número de sobre-coberturas na solução corrente.

O conjunto de colunas retirado da solução inicial ou solução corrente deve ser em número reduzido, de forma que possa ser resolvido de forma exata sem grandes dificuldades. O modelo de programação linear utilizado para resolver o Subproblema de Recobrimento (SR) é dado a seguir, por (5) – (10).

$$\textit{Minimizar} \quad \sum_{j=1}^{n} c_j x_j + \sum_{i=1}^{m} s_i k \tag{5}$$

$$\textit{Sujeito a:} \quad \sum_{j=1}^{n} \left(a_{ij} x_j \right) - s_i = 1 \qquad \forall i = 1,...,m \tag{6}$$

$$\sum_{j=1}^{n} \left((1 - a_{ij}) \cdot x_j \right) - s_i = 0 \qquad \forall i = 1,...,m \tag{7}$$

$$\sum_{j=1}^{n} \left(h_{lj} x_j t_j \right) + H_l \geq \left(p_l - desvio \right) \sum_{l'=1}^{q} H_{l'} \qquad \forall l = 1,...,q \tag{8}$$

$$\sum_{j=1}^{n} \left(h_{lj} x_j t_j \right) + H_l \leq \left(p_l + desvio \right) \sum_{l'=1}^{q} H_{l'} \qquad \forall l = 1,...,q \tag{9}$$

$$x_j \in \{0,1\} \qquad \forall j = 1,...,n \tag{10}$$

O modelo consiste na cobertura de linhas de uma matriz binária de m linhas por n colunas. Cada coluna tem custo c_j. Se uma linha i for coberta por uma coluna j e não for coberta pelo conjunto solução corrente, então a_{ij} será igual a 1 e, caso a linha seja coberta pelo conjunto solução corrente, então a_{ij} será igual a 0. A restrição (6) do modelo acima garante que todas as linhas ainda não cobertas na solução corrente sejam cobertas por este subconjunto. A variável de relaxação s_i permite que haja sobre-coberturas, entretanto penaliza na função de avaliação, com um multiplicador k, cada uma das sobre-coberturas. A restrição (7) procura evitar que linhas já cobertas pela solução sejam sobre-cobertas, mas não impede que isto ocorra. Novamente, é utilizada a variável de relaxação s_i, com cada sobre-cobertura sendo penalizada com peso k na função objetivo. As restrições (8) e (9) tratam do balanceamento das q bases do problema. Sabendo-se que o modelo é gerado a cada iteração do algoritmo de forma explícita, e que é possível reconhecer a qual base cada coluna pertence, assume-se que h_{lj} será igual a 1 quando a coluna j pertencer à base l e 0 caso contrário, que H_l é a soma do tempo total trabalhado por colunas da base l que pertencem à solução corrente. Considera-se também que t_j é o tempo de trabalho da coluna j e que p_l é a porcentagem de trabalho desejada para a base l em relação ao total de trabalho da empresa aérea. Admite-se, ainda, um desvio em relação às restrições de cada base, representado pela constante *desvio*. Portanto, as restrições (8) e (9) garantem que a solução sempre será viável em relação às bases.

4.2.1. Limite Superior para os Subproblemas de Recobrimento

A cada iteração, o algoritmo retira algumas colunas (X) da solução e procura no conjunto total C de colunas aquelas que cobrem as linhas desalocadas. A partir da resolução do subproblema de recobrimento aplicado às colunas que cobrem as linhas desalocadas, têm-se as colunas "substitutas" daquelas que foram retiradas. Se o conjunto X de colunas retiradas da solução for submetido à função de avaliação seguinte, dada pela equação (11), um limite superior para o subproblema de recobrimento é obtido.

$$f(X) = t \cdot k + \sum_{j=1}^{n} \left(c_j x_j \right) \tag{11}$$

Na equação (11), c_j é o custo de cada coluna, t é o total de sobre-coberturas no conjunto e k é o peso na função de avaliação referente às sobre-coberturas.

4.3. ALGORITMO PROPOSTO

O algoritmo proposto para a resolução do Problema do Recobrimento é apresentado na Figura 3.

Algoritmo BuscaRestrita
1. Seja *C* o conjunto que contém todas as colunas do PR;
2. Sejam *desvio*, *minlinhas*, *maxlinhas* e *pct* os parâmetros considerados para a execução deste algoritmo;
3. $S \leftarrow$ *GerarSolucaoInicial* ( );
4. enquanto (critério de parada não satisfeito) faça
5. *nlinhas* ← número aleatório entre *minlinhas* e *maxlinhas*;
6. Seja *S´* um conjunto de colunas a serem retiradas da solução *S* corrente;
7. enquanto (número de linhas cobertas por $S' \leq$ *nlinhas*) faça
8. Seja *c* uma coluna qualquer do conjunto *S*;
9. $S' \leftarrow S' \cup \{c\}$;
10. $S \leftarrow S - \{c\}$;
11. fim-enquanto;
12. *S''* ← todas as colunas de *C* em que *pct %* de suas linhas não fazem parte de *S;*
13. $S'' \leftarrow S' \cup S''$;
14. S' ← *RefinamentoRestritoSucessivo* (*S', S'', desvio*); {vide seção 4.2}
15. $S \leftarrow S \cup S'$;
16. fim-enquanto;
17. retorne *S;*

Fim BuscaRestrita;

Figura 3 – Algoritmo BuscaRestrita

Neste algoritmo, cada solução *S* produzida é avaliada com base na função dada pela equação (12).

$$FO(S) = t \cdot k + \sum_{j \in S} c_j \qquad (12)$$

em que: c_j é o custo de cada coluna $j \in S$, *t* é o total de sobre-coberturas no conjunto e *k* é o coeficiente utilizado para tornar mais relevante o termo relativo à penalização de sobre-cobertura das linhas.

O algoritmo possui basicamente quatro parâmetros:

- *desvio*: valor do desvio admitido em relação ao balanceamento das bases;
- *minlinhas*: número mínimo de linhas que devem ser desalocadas da solução, a cada iteração, para gerar o SR;
- *maxlinhas*: número máximo de linhas que devem ser desalocadas da solução, a cada iteração, para gerar o SR
- *pct*: porcentagem mínima de linhas não cobertas pela solução que uma coluna deve ter para ser utilizada na fase de Refinamento Restrito Sucessivo.

A cada iteração, conforme observado na linha 5 da Figura 3, um número aleatório entre *minlinhas* e *maxlinhas* é gerado. Esse número representa a quantidade de linhas que serão desalocadas da solução. Observe que se o PR contiver um número excessivamente

grande de colunas, os valores de *minlinhas* e *maxlinhas* devem ser suficientemente pequenos de forma que o conjunto de colunas que fará parte do SR seja de tamanho reduzido, para possibilitar sua resolução pelo método de programação matemática. Numa situação limite, em que os valores de *minlinhas* e *maxlinhas* são iguais ao total de linhas do problema, cada SR gerado será, na verdade, o próprio PR.

O parâmetro *pct* permite controlar, de certa forma, quantas colunas serão utilizadas para se resolver um SR gerado numa dada iteração. Por exemplo, em um problema com um número muito grande de colunas, pode ser interessante fazer *pct* = 100%. Neste caso, as colunas compostas apenas por linhas não pertencentes à solução corrente serão selecionadas para o subproblema, reduzindo assim a dificuldade de se resolver o SR. Em um problema com poucas colunas, por outro lado, se o valor de *pct* for de 100% o algoritmo pode não se comportar de forma satisfatória, pois um número muito pequeno de colunas será utilizado para gerar o SR. Por exemplo, na Figura 4, supõe-se que apenas 3 das 6 linhas da coluna *i* (no caso, linhas 2, 6 e 17) não estão, em uma dada iteração, sendo cobertas pela solução corrente. Neste caso, 50% das linhas da coluna *i* não são cobertas pela solução corrente. Assim, a coluna *i* fará parte do subproblema gerado na dada iteração somente se o valor de *pct* for menor ou igual a 50%.

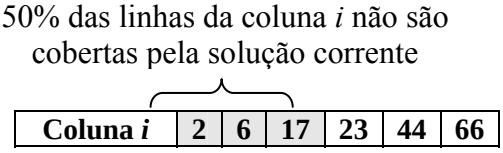

Figura 4 – Porcentagem de linhas não cobertas de uma coluna

5. TESTES E RESULTADOS

Os algoritmos foram desenvolvidos na linguagem C++ com a IDE e o compilador do Borland C++ Builder 6.0, utilizando o pacote de programação linear LINGO 7.0, da Lindo Systems. Os testes foram realizados em um microcomputador Pentium IV HT 3,0 GHz, com 512 MB de memória RAM, sob sistema operacional Windows XP SP2.

A seção 5.1 explica como as instâncias-teste para o problema foram geradas, a seção 5.2 mostra os parâmetros utilizados, enquanto a seção 5.3 mostra os resultados computacionais do algoritmo proposto.

5.1. INSTÂNCIAS-TESTE

As instâncias-teste são fictícias, mas foram geradas de forma a simular uma situação real. Elas consistem em um conjunto de jornadas de trabalho, a cada qual associado um custo, tempo de vôo, base e uma lista de tarefas, que por sua vez, é composta por uma viagem com data, hora, tempo de vôo e custo definidos. As instâncias encontram-se disponíveis na página http://www.decom.ufop.br/prof/marcone/projects/scp/instancias.html.

Para todas as instâncias considerou-se 3 bases, com as seguintes distribuições desejadas para cada uma: 40% do tempo total de vôo deve partir da base 1, 30% da base 2 e 30% da base 3.

A Tabela 1 mostra as características dessas instâncias.

Tabela 1: Características das instâncias-teste.

Característica	Instância				
	Inst_1	Inst_2	Inst_3	Inst_4	Inst_5

Número de Linhas (Tarefas)	100	150	150	150	150
Número de Colunas (Jornadas)	74.352	33.293	66.540	1.355.067	3.304.438
Duração Máxima Permitida de uma Jornada	4 dias	4 dias	4 dias	4 dias	4 dias

5.2. PARÂMETROS E PESOS ADOTADOS

A Tabela 2 mostra os valores dos parâmetros e pesos adotados para cada instância.

Tabela 2: Parâmetros e pesos adotados para cada instância.

Parâmetro	**Instância**				
	Inst_1	**Inst_2**	**Inst_3**	**Inst_4**	**Inst_5**
Peso para cada sobrecobertura na FO	150	150	150	150	150
Desvio admitido em relação às distribuições das bases	3%	3%	3%	3%	3%
Número mínimo de linhas (tarefas) em cada SR (parâmetro ***minlinhas***)	30	30	30	30	30
Número máximo de linhas (tarefas) em cada SR (parâmetro ***maxlinhas***)	65	100	90	70	60
Proporção mínima de linhas (tarefas) não cobertas pela solução que uma coluna deve conter (parâmetro ***pct***)	70%	70%	75%	80%	85%

Para a realização dos testes, um tempo máximo de 90 segundos foi estipulado para que cada subproblema de recobrimento (SR) fosse resolvido. Este procedimento foi adotado para evitar que muito tempo de processamento fosse dedicado a uma única iteração. Caso este tempo seja atingido antes de a solução ótima para o SR ser encontrada, é considerada a melhor solução obtida antes de o tempo limite ser atingido.

5.3. RESULTADOS COMPUTACIONAIS

A Tabela 3 mostra o valor da melhor solução obtida ($\mathbf{FO^H}$) para cada instância pela metodologia proposta, bem como o desvio médio (*gap*) em relação à melhor solução obtida em 10 execuções do algoritmo, tendo como critério de parada 30 minutos de tempo de processamento. Nesta tabela, $\mathbf{FO^L}$ é o valor da melhor solução encontrada pelo otimizador LINGO, versão 7.0.

Tabela 3: Desempenho do algoritmo

Item	**Instância**				
	Inst_1	**Inst_2**	**Inst_3**	**Inst_4**	**Inst_5**
$\mathbf{FO^H}$	44.808	73.645	70.856	67.890	67.619
Desvio (*gap*)	0,002%	0,3%	0,22%	0,44%	1,52%
$\mathbf{FO^L}$	44.808 (1)	77.639 (2)	77.404 (2) 71.014 (3)	- (4)	- (4)

(1) Solução ótima, encontrada em 3 minutos de processamento.
(2) Melhor solução encontrada pelo Lingo em 60 minutos de processamento.
(3) Melhor solução encontrada pelo Lingo em 1440 minutos (24 horas) de processamento
(4) O Lingo foi incapaz de resolver, ocorrendo estouro de memória.

Pela Tabela 3, verifica-se que o método proposto é capaz de encontrar a solução ótima para a instância **Inst_1** com desvio de 0,002%. Para as instâncias **Inst_2** e **Inst_3**, o método gera, em 30 minutos, soluções com qualidade muito superior àquelas geradas pelo Lingo em 60 minutos (respectivamente, 5% e 8% de melhora). Para a instância **Inst_3**, quando o tempo de processamento do Lingo é de 22 horas, a qualidade da solução produzida pelo método continua melhor, no caso, em 0,2%. Quanto às instâncias **Inst_4** e **Inst_5**, o otimizador Lingo não foi capaz sequer de resolvê-las. Destaca-se também a pequena variabilidade das soluções finais produzidas pelo método, com *gap* máximo de 1,52%.

A Tabela 4 mostra as características da melhor solução obtida em cada instância.

Tabela 4: Características das melhores soluções

Características	**Instâncias**				
	Inst_1	**Inst_2**	**Inst_3**	**Inst_4**	**Inst_5**
Distribuição do tempo total de vôo entre as bases	B1=40,00% B2=32,99% B3=27,01%	B1=42,57% B2=30,20% B3=27,23%	B1=42,86% B2=29,87% B3=27,27%	B1=42,52% B2=30,25% B3=27,23%	B1=42,04% B2=30,49% B3=27,47%
Número de sobre-coberturas	14	42	33	27	28
Número de jornadas	24	35	35	35	33
Soma dos custos das jornadas	42.708	67.345	65.906	63.840	63.419

Na Tabela 4, verifica-se que as soluções retornadas pelo método são todas viáveis em relação às restrições de base e que as sobre-coberturas variam de 14% (**Inst_1**) a 28% (**Inst_2**) do número de tarefas.

A Figura 5 ilustra a evolução típica do melhor valor da função de avaliação (FO) nos 30 minutos de processamento em uma execução do método aplicado à instância **Inst_4**. Observa-se nesta figura que o método rapidamente gera soluções de boa qualidade.

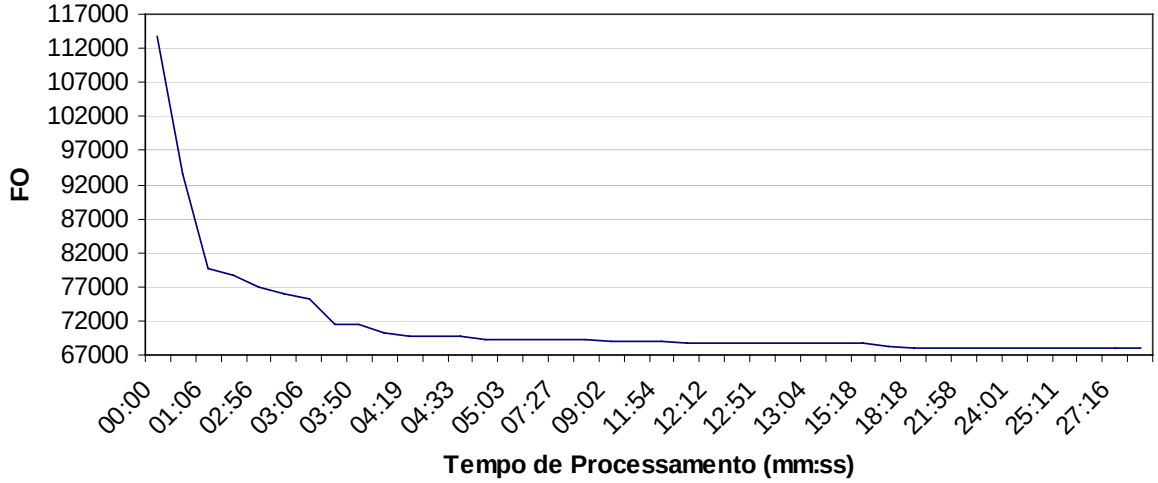

Figura 5 – Evolução típica do melhor valor da função de avaliação

6. CONCLUSÕES

Este trabalho propõe uma metodologia heurística para resolver problemas de recobrimento de conjuntos oriundos da Programação de Tripulações Aéreas. Esta metodologia

consiste em fazer refinamentos sucessivos envolvendo a geração restrita de coberturas ótimas para pequenas partes do problema. Os parâmetros do método permitem que ele seja ajustado para resolver problemas de diferentes dimensões.

Esta metodologia mostrou-se adequada para esta classe de problemas, uma vez que foi capaz de gerar soluções de boa qualidade rapidamente, com baixa variabilidade nas soluções finais.

Agradecimentos
Os autores agradecem ao CNPq e à UFOP pelo apoio ao desenvolvimento deste trabalho, bem como à Borland Latin America pela concessão de uma licença de uso do software C++ Builder 6.0.

7. REFERÊNCIAS BIBLIOGRÁFICAS

[1] CABRAL, L. A. F.; PONTES, R. C. V.; SOUZA, M. J. F.; MACULAN, N. (2000) An heuristic approach for large scale crew scheduling problems at Rio Sul Airlines. In: *Proceedings of the 40th International Symposium of the AGIFORS*, p. 1-8, Istambul.

[2] CAPRARA, A.; FISCHETTI, M.; TOTH, P. (1996) A heuristic method for the Set Covering Problem. *European Journal of Operational Research*, 94:392-404.

[3] CERIA, S.; NOBILI, P.; SASSANO, A. (1995) A Lagrangian-based Heuristic for Large-scale Set Covering Problems. *Mathematical Programming*.

[4] CHVÁTAL, V. (1979) A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4:233-235.

[5] GAREY, M. R. & JOHNSON, D. S. (1979) Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, San Francisco.

[6] MARCHIORI, E. & STEENBEEK, A. (1998) An iterated heuristic algorithm for the set covering problem. In Kurt Mehlhorn, editor, *Proceedings of the Workshop on Algorithm Engineering*, p. 155-166.

[7] RUBIN, J. (1973) A technique for the solution of massive set covering problems, with application to airline crew scheduling. *Transportation Science*, 7:34- 48.

[8] WEDELIN, D. (1995) An Algorithm for Large Scale Integer Programming with Application to Airline Crew Scheduling. *Annals of Operations Research*.