DEPARTAMENTO DE COMPUTAÇÃO

D E C O M

U F O P

UNIVERSIDADE FEDERAL DE OURO PRETO

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

Heurísticas para o Problema da Programação de Tripulações Aéreas

Túlio Ângelo Machado Toffolo

Orientador: Prof. Dr. Marcone Jamilson Freitas Souza
Co-orientador: Prof. Dr. Gustavo Peixoto Silva

Relatório Técnico DECOM xx/2006

Ouro Preto - MG
Outubro de 2006

FOLHA DE APROVAÇÃO DA BANCA EXAMINADORA

Heurísticas para o Problema da Programação de Tripulações Aéreas

Túlio Ângelo Machado Toffolo

Monografia apresentada ao Departamento de Computação do Instituto de Ciências Exatas e Biológicas da Universidade Federal de Ouro Preto como requisito parcial da disciplina CIC391 – Monografia, do curso de Bacharelado em Ciência da Computação, e aprovada pela Banca Examinadora abaixo assinada:

_____

Prof. Dr. Marcone Jamilson Freitas Souza
Orientador
Departamento de Computação – UFOP

_____

Prof. Dr. Gustavo Peixoto Silva
Co-orientador
Departamento de Computação – UFOP

_____

Prof. Ms. Elton José da Silva
Examinador
Departamento de Computação – UFOP

Ouro Preto, 6 de Outubro de 2006

Sumário

Lista de figuras

Lista de tabelas

Resumo

Este trabalho aborda o Problema da Programação de Tripulações Aéreas. Tal problema consiste em se obter uma escala atribuindo diferentes jornadas de trabalho aos tripulantes de forma que todas as viagens de uma empresa aérea sejam executadas com o menor custo possível. Esta escala deve satisfazer a uma série de restrições trabalhistas e operacionais. Dado o elevado número de escalas diferentes a analisar e a natureza combinatorial do problema, sua resolução é geralmente feita em duas etapas. Primeiramente, a partir de cada base da empresa, é gerado um número elevado de jornadas de trabalho ou rotações (pairings), as quais podem variar de um a cinco dias, em geral. Estas jornadas consistem em um conjunto de viagens, sendo que cada viagem possui data e local de início e fim pré-determinados. A seguir, uma escala é obtida por meio da resolução de um problema de recobrimento de conjuntos (set covering problem), considerando as jornadas geradas pela primeira etapa. Neste trabalho é apresentada uma metodologia baseada em um algoritmo híbrido, que faz uso de técnicas heurísticas e de programação linear inteira para resolver o problema de recobrimento de conjuntos. Uma solução inicial para o problema é gerada por uma adaptação da heurística proposta por Chvátal. A partir da solução inicial, o algoritmo realiza refinamentos sucessivos, que envolvem a geração restrita de subcoberturas ótimas para pequenas partes do problema. Estas subcoberturas ótimas são obtidas através de um modelo de programação linear inteira, e devem ter tamanho reduzido de forma a serem resolvidas rapidamente. Os parâmetros dos métodos permitem que o algoritmo proposto seja ajustado para resolver problemas de diferentes dimensões. Resultados computacionais satisfatórios validam o trabalho, mostrando que a metodologia proposta é adequada para esta classe de problemas, uma vez que foi capaz de gerar soluções de boa qualidade rapidamente, com baixa variabilidade nas soluções finais.

Palavras-chave

Heurísticas, Problema da Programação de Tripulações Aéreas, Problema do Recobrimento, Otimização Combinatória.

Abstract

This work deals with the Airline Crew Scheduling Problem. Such problem consists in obtaining a work schedule assigning the set of duties of a given airline company to a set of drivers with minimal cost. This schedule must satisfy labor rules and operational restrictions. Given the high number of different schedules to analyze and the combinatorial nature of the problem, its resolution is usually made in two steps. First, from each base of the airline company is generated a high number of pairings or set of duties, which usually can last from one to five days. The pairings are considered groups of trips. Each trip has date and time of departure and arrival already determined. Following, an schedule is obtained through the solution of a set covering problem, using the pairings generated during the first step. This work presents an hybrid algorithm that uses both heuristics and linear programming techniques to solve the set covering problem. An initial solution for the problem is obtained thought an adaptation of Chvátal's greedy heuristic. From the initial solution, the algorithm makes continuous refinements, which involves the restricted generation of optimal coverings for small pieces of the problem. Such optimal coverings are obtained thought a linear programming model. The parameters of the method allows it to be used to solve problems of different size. The results are satisfactory, showing that the methodology is adequate to solve this kind of problems, since it was able to generate good solutions spending little time, with low variability in the final solutions.

Keywords

Heuristics, Airline Crew Scheduling Problem, Set Covering Problem, Combinatorial Optimization.

1. 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 esta definir a própria sobrevivência das empresas. Neste contexto, técnicas de pesquisa operacional vêm sendo usadas por companhias aéreas com uma intensidade sem igual em outros setores da atividade econômica. São inúmeros os exemplos bem sucedidos de adoção de soluções envolvendo a pesquisa operacional em companhias aéreas.

Devido à sua complexidade, o planejamento da alocação de recursos em companhias aéreas é usualmente dividido em quatro etapas: (1) Construção do quadro de horários; (2) Alocação de aeronaves; (3) Construção das rotações e (4) Alocação das tripulações.

Na fase de construção do quadro de horários, 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 à demanda e as restrições referentes à quantidade de aeronaves disponíveis na empresa e a disponibilidade de operação dos aeroportos.

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 (Holloway, 1997). 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 (Anbil *et al.*, 1992). 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 (Graves *et al.*, 1993). Outras grandes companhias, como a Nortwest 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 (Holloway, 1997).

Na fase de alocação de aeronaves, é realizado o processo de alocação das aeronaves 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 qual aeronave. Normalmente este problema é 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, problema referenciado na literatura como *fleet assignment problem* e, em seguida, são determinadas as seqüências de vôos a ser realizada por cada aeronave de uma dada frota, problema conhecido na literatura por *aircraft routing problem*.

Na terceira etapa do planejamento, é realizada a tarefa de construção de rotações. Uma rotação ou *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. Nessa figura, a rotação A, por exemplo, consiste na seguinte jornada de trabalho: Uma aeronave sai de São Paulo com destino ao Rio de Janeiro pelo vôo 1. A seguir, depois de um certo tempo de solo, essa aeronave ou possivelmente outra, sai do Rio de Janeiro com destino à Porto Alegre pelo vôo 5. Após um certo período de inatividade em Porto Alegre, a aeronave que pousou em Porto Alegre ou possivelmente uma outra aeronave, sai dessa cidade com destino à São Paulo pelo vôo 8. Esta seqüência de vôos saindo e retornando à cidade base São Paulo constitui uma rotação ou *pairing*.

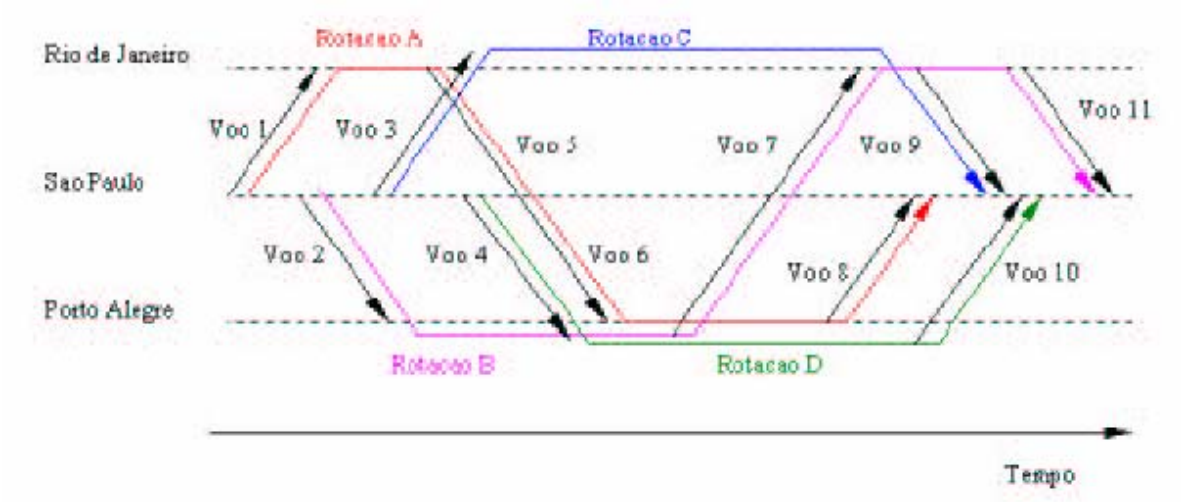


Figura 1.1 – Jornadas de Trabalho ou Rotações (*Pairings*)

Observa-se que as rotações ainda não estão alocadas às tripulações, sendo apenas relacionadas a uma determinada base. Uma rotação pode envolver trechos de vôos de diferentes trilhos, ou seja, realizadas 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 trabalhos contínuos, máximos 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 de trabalho permitido sem que haja folga para uma tripulação. Uma rotação pode incluir também os denominados *deadheads*, isto é, 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. Nesta etapa, o trabalho consiste em estabelecer um conjunto de rotações que cubra todos os trechos de vôos operados pela empresa, de forma a minimizar uma função de custo adotada. O custo envolvido na operação das rotações compreende diversos fatores, podendo ser expresso em termos de: utilização dos tripulantes, número de pernoites fora da base (despesas com hotel, alimentação, etc), e outros.

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.

Após a construção das rotações, segue-se a alocação das mesmas aos tripulantes, sendo este o problema objeto de investigação neste projeto, o qual é conhecido como problema de programação de tripulações aéreas (*airline crew scheduling problem*). O objetivo básico desta fase é designar a cada tripulante um conjunto de tarefas, que podem ser: rotações, folgas, sobreaviso (período no qual o tripulante deve estar à disposição da companhia para a realização de algum vôo não planejado), reserva (período no qual o tripulante deve permanecer no aeroporto para a eventual realização de um vôo), treinamentos, férias, etc, de forma que sejam atendidas todas as restrições da regulamentação. A alocação pode ainda incluir uma minimização do número de tripulações necessárias, devendo, no entanto, preservar uma certa folga devido a possíveis imprevistos durante a execução da escala. 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 é base. Dessa forma, a alocação deve respeitar os percentuais existentes, de forma que a quantidade de trabalho seja uniforme para todos eles, até porque a remuneração em geral é feita em função das horas efetivamente trabalhadas.

O problema de programação de tripulações é usualmente tratado por um modelo de recobrimento ou particionamento de conjuntos. (Barnhart *et al*, 1998, Haase e Friberg, 1999; Mingozzi *et al*., 1999; Wedelin, 1995; Ceria *et al*, 1995; Smith & Wren, 1988). A variedade de trabalhos em geral deriva das diferentes possibilidades de se encontrar uma solução inteira a partir da solução do Problema de Programação Linear. Entre as técnicas de programação matemática mais exploradas estão a de *branch-and-bound*, *branch-and-price* e a relaxação lagrangeana.

Da fase anterior do problema, a construção de rotações, é gerado um número muito grande de rotações viáveis obedecendo à Regulamentação dos Aeronautas. Assim, no

modelo de recobrimento ou particionamento, as linhas são 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 um problema de médio porte de uma empresa aérea média, a geração de um conjunto de rotações semanais envolve aproximadamente 1.500 linhas por 30 milhões de colunas. Como esse problema é NP-difícil, o uso exclusivo de técnicas exatas de solução é limitado a instâncias de pequenas dimensões. Para instâncias maiores, o procedimento mais comum é utilizar técnicas heurísticas de solução. Com esta metodologia podemos citar, dentre outros: Caprara *et al*. (1996), Chvátal (1979), Rubin (1973) e Marchiori & Steenbeek (1998).

No presente projeto pretende-se tratar o problema através de técnicas híbridas, combinando heurísticas com métodos de programação matemática. Esta é uma estratégia que combina o poderio dos métodos de programação matemática (ditos exatos) com a flexibilidade das heurísticas. Como os métodos exatos têm dificuldade para resolver o problema de dimensões mais elevadas, a idéia então é chamá-lo apenas para resolver uma parte "pequena" do problema, encarregando-se a heurística de escolher que parte do problema será resolvida a cada iteração.

Uma primeira idéia seria partir de um conjunto inicial grande de colunas e, a partir dele, por uma heurística construtiva parcialmente gulosa, tal como na heurística de Chvátal, formar uma solução inicial com R colunas. Escolhido um conjunto pequeno de *m* linhas (trechos de vôos), seriam retiradas todas as colunas da solução inicial que cobrissem essas linhas, digamos S colunas. A seguir, seriam procuradas no conjunto original de colunas todas as colunas que cobrissem pelo menos uma das *m* linhas referenciadas. Para essas colunas que cobrissem as *m* linhas seria acionado um procedimento de programação matemática para resolver o modelo de recobrimento, encontrando o melhor conjunto de colunas, digamos T colunas. Observa-se que a resolução de forma exata desse subproblema é factível se o conjunto T for reduzido. A seguir, esse conjunto T de colunas seria, então, inserido na solução corrente em lugar das S colunas retiradas anteriormente. Caso a solução representasse uma melhora no valor da função de avaliação, essa passaria a ser a nova solução corrente.

2. Revisão Bibliográfica

2.1. Problema do Recobrimento de Conjuntos

Considere-se um conjunto $I = \{1, ..., m\}$ e um outro conjunto, $J = \{I_j: j = 1, ..., n\}$, 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 Garey *et al.* (1979), 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 Caprara *et al*. (1996) e Ceria *et al*. (1995). 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 Chvátal (1979). Essa heurística pode ser sofisticada através da adição de aleatoriedade parcial e da eliminação de redundâncias na solução durante a construção, como proposto por Marchiori *et al*. (1998). 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 (1973). 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.

2.2. Heurística de Rubin

A proposta básica de Rubin (1973) 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 $(J^*|J^{**})$. Sendo assim, tem-se que resolver um SR relativo ao recobrimento de m^* elementos de I, que estão a descoberto em $(J^*|J^{**})$. 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 $(J^*|J^{**})$ é, em geral pequeno (da ordem de alguns milhares). Todos esses elementos são explicitamente 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.

3. Metodologia

3.1. Descrição do problema abordado

3.1.1. A Escala de Tripulações 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 geralmente podem durar de um a cinco 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.

3.1.2. O Problema de Recobrimento de Conjuntos

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):

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

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

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

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% ≤ Total Trabalho RIO ≤ 38%
42% ≤ Total Trabalho SAO ≤ 48%
17% ≤ Total Trabalho POA ≤ 23%

3.2. Representação

Conforme visto anteriormente, a resolução do Problema do Recobrimento corresponde à fase de Alocação das Tripulações, de acordo com a forma como o Problema da Programação de Tripulações Aéreas está sendo tratado neste trabalho. Tal solução corresponde a um subconjunto de jornadas obtido a partir de um conjunto de jornadas de grandes dimensões. Este subconjunto deve satisfazer a todas as restrições apresentadas na seção 3.1.2.
Portanto, uma solução do Problema da Programação de Tripulações Aéreas pode ser vista como um subconjunto de jornadas, sendo que cada jornada possui custo, tempo de vôo, base e um conjunto de tarefas associado. De forma a simplificar a representação, a forma como o custo pode ser calculado será omitida.

A Figura 3.1 ilustra a representação de uma solução de um Problema de Recobrimento fictício, contendo 100 tarefas por 74352 jornadas. O formato de matriz de bits foi substituído a fim de reduzir o tamanho dos arquivos, tanto de entrada quanto de solução. Acima, para facilitar a visualização, as colunas representam as tarefas e as linhas representam as jornadas (se fosse utilizado o formato original, as colunas representariam as jornadas e as linhas representariam as tarefas). Segundo a Figura 3.1, cada linha (jornada) possui, respectivamente:

- Número da jornada;
- Custo da jornada;
- Base em que a jornada pertence;
- Tempo de vôo total da jornada (minutos);
- Quantidade *n* de tarefas realizadas pela jornada;
- Lista dos números das *n* tarefas realizadas pela dada jornada.

```
369 2301 1 345 8 1 17 30 39 47 56 72 82
1596 1611 1 180 4 1 18 50 75
2131 1389 1 120 4 1 29 32 43
2839 1747 1 210 4 1 33 48 85
4615 2586 3 420 9 2 5 6 37 40 73 88 96 100
17919 1139 2 90 3 3 11 35
27308 2200 1 285 5 4 16 31 89 99
39242 1880 1 270 7 11 20 34 41 59 62 71
42692 1621 2 210 5 13 21 53 61 66
48762 1891 2 315 5 15 25 36 73 91
56679 1601 1 180 4 54 70 74 95
56734 1585 1 180 4 58 65 69 76
56868 1290 1 165 3 64 81 93
56891 1501 3 210 3 67 78 84
56944 1464 3 180 4 83 88 92 98
57087 1887 2 240 5 7 14 79 87 90
57247 1576 2 180 3 7 19 52
57508 2276 2 375 5 7 22 25 45 51
58916 1385 2 150 3 7 26 46
59488 1608 2 210 4 7 28 38 60
62990 2318 1 375 6 8 49 55 63 73 80
65957 2508 3 375 8 9 24 44 55 77 86 94 97
67808 2056 3 390 5 10 23 30 57 68
73377 1288 2 120 3 12 27 42
```

Figura 3.1 – Representação da Solução do Problema

Mais tarde, na seção 4.5, será visto como esta mesma solução é apresentada pelo sistema desenvolvido.

O dado de entrada para o Problema de Recobrimento (PR) é o conjunto do qual a solução foi extraída, e possui o mesmo formato da solução apresentada acima. O único dado adicional é que deve ser apresentada a distribuição desejada para as bases. A Figura 3.2 mostra um exemplo de dado de entrada (ou arquivo de entrada) para o PR. Nesta figura, a primeira linha exibe a distribuição desejada para as bases, que é apresentada da seguinte forma: número da base seguido da sua distribuição desejada.

```
3 1 0.400000 2 0.300000 3 0.300000
1 1197 1 90 3 1 17 30
2 1275 1 105 4 1 17 30 34
3 1598 1 180 5 1 17 30 34 41
4 1829 1 225 6 1 17 30 34 41 50
5 2107 1 300 7 1 17 30 34 41 50 75
6 2358 1 375 8 1 17 30 34 41 50 75 84
7 2106 1 285 7 1 17 30 34 41 50 87
8 2213 1 315 8 1 17 30 34 41 50 87 90
.....

74346 1535 2 195 3 12 83 91
74347 1408 2 150 3 12 83 98
74348 1285 2 120 3 12 83 100
74349 1520 2 150 3 12 90 97
74350 1521 2 150 3 12 90 99
74351 1523 2 150 3 12 94 97
74352 1524 2 150 3 12 94 99
```

Figura 3.2 – Formato do Dado de Entrada (Arquivo de Entrada)

3.3. Função de Avaliação

Cada solução S produzida é avaliada com base na função dada pela equação (4), a seguir:

$$FO(S) = t \cdot k + \sum_{j \in S} c_j \qquad (4)$$

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.

3.4. Geração de uma solução inicial

As implementações realizadas demonstraram que a qualidade da solução inicial não necessita ser muito apurada, pois a fase de refinamento rapidamente atinge para bons resultados, independentemente da qualidade dessa solução.

A geração adotada é baseada na construção realizada através de um método guloso, conforme proposto por Chvátal (1979). 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 3.3 – Algoritmo "GerarSolucaoInicial"

A Figura 3.3 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.

3.4.1. Cálculo do Custo-benefício das Colunas (Jornadas)

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 (5):

$$Custo\,Benefício = \frac{Custo\ da\ Coluna}{Nro\ de\ linhas\ não\ cobertas} + (Nro\ de\ linhas\ \ já\ cobertas) \cdot k \qquad (5)$$

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

Para desempenho ideal dos algoritmos que serão vistos na 3.5, 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.

A Figura 3.4 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 3.4 – Algoritmo "CB", que calcula o custo-benefício de cada coluna

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.

3.5. Algoritmo Proposto

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. 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. A Figura 3.5 ilustra este processo.

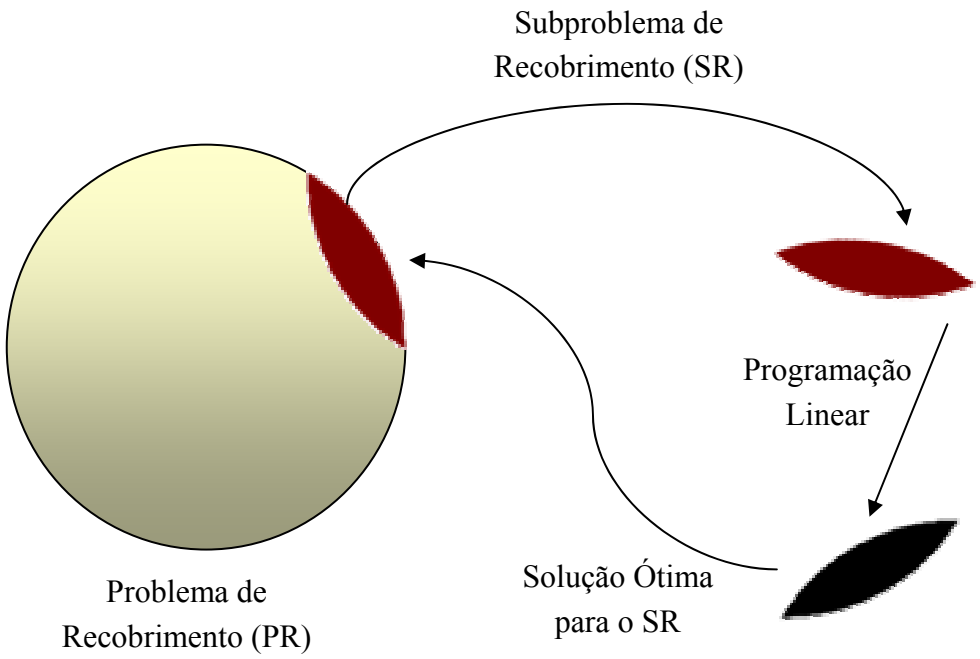


Figura 3.5 – Ilustração do Algoritmo

O método proposto neste trabalho consiste em uma variação no algoritmo de Rubin. Esta variação consiste em 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 de forma exata. O algoritmo proposto para a resolução do Problema do Recobrimento é apresentado na Figura 3.6.

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. $S' \leftarrow$ *RefinamentoRestritoSucessivo* (*S', S'', desvio*); {vide seção 3.6}
15. $S \leftarrow S \cup S'$;
16. fim-enquanto;
17. retorne *S;*

Fim BuscaRestrita;

Figura 3.6 – Algoritmo BuscaRestrita

Na Figura 3.6, a linha 14 faz alusão ao procedimento *RefinamentoRestritoSucessivo*. Este procedimento é o responsável pela resolução de cada SR. Seus parâmetros são: conjunto de colunas retiradas da solução, conjunto de colunas que farão parte do SR e o desvio admitido para as restrições de distribuição das bases. Este procedimento será estudado na seção 3.6.

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 3.6, 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 3.7, 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%.

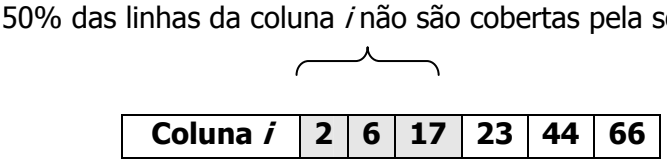


Figura 3.7 – Porcentagem de linhas de uma coluna

3.6. Refinamento Restrito Sucessivo

Cada subproblema de recobrimento (SR) é resolvido de forma exata, ou seja, a solução ótima para cada SR é obtida. 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.

O algoritmo *RefinamentoRestritoSucessivo* aludido na linha 14 da Figura 3.6 consiste em resolver um subproblema de recobrimento, retornando um resultado. Conforme visto

anteriormente, um SR é um problema de recobrimento com uma característica especial: nem todas as linhas precisam ser cobertas pelo conjunto solução, sendo que, inclusive, a cobertura de uma destas linha é desencorajada.

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, pelas equações (6) – (10).

$$\textit{Minimizar} \quad \sum_{j=1}^{n} c_j x_j + \sum_{i=1}^{m} s_i k \tag{6}$$

$$\textit{Sujeito a:} \quad \sum_{j=1}^{n} \left(a_{ij} x_j \right) + r_i - s_i = 1 \qquad \forall\, i = 1,\ldots,m \tag{7}$$

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

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

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

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*, então a_{ij} será igual a 1. Caso contrário, a_{ij} será igual a 0. Se esta linha *i* for coberta pelo conjunto solução corrente, então r_i será igual 1. Caso contrário, r_i será igual a 0. Desta forma, r_i indica se a linha *i* já está sendo coberta pela solução corrente. Assim, a restrição (7) 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. 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.

3.6.1. 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.

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

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. Sistema Desenvolvido

O sistema desenvolvido é denominado "SCP Solver" (Set Covering Problem Solver). Ele foi desenvolvido na linguagem C++ com a ferramenta C++ Builder 6.0, da Borland. Essa escolha se baseou no poder e desempenho da linguagem e na disponibilidade de recursos da ferramenta. O sistema foi implementado segundo o paradigma de Orientação a Objetos. Além disso, o sistema interage com o software de otimização Lingo, da Lindo Systems. O Lingo é o responsável pela resolução de problemas por meio de programação matemática.

O sistema tenta oferecer um ambiente de uso agradável, flexível e simples. Para isso, o sistema foi desenvolvido com uma interface bem parecida com a de "softwares comuns", de maneira que não necessite de um usuário "especial" para operá-lo. Ele permite ainda que o usuário faça ajustes nos parâmetros em tempo real. Desta forma, é possível inclusive explorar o espaço de soluções de diversas formas.

O funcionamento básico do sistema pode ser descrito em quatro etapas:

(a) **Configurar Parâmetros**: permite ao usuário alterar os parâmetros de execução dos algoritmos;

(b) **Ler Colunas:** carrega o arquivo de entrada;

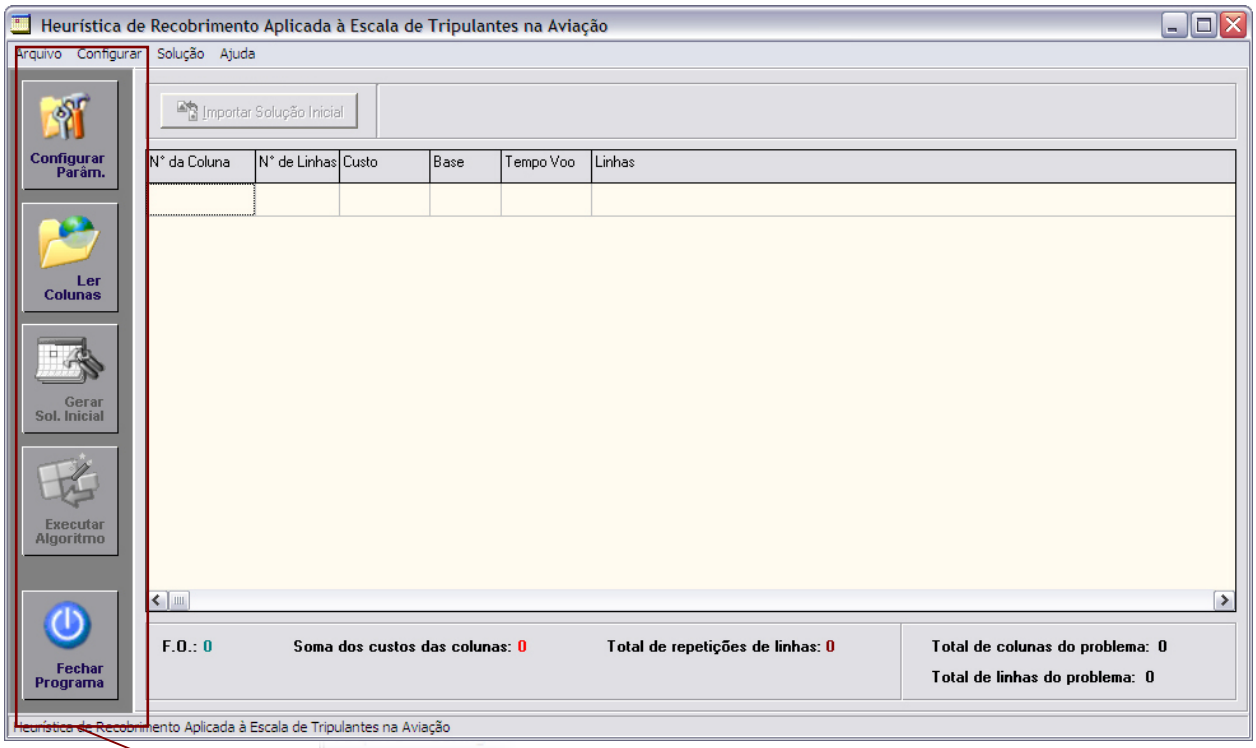
(c) **Gerar Solução Inicial**: gera uma solução inicial para o problema;

(d) **Executar Algoritmo**: executa o algoritmo segundo os parâmetros configurados;

As etapas acima são detalhadas com ilustrações a seguir, a fim de que haja uma melhor compreensão sobre o funcionamento do sistema.

4.1. Tela Inicial

A primeira tela a ser vista ao abrir o sistema é a Tela Principal. Esta tela possui todas as ferramentas necessárias para que o usuário utilize os recursos do sistema. A navegação pelo sistema pode ser feita por meio dos botões laterais (Figura 4.1), ou pelo menu no topo da Tela.



(a) **Configurar Parâmetros**: permite ao usuário alterar os parâmetros de execução dos algoritmos;

(b) **Ler Colunas:** carrega o arquivo de entrada;

(c) **Gerar Solução Inicial**: gera uma solução inicial para o problema;

(d) **Executar Algoritmo**: executa o algoritmo segundo os parâmetros configurados;

(e) **Fechar Programa**.

Figura 4.1 – Tela Principal do Programa

Na Figura 4.2 são apresentados os itens de cada um dos menus.

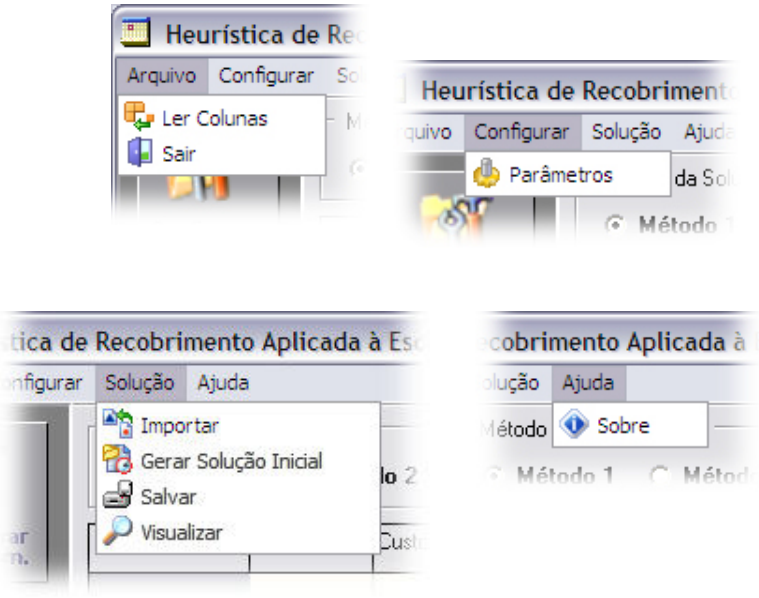


Figura 4.2 – Principais itens do menu de navegação

4.2. Configurar Parâmetros

A tela Configurar Parâmetros pode ser acessada através dos botões laterais (Figura 4.1) ou pelo menu de navegação (Figura 4.2).

As subseções seguintes serão descritas cada uma das abas da tela Configurar Parâmetros. A qualquer momento, esta tela pode ser fechada. Os botões seguintes (Figura 4.3) estão sempre à disposição do usuário nesta tela, e efetuam as seguintes funções:

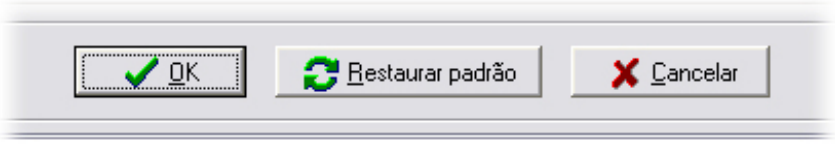


Figura 4.3 – Itens do menu de navegação

Clicando no botão "OK", os valores dos parâmetros são alterados para os valores apresentados na tela, e a tela é fechada. As alterações são salvas automaticamente.
Clicando no botão "Restaurar padrão", os campos são preenchidos com os valores padrão.
Clicando no botão "Cancelar", as alterações realizadas pelo usuário não têm efeito sobre os parâmetros, e a tela é fechada.

4.2.1. Critério de Parada

A Figura 4.4 mostra a definição de um critério de parada. A qualquer momento, durante a execução do algoritmo, o usuário pode interromper a sua execução. Essa interrupção pode ser feita pelo sistema, se o usuário marcar a caixa "Limitar o tempo máximo de execução". Este tempo é definido logo abaixo, no campo de texto seguido de "Tempo máximo de execução".

É possível também definir a semente de números aleatórios que será utilizada. Este parâmetro é útil quando se está submetendo o sistema a testes.

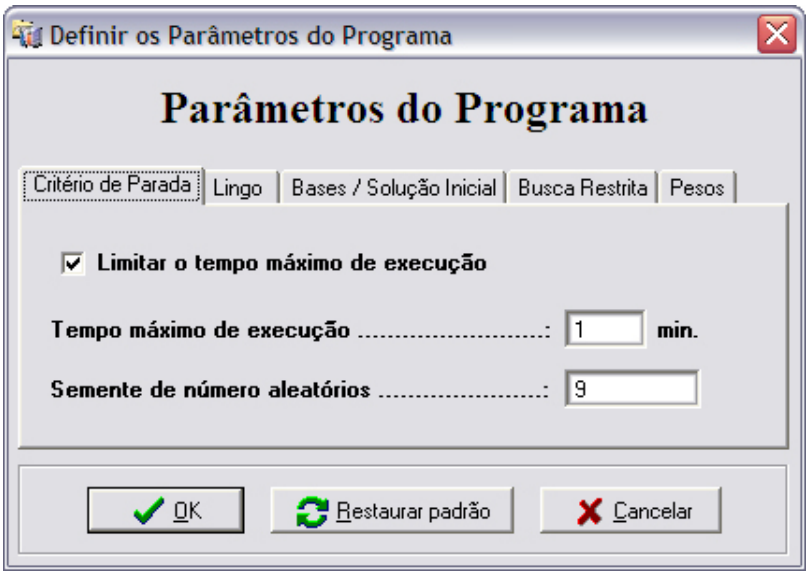


Figura 4.4 – Parâmetros: Critério de Parada

4.2.2. Lingo

A Figura 4.5 mostra os parâmetros para execução do Lingo. Esta tela permite ao usuário limitar o tempo máximo de execução de cada iteração do Lingo (tempo máximo para resolução exata de um subproblema de recobrimento). Este tempo é dado em segundos.

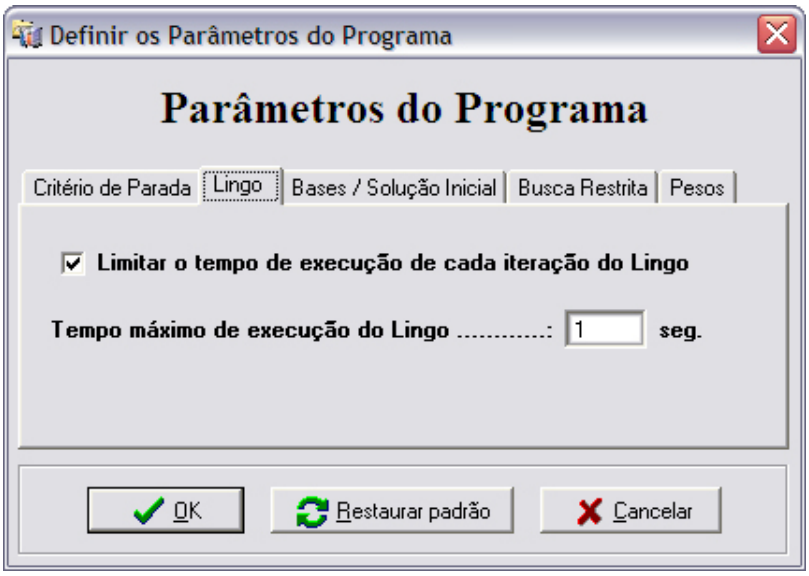


Figura 4.5 – Parâmetros: Lingo

4.2.3. Bases / Solução Inicial

A Figura 4.6 mostra os parâmetros relacionados às Bases Esta tela permite ao usuário definir se as restrições de base serão consideradas ou não, bem como definir o valor do desvio admitido na distribuição das bases. Existe ainda a opção de definir um desvio diferente para a solução inicial (este parâmetro é útil para testes), entretanto, o desvio considerado durante a geração da solução inicial deve ser menor do que o considerado para o problema.

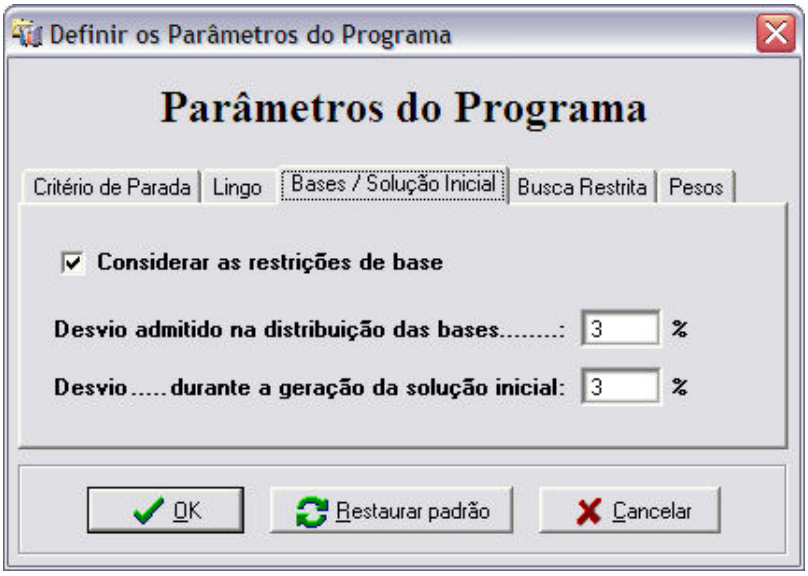


Figura 4.6 – Parâmetros: Bases / Solução Inicial

4.2.4. Busca Restrita

A Figura 4.7 mostra os parâmetros relacionados ao algoritmo de BuscaRestrita.

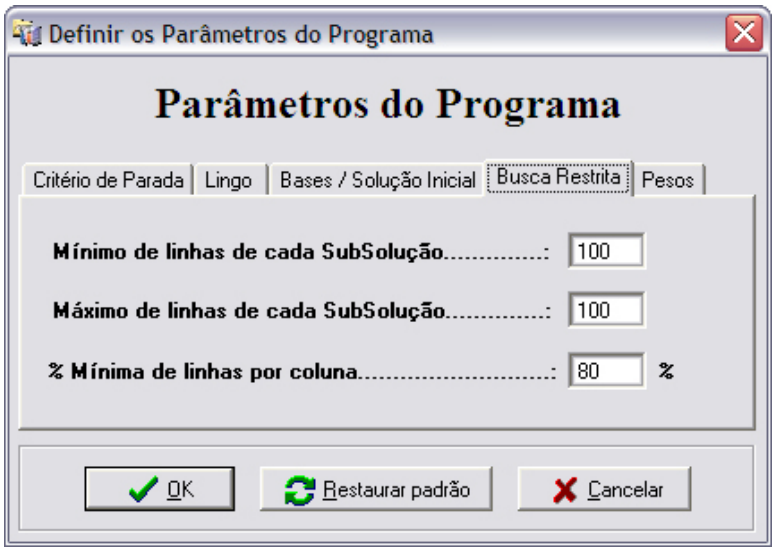


Figura 4.7 – Parâmetros: Busca Restrita

Os parâmetros exibidos na Figura 4.7 são explanados pela seção 3.5, e são:

- Número mínimo de linhas de cada SubSolução (*minlinhas*);
- Número máximo de linhas de cada SubSolução (*maxlinhas*);
- Porcentagem mínima de linhas por coluna (*pct*);

4.2.5. Pesos

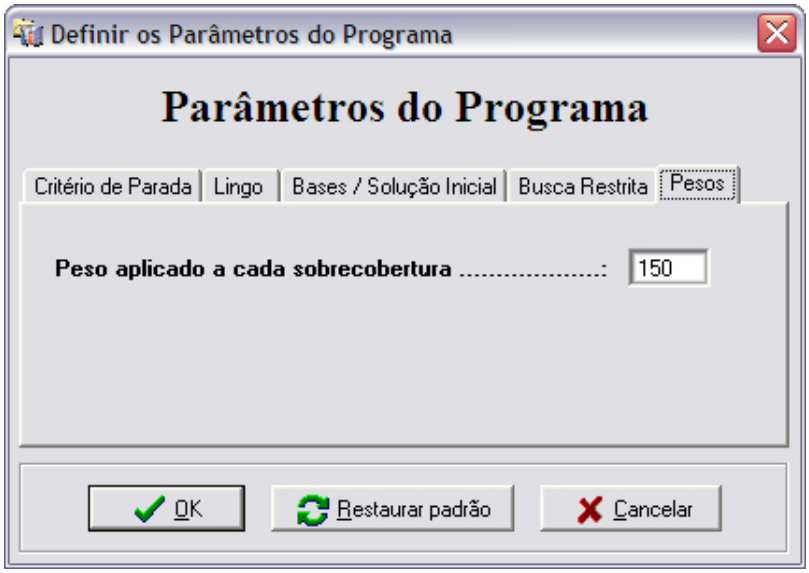


Figura 4.8 – Parâmetros: Pesos

A Figura 4.8 mostra o único peso considerado, associado a cada sobrecobertura.

4.3. Ler Colunas

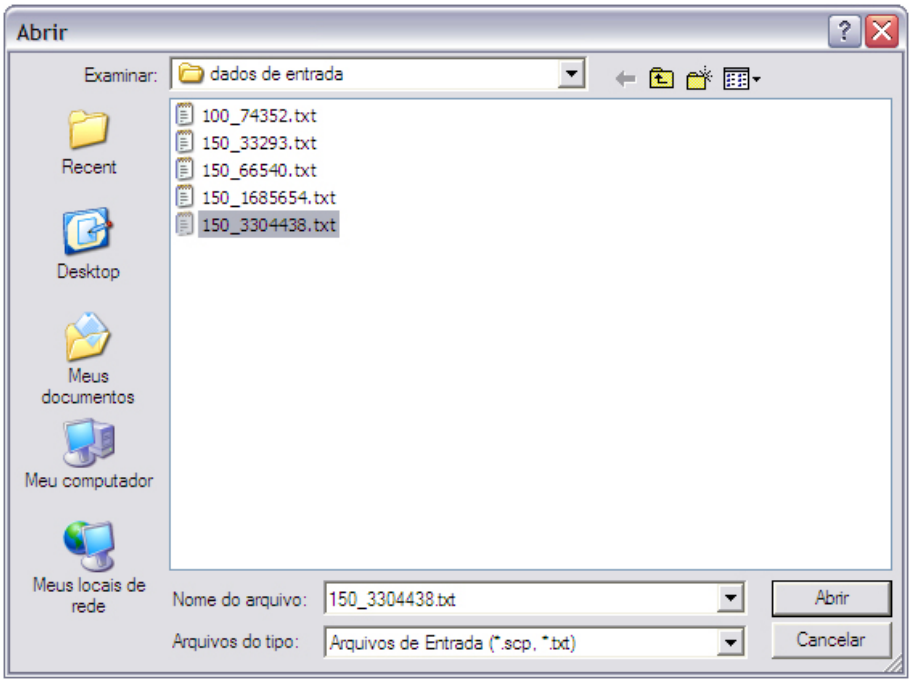


Figura 4.9 – Ler Colunas (Arquivo de Entrada)

Ao acessar o botão "Ler Colunas" (Figura 4.1), o usuário deve selecionar um arquivo de entrada válido, com a extensão .scp ou .txt. A Figura 4.9 mostra a tela de seleção de arquivo.

4.4. Gerar Solução Inicial

Após o arquivo de entrada ser carregado, o próximo passo é gerar ou importar uma nova solução inicial. O sistema possibilita ao usuário importar uma solução, ao invés de gerar uma solução inicial por meio do método descrito na seção 3.4. Para gerar uma solução inicial, basta clicar no botão intitulado "Gerar Solução Inicial" (Figura 4.1), ou se dirigir ao menu "Solução" e clicar em "Gerar Solução Inicial" (Figura 4.2). Para importar uma solução, o usuário deve colocar o mouse sobre o menu "Solução" e clicar em "Importar" (Figura 4.2). Uma tela idêntica à tela da Figura 4.9 se mostrará, e o usuário deverá selecionar o arquivo que contém a solução. Caso o arquivo contenha uma solução inválida para o problema considerado, o sistema apenas informará ao usuário sobre a situação e não importará a solução.

4.5. Tela Principal

Assim que a solução inicial for gerada, a tela principal trará uma série de informações sobre o problema que está sendo considerado e sobre a solução atual. A Figura 4.10 mostra a tela principal após a geração (ou carregamento) da solução inicial. Esta figura mostra detalhes sobre as colunas da solução (número da coluna, número de linhas, custo, base, tempo de vôo e o número das linhas).

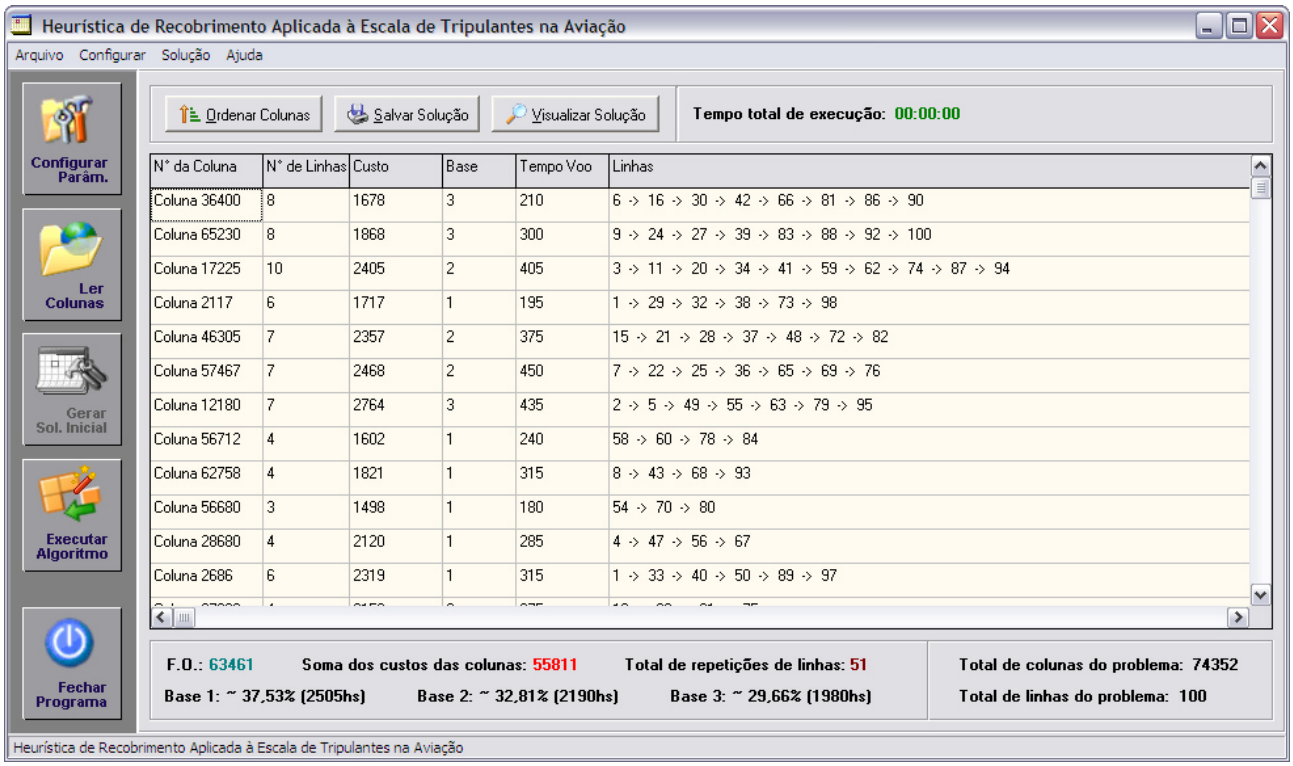


Figura 4.10 – Tela Principal após a geração da solução inicial

A Figura 4.11 mostra o valor atual (FO) da solução, a soma dos custos das colunas e o número de repetição de linhas (sobre-coberturas) na solução.

Figura 4.11 – Características da Solução

A Figura 4.12 mostra as características do PR considerado: número de linhas (tarefas) do problema e número de colunas (jornadas) do problema.

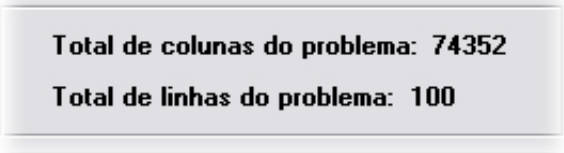


Figura 4.12 – Características do Problema

4.6. Executar Algoritmo

A etapa seguinte à geração de uma solução inicial é a fase de refinamento restrito sucessivo. Esta etapa se inicia após o usuário clicar no botão "Executar Algoritmo" (Figura 4.1). Durante a execução do algoritmo, a tela representada pela Figura 4.13 é exibida. Esta tela é exibida no centro da tela, mantendo sempre visível ao usuário as características da solução atual (Figura 4.14).

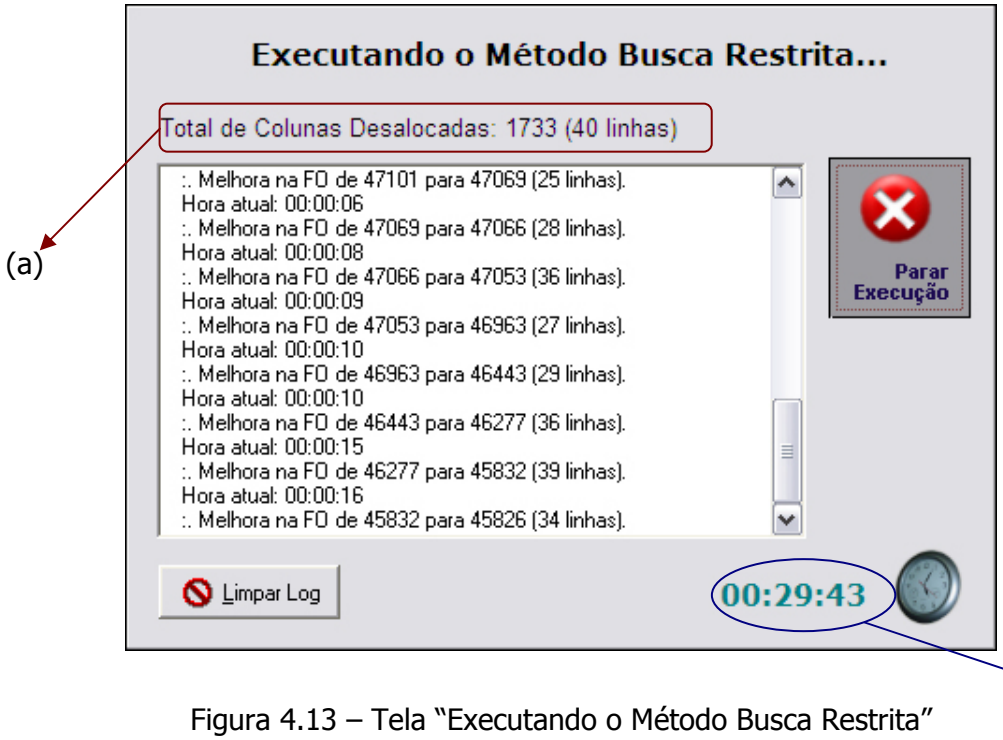


Figura 4.13 – Tela "Executando o Método Busca Restrita"

Observe, na Figura 4.13, que o usuário é informado todo o tempo sobre as características de cada SR gerado (a), bem como sobre as iterações do refinamento restrito sucessivo em que houve melhora na solução. Nesta figura, em (b) é exibido o tempo restante até que o critério de parada seja atingido. Caso não haja critério de parada, o indicador de tempo (b) cresce indefinidamente, até que o usuário clique em "Parar Execução", e o relógio à esquerda de (b) é ocultado.

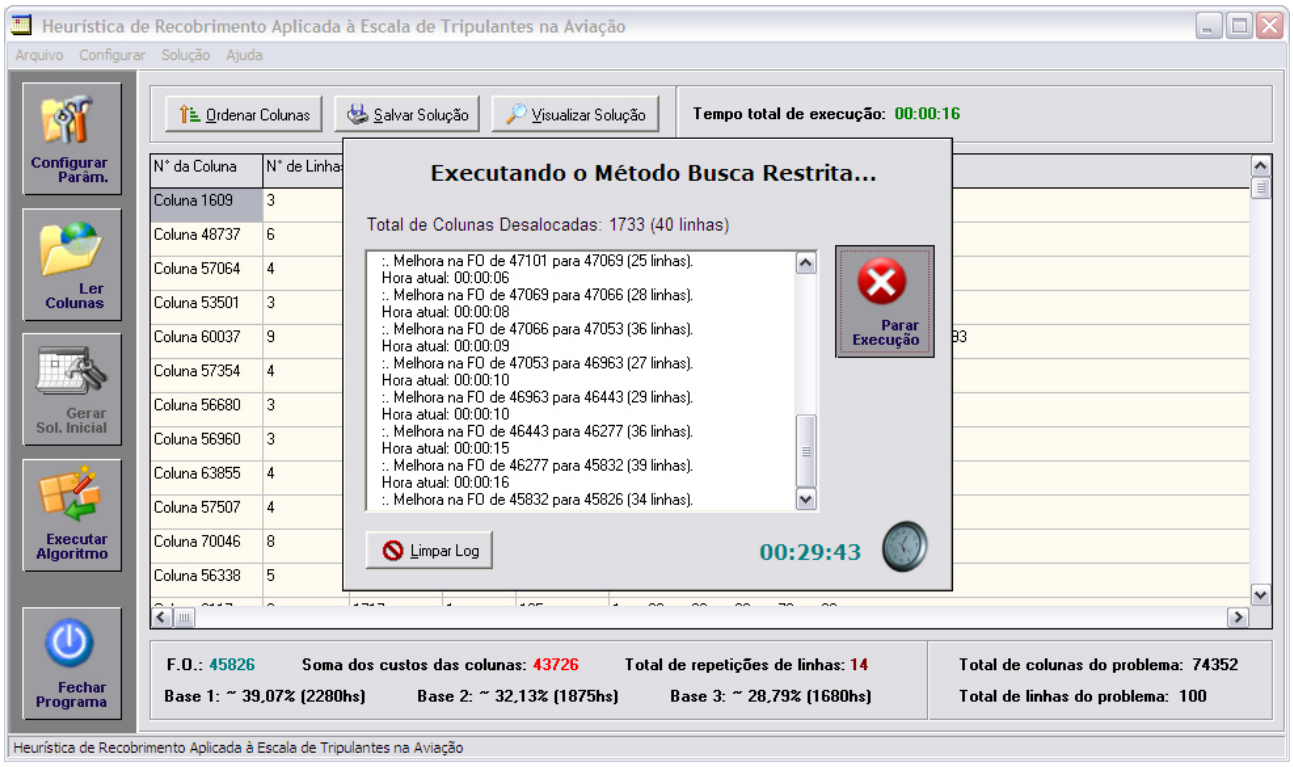


Figura 4.14 – Tela Principal durante a execução do "Método Busca Restrita"

4.7. Exportar Solução

Uma vez gerada a solução final, o usuário pode exportar essa solução. A Figura 4.15 mostra os botões da parte superior da tela principal. Clicando no botão "Ordenar Colunas", as colunas da solução são ordenadas pelo seu número, clicando em "Salvar Solução", é aberta a janela da Figura 4.16, para que o arquivo solução seja salvo, e clicando em "Visualizar Solução" um arquivo de texto temporário contendo a solução é aberto no Bloco de Notas do Windows.

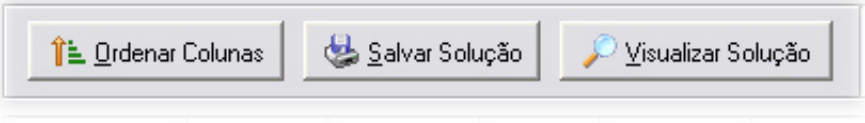


Figura 4.15 – Botões utilizados para exportar uma solução

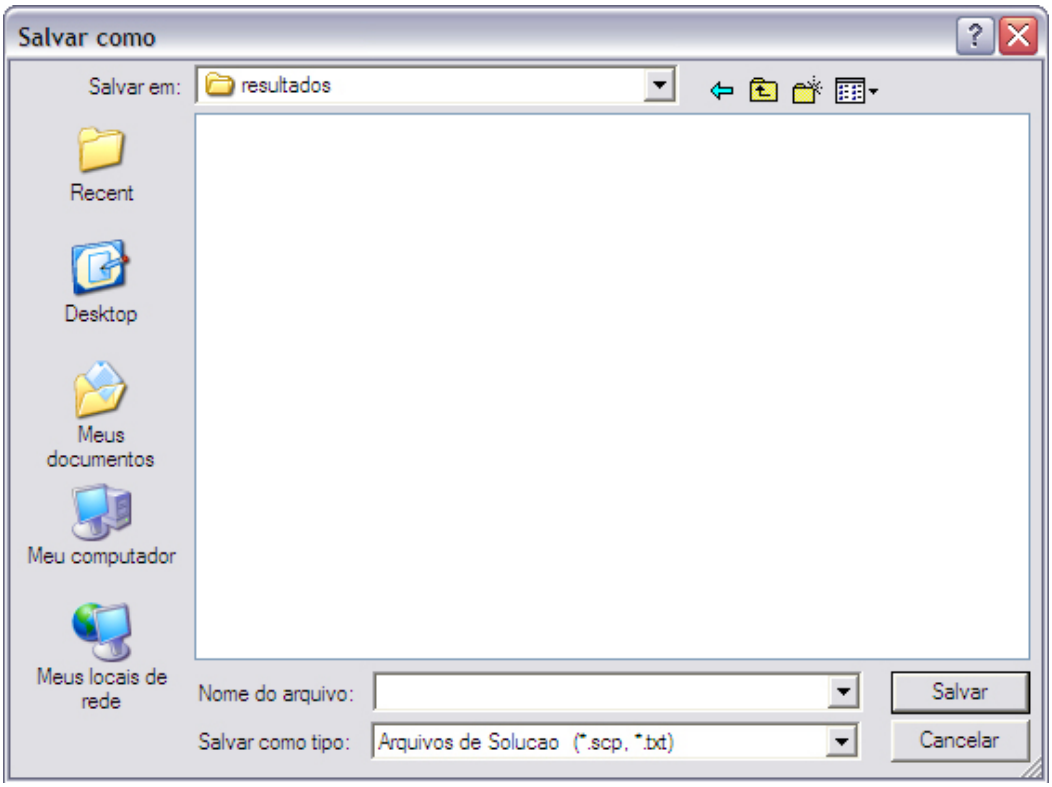


Figura 4.16 – Janela "Salvar Solução"

5. Resultados

Os algoritmos foram desenvolvidos na linguagem C++ com a IDE e o compilador do Borland C++ Builder 6.0, e 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.

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: 40% do tempo total de vôo deve partir da base 1, 30% da base 2 e 30% da base 3. A Tabela 5.1 mostra as características dessas instâncias.

Tabela 5.1: 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 5.2 mostra os valores dos parâmetros e pesos adotados para cada instância.

Tabela 5.2: 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

A Tabela 5.3 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.

Tabela 5.3: Desempenho do algoritmo

Item	**Instância**				
	Inst_1	**Inst_2**	**Inst_3**	**Inst_4**	**Inst_5**
FO^H	44.808	73.645	70.856	67.890	67.619
Desvio (*gap*) FO^H	0,002%	0,3%	0,22%	0,44%	1,52%
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).
(4) O Lingo foi incapaz de resolver, ocorrendo estouro de memória.

Pela Tabela 5.3, 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 5.4 mostra as características da melhor solução obtida em cada instância. Na tabela 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.

Tabela 5.4: Características das melhores soluções obtidas

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
Valor da FO	44.808	73.645	70.856	67.890	67.619

A Figura 5.1 ilustra a evolução típica do melhor valor da função de avaliação (FO) nos 30 minutos de processamento em uma execução do método utilizando programação matemática aplicado à instância **Inst_4**. Observa-se nesta figura que o método gera soluções de boa qualidade rapidamente.

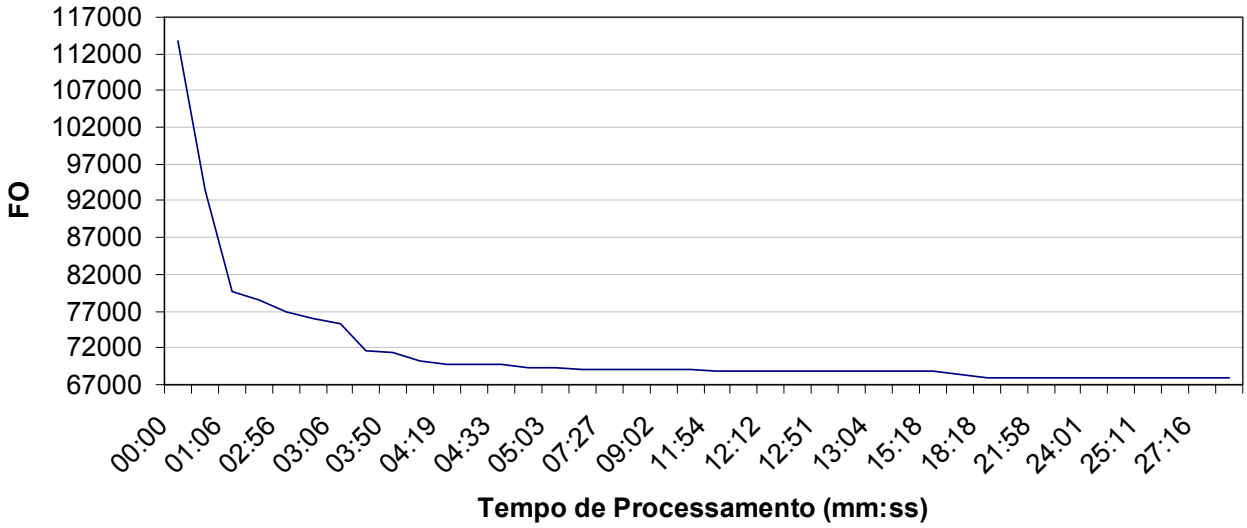


Figura 5.1 – Evolução típica do melhor valor da função de avaliação

6. Conclusões

Este trabalho propõe uma metodologia 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.

Referências Bibliográficas

ANBIL, R.; TANGA, R.; JOHNSON, E. (1992) - A Global Approach to Crew Pairing Optimization. IBM System Journal.

BALL, M. & BENOIT-THOMPSON, H. (1988) - A lagrangean relaxation based heuristic for the urban transit crew scheduling problem. In *Computer-Aided Transit Scheduling*, Daduna, J. R.; Wren, A. (eds), Springer-Verlag, p. 54-67.

BARNHART, C.; JOHNSON, E. L.; NEMHAUSER, G. L.; SAVELSBERGH, M. P. & VANCE, P. H. (1998) - Branch-and-price: column generation for solving huge integer programs. Operations Research, v.46, p.316-329.

BEASLEY, J.; CAO, B. (1998) - A Dynamic Programming Based Algorithm for the Crew Scheduling Problem. *Computers and Operations Research*.

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 40 th International Symposium of the AGIFORS, p. 1-8, Istambul.

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

CARRARESI, P. & GALLO, G. (1984). - Network models for vehicle and crew scheduling. European Journal of Operational Research, v.16, 139-151.

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

CHVATAL, V. (1979) - A greedy heuristic for the set-covering problem. Mathematics of Operations Research, v. 4, n. 3, p. 233-235.

CLEMENT & WREN, A. (1995) - Greedy genetic algorithms, optimizing mutantis and bus driver scheduling. In *Computer-Aided Transit Scheduling*, Daduna, J. R.; Branco, I.; Paixão, J. M. P. (eds.), Springer-Verlag, p. 213-235.

DESROCHERS, M & SOUMIS, F. (1989) - A column generation approach to the urban transit crew scheduling problem. Transportation Science, v.23, n.1, p.1-13.

DESROCHERS, M; GILBERT, J.; SAUVE, M.; SOUMIS, F. (1992) - CREW-OPT: Subproblem modeling in a column generation approach to urban crew scheduling. In: Computer-Aided Transit Scheduling, Desrochers, M. & Rousseau, J. M. (eds.), Spring, Berlin, p. 395-406.

DIAS, T. G.; de SOUZA, J. P.; CUNHA, J. F. (2002) - Genetic algorithms for the bus driver scheduling problem: a case study. Journal of the Operational Research Society, v.53, n.3, p.324-335.

FEO, T. A.; RESENDE, M. G. C. (1995) - Greedy randomised adaptive search procedures. *Journal of Global Optimization*, v.6, p.109-133.

ELIAS, S. E. G. (1964) - The use of digital computers in the economic scheduling for both man and machine in public transport. Technical Report 49, Kansas State University Bulletin, Kansas, EUA.

FORES, S.; PROLL, L. & WREN, A. (1999) - An Improved ILP System For Driver Scheduling. In: Computer-Aided Transit Scheduling, Wilson, N. H. M. (ed.), Springer, Berlin , p. 43-61.

FORES, S.; PROLL, L. & WREN, A. (2002) – TRACS II: A hybrid IP/Heuristic Driver Scheduling System for Public Transport. Journal of the Operational Research Society, v. 53, p. 1093-1100.

FRIBERG, C.; HAASE, K. (1999) - An exact branch and cut algorithm for the vehicle and crew scheduling problem. In: Computer-Aided Transit Scheduling, Wilson, N. H. M. (ed.), Springer, Berlin, p. 63-80.

GLOVER, F. (1989) - Tabu Search: Part I. ORSA Journal on Computing, 1, p.190-206.
GOLDBERG, D. E. (1989) - Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Berkeley.

GRAVES, G.; McBRIDE, R.; GERSHKOFF, L.; ANDERSON, D.; MAHIDHARA, D. (1993) - Flight Crew Scheduling. *Management Science*.

HOLLOWAY, S. (1997) - Straight and Level: Practical Airline Economics. Ashgate Publications Company.

KIRKPATRICK, S.; GELLAT, D.C. & VECCHI, M.P. (1983) - Optimization by Simulated Annealing, *Science*, v. 220, p. 671-680.

KWAN, A. S. K.; KWAN, R. S. K.; WREN, A. (1999) - Driver scheduling using genetic algorithms with embedded combinatorial traits. In *Computer-Aided Transit Scheduling*, Wilson, N. H. M. (ed.), Springer-Verlag, Berlin, p. 81-102.

LOURENÇO, H. R.; MARTIN, O.; STÜTZLE, T. (2002) – Iterated Local Search. In: F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, Kluwer Academic Publishers, Boston, MA, USA.

LOURENÇO, H. R.; PAIXÃO, J. P. & PORTUGAL, R. (2001) – Metaheuristics for the bus-driver scheduling problem. *Transportation Science*, v. 35, p. 1093-1100.

MARINHO, E. H.; OCHI, L. S.; DRUMMOND, L. M. A.; SOUZA, M. J. F.; SILVA, G. P. (2004) – Busca Tabu aplicada ao Problema de Programação de Tripulações de Ônibus Urbano. In Anais do XXXVI Simpósio Brasileiro de Pesquisa Operacional, v. 1, p. 1471-1482.

MANINGTON, B. & WREN, A. (1975) - A general computer method for bus crew scheduling. Preprints of the Workshop on Automated Techniques for Scheduling of Vehicle operators for Urban Public Transportation Services, Chicago.

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.

MINGOZZI, A.; BOSCHETTI, M.; RICCIARDELLI, S.; BIANCO, L. (1999) - A Set Partitioning Approach to the Crew Scheduling Problem. *Operations Research*.

MLADENOVIC, N. & HANSEN, P. (1997) - Variable neighborhood Search. Computers and Operations Research, v. 24, p. 1097 – 1100.

PIMENTEL, A. L. G. (2005) - Uma Abordagem Heurística para a Solução de Problemas de Recobrimento de Conjuntos de Grande Porte, com Aplicação à Alocação de Tripulações para Companhias Aéreas. Tese de doutorado, Programa de Engenharia de Engenharia de Sistemas e Computação, COPPE/UFRJ, Rio de Janeiro.

ROUSSEAU, J. M. & BLAIS , J. Y. (1985) - HASTUS: an interactive system for buses and crew scheduling. In: Computer scheduling of public transport 2, J. M. Rousseau (ed.), p. 45-60, North-Holland.

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

RYAN, D. (1992) – The solution of the massive generalized set partitioning problems in aircrew rostering. Journal of the Operational Research Society, v. 43, n. 5, p. 459-467.
SHEN, Y. & KWAN, R. S. K. (2001) - Tabu Search for driver scheduling. Berlin, *In* Voß, S; Daduna, J. (eds.), p. 121-135.

SHEN, Y. & KWAN, R. S. K. (2000) – Tabu Search for Time Windowed Public Transport Driver Scheduling. Report 2000.14, Research Report Series, School of Computer Studies, University of Leeds.

SIQUEIRA, P. H. (1999) - Aplicação do algoritmo do matching no problema da construção de escalas de motoristas e cobradores de ônibus. Dissertação de mestrado, Setor de Tecnologia e de Ciências Exatas, Universidade Federal do Paraná.

SILVA, G. P.; SOUZA, M. J. F. & ALVES, J. M. C. B. (2002a) - Resolução do Problema de Programação Diária da Tripulação de Ônibus Urbano via Simulated Annealing. In XVI Congresso de Pesquisa e Ensino em Transportes. Panorama Nacional de Pesquisa em Transportes. Rio de Janeiro: ANPET, 2002, v. 2, p. 95-104.

SILVA, G. P.; SOUZA, M. J. F. & ALVES, J. M. C. B. (2002b) - Simulated Annealing Approach to Solve the Daily Crew Scheduling Problem. In Proceedings of the IV ALIO/EURO Workshop on Applied Combinatorial Optimization. Pucon: Universidad de Concepción, 2002, v.1, p. 127-129.

SMITH, B. M. & WREN, A. (1988) - A bus crew scheduling system using a set covering formulation. *Transportation Research*, v. 22A, p. 97-108.

SOUZA, M. J. F.; SILVA, G. P.; MAPA, S. M. S. (2004) – Métodos de Pesquisa em Vizinhança Variável Aplicados à Resolução do Problema de Programação Diária de Ônibus Urbano. In Panorama Nacional da Pesquisa em Transportes, XVIII Congresso de Pesquisa e Ensino em Transportes, v. 2, p. 1492-1502.

SOUZA, M. J. F.; MAPA, S. M.; RODRIGUES, M. M. S. & SILVA, G. P. (2003a) - Um estudo das heurísticas Simulated Annealing e VNS aplicadas ao problema de programação de tripulações. In XXIII Encontro Nacional de Engenharia de Produção. Ouro Preto: ABEPRO, 2003, 8 p.

SOUZA, M. J. F.; CARDOSO, L. X. T. & SILVA, G. P. (2003b) - Programação de tripulações de ônibus urbano: uma abordagem heurística. In XXXV Simpósio Brasileiro de Pesquisa Operacional. Natal: SOBRAPO, 2003, p. 1285-1294.

WEDELIN, D. (1995) - An Algorithm for Large Scale 0-1 Integer Programming with Application to Airline Crew Scheduling. *Annals of Operations Research*.

WREN, A. (1996) - Scheduling, timetabling e rostering – a special relationship?. In Burke, E. K. And Ross, P. (eds), Practice and Theory of Automated Timetabling, Springer-Verlag, p. 46-75.

WREN, A.; KWAN, R. S. K. & PARKER, M. E. (1994) - Scheduling of rail driver duties. In: Computers in Railways IV, vol. 2, Murty, T. K. S.; Mellitt, B.; Brebbia, C. A.; Sciutto, G. & Sone, S. (eds.), Southampton, Boston.

WREN, A. & ROUSSEAU, J. M. (1995)- Bus driver scheduling – An overview. In *Computer-Aided Transit Scheduling*, Daduna, J. R.; Branco, I.; Paixão, J. M. P. (eds.), Springer-Verlag, p. 173-187.

YUNES, Tallys. (2000) Problemas de Escalonamento no Transporte Coletivo: Programação por Restrição e Outras Técnicas, UNICAMP, Campinas.

VAZ, Glauber José. (2003) Uma abordagem alternativa para os escalonamentos de ônibus e de motoristas, UNICAMP, Campinas.

ANEXO A
Trabalhos publicados

Como resultado deste projeto de pesquisa foram submetidos artigos ao Simpósio Brasileiro de Pesquisa Operacional (SBPO 2006) e ao Simpósio de Pesquisa Operacional e Logística da Marinha (SPOLM 2006), na categoria "trabalhos completos". Ambos os artigos foram aceitos para publicação.

Anexo o artigo aceito para publicação nos anais do XXXVIII Simpósio Brasileiro de Pesquisa Operacional, a ocorrer em Goiânia (GO), no período de 12 a 15 de setembro de 2006.

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO - DECOM
CAMPUS UNIVERSITÁRIO - MORRO DO CRUZEIRO
CEP 35400-000 - OURO PRETO - MINAS GERAIS