MODELOS E ALGORITMOS PARA ALOCAÇÃO DE TRIPULAÇÃO EM REDES DE TRANSPORTE

Geraldo Robson Mateus
Jayme Assunção Casimiro
Universidade Federal de Minas Gerais
Av. Presidente Antônio Carlos, 6627
Campus Pampulha - CEP 31270-010
Belo Horizonte - MG – Brasil
mateus , jayme@dcc.ufmg.br

Abstract

The main objective of *Crew Scheduling Problem* (***CSP***) is group the tasks on feasible duties, select a minimum cost of duties that cover all the tasks and, finally, produce one or more rosters (set of duties that can be executed by a unique crew) and assign them for the crews. The procedure to solve the ***CSP*** is motivated by the fact that about 30% of the expenses of mass transit systems (buses, airlines or railway companies) are to pay their employees. So, algorithms, which are able of proposing good solutions to this problem, are always explored. After all, profits, even though when they are small, in some cases, could result in a considerable save of expenses. To figure out the *Crew Scheduling problem* we split it into three phases: *Column Generation, Set Covering and Crew Rostering*. We took the results from the *Column Generation* algorithm and applied *Set Covering* and *Crew Rostering* consecutively to achieve the final desired objective yet described. Moreover, we make use of simple (free of so many details) but efficient algorithms to achieve solutions of good quality for both problems. On this article we will show models, algorithms and report computational results for a set of problems in the Literature and real-world instances.
Key-words: Crew Scheduling, Set Covering, Crew Rostering.

Resumo

O escalonamento de mão-de-obra (crew scheduling) é uma tarefa bastante rotineira no contexto de grandes empresas de transporte. Tal atividade envolve, basicamente, subdividir um determinado conjunto de tarefas entre diferentes tripulações respeitando os direitos contratuais de cada classe de funcionários. A realização de um planejamento racional do escalonamento de funcionários é de suma importância e tem sido perseguida tanto na área científica quanto na empresarial. Afinal, grande parte do custo naturalmente despendido por empresas de transporte destina-se ao pagamento de funcionários. Logo, a formulação de modelos e algoritmos capazes de propor soluções melhores que as já disponíveis, mesmo considerando melhoras de pequeno porte, pode resultar em ganhos extremamente consideráveis. Para resolver o problema de *Crew Scheduling* o mesmo foi dividido em três fases: *Geração de Colunas, Set Covering and Crew Rostering*. Os resultados da fase *Geração de Colunas* foram aplicados ao problema de *Set Covering* e *Crew Rostering* consecutivamente de modo a se alcançar o objetivo final desejado já especificado. Além disso, os algoritmos a serem apresentados mostraram-se de simples implementação e entendimento sem, no entanto, prejudicar a qualidade das soluções finais obtidas.
Palavras-Chaves: Crew Scheduling, Set Covering, Crew Rostering.

1. Introdução

Várias abordagens têm sido utilizadas para a resolução do *CSP*. Em [1], Smith salienta que, em múltiplas aplicações, a fase de *crew scheduling* é precedida pela de *vehicle scheduling*. Assim, baseado na tabela de escalonamento encontrada para cada veículo de uma frota, períodos de descanso e troca de mão-de-obra são determinados visando a composição de jornadas viáveis de trabalho.

A abordagem aqui utilizada consiste em dividir o problema de *crew scheduling* em três partes principais: ***geração de jornadas, set covering*** e ***crew rostering***:

(a) *Geração de Jornadas* – um grande número de jornadas é gerado através da associação de tarefas afins e possíveis de serem executadas consecutivamente por uma única tripulação. Essas jornadas devem respeitar regras trabalhistas impostas por uma classe de funcionários.

(b) *Set Covering* – é um problema clássico de otimização combinatória utilizado como modelo em inúmeras aplicações, em particular ***Crew Scheduling***. No contexto aqui aplicado, ele pode ser resumidamente descrito da seguinte maneira: apresentando como dado de entrada um conjunto de tarefas a serem executadas por uma dada tripulação e um conjunto de jornadas viáveis capazes de cobrir essas tarefas a um determinado custo, esse problema procura por um conjunto de jornadas capazes de executar toda a demanda a um custo mínimo.

(c) *Crew Rostering* – recebendo como entrada as jornadas anteriormente obtidas por meio da solução de ***(b)***, esse problema objetiva a criação de *rosters* (seqüência de jornadas) possíveis de serem executados por uma única tripulação dentro de um determinado intervalo de tempo. Além disso, para a construção de *rosters*, há restrições trabalhistas e operacionais de uma dada empresa de transporte que afetam a forma com que a alocação pode ser feita.

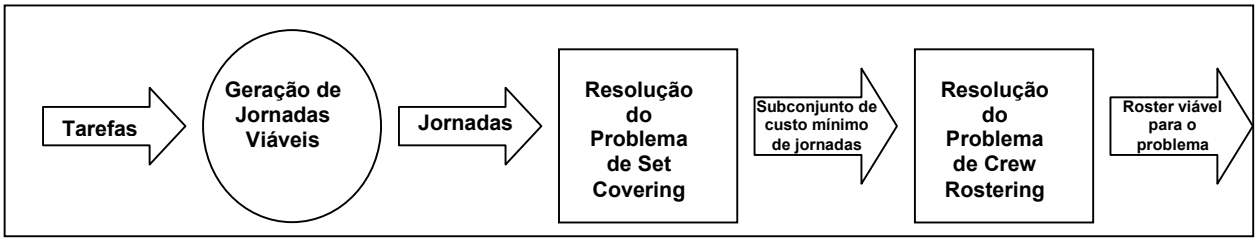

Figura 1: Resumo Esquemático das Fases do Problema de *Crew Scheduling*.

A figura 1sumariza todo o processo desde sua etapa inicial até a obtenção do resultado desejado.

Esse artigo tem sua atenção focadas nas partes (b) e (c) sumariamente descritas acima. Assim, uma primeira parte do mesmo apresentará a descrição, modelagem, algoritmos e testes das duas fases acima especificadas. Por fim, serão mostrados testes executados quando os dois algoritmos implementados – (b) e (c), são executados conjuntamente para produção de tabelas de escalonamento viáveis de trabalho.

2. Set Covering

O problema intitulado *Set Covering* ***(SCP)*** corresponde a um modelo essencialmente importante para diversas aplicações da Pesquisa Operacional. Dentre essas, inclui-se "*crew scheduling*" em que, dado um conjunto de tarefas a serem realizadas, as mesmas devem ser cobertas por um conjunto de mínimo custo de jornadas (seqüência de tarefas) que, por sua vez, será executada por uma única tripulação (conjunto de funcionários).

Algoritmos exatos para a solução de tal problema existem. Entretanto, quando as dimensões da matriz que o modela aumentam torna-se inviável recorrer aos mesmos devido ao grande intervalo de tempo naturalmente despendido para se encontrar tais soluções. Assim, recorre-se, não pouco freqüentemente, à heurísticas que objetivam, além de obter soluções de qualidade, encontrá-las em um período de tempo satisfatório para a aplicação em questão.

Esse seção pretende apresentar uma heurística que tem como base a *Relaxação Lagrangeana* e o *Método de Otimização Simplex* para resolver o problema de *Set Covering*. Em primeira instância será

mostrado o modelo utilizado para a solução do problema seguido do respectivo algoritmo e análise do desempenho alcançado por ele.

2.1. Modelagem

O problema de *Set Covering* ***(SCP)*** corresponde à cobertura de ***m***-linhas de uma matriz 0-1 $[a_{ij}]$ m x n por um subconjunto de ***n*** colunas de custo mínimo. Sendo:

- $M = \{1, 2, ..., m\}$, conjunto de linhas (tarefas) a serem cobertas.
- $N = \{1, 2, ..., n\}$, conjunto de colunas (jornadas) geradas.
- c_j ($j \in N$), o custo da coluna ***j*** quando a mesma encontra-se na solução do ***SCP***.
- $a_{ij} = 1$, se a coluna $j \in N$ cobre a linha i $\in M$.

Pode-se dizer que ***SCP*** procura por um subconjunto de mínimo custo $S \subseteq N$ de colunas de tal forma que cada linha $i \in M$ é coberta por, pelo menos, uma coluna $j \in S$. Um modelo matemático para o ***SCP*** pode ser descrito da seguinte forma:

$$f(SCP) = \min \sum_{j \in N} c_j x_j \tag{1}$$

Sujeito a:

$$\sum_{j \in N} a_{ij} x_j \geq 1 \qquad i \in M \tag{2}$$

$$x_j \in \{0,1\} \qquad j \in N \tag{3}$$

onde: $x_j = 1$ se $j \in S$, e $x_j = 0$ caso contrário.

Define-se também, como notação, para cada linha $i \in M$:

$$J_i = \{j \in N: a_{ij} = 1\} \text{ como o conjunto de colunas que cobrem a linha } i. \tag{4}$$

Analogamente, para cada coluna $j \in N$:

$$I_j = \{i \in M: a_{ij}=1\} \text{ como o conjunto de linhas que cobrem a coluna } j. \tag{5}$$

O primeiro conjunto de restrições, especificado em (2), garante que cada linha é coberta por, pelo menos, uma coluna e o segundo (3) impõem a integralidade das variáveis.

Aplicando-se a Relaxação Lagrangeana para o problema cuja função objetivo é (1) sujeito às restrições (2) e (3) (maiores detalhes sobre Relaxação Lagrangeana em [7]) obtém-se:

$$L(u) = \min_{x_j \in (0,1)} \sum_{j \in N} c_j(u)\, x_j + \sum_{i \in M} u_i \tag{6}$$

$$c_j(u) = c_j - \sum_{i \in I_j} u_i \tag{7}$$

onde (7) é o custo lagrangeano associado a cada coluna $j \in N$.

A Determinação de Multiplicadores Lagrangeanos

Para se determinar vetores lagrangeanos próximos do ótimo dentro de um pequeno intervalo de tempo uma boa aproximação consiste em utilizar um vetor subgradiente $s(u) \in R^m$ associado a um dado u definido da seguinte forma:

$$s_i(u) = 1 - \sum_{j \in J_i} x_j(u) \quad , \quad i \in M \tag{8}$$

Com essa aproximação é possível gerar uma seqüência de vetores u^0, u^1, u^2 ... não negativos onde u^0 pode ser arbitrariamente definido e o restante u_k^i, $k \geq 1$ pode ser determinado por meio da seguinte seqüência:

$$u_i^{k+1} = \max\left\{ u_i^k + \lambda \frac{UB - L(u^k)}{\left\| s(u^k) \right\|^2} s_i(u^k), \quad 0 \right\}, \quad i \in M \tag{9}$$

onde: ***UB*** é um limite superior para (1) e $\lambda \geq 0$ corresponde a um salto na direção de crescimento da função (vetor subgradiente).

2.3. Algoritmo – Descrição e Detalhamento

Seja *P1* o problema o problema definido pela função objetivo (1) e as restrições (2) e (3) e *P2*, o problema caracterizado pela função objetivo (6). Assim, temos que, tendo sido relaxada a restrição (2) de *P1*, a solução obtida por meio de *P2* corresponde a um limite inferior que, nem sempre, é viável (raramente atende-se à condição de viabilidade). Logo, é necessária a aplicação de uma heurística que busque viabilizar a solução obtida.

Uma abordagem bastante utilizada parte da solução *S** obtida em *P2* e, gradativamente, adiciona colunas à *S** até que todas as restrições violadas sejam atendidas. Por fim, colunas

de uma heurísti Log,18losaior pa(a o problem)78.5(ação)]TJ/TT14 1 TJ139Tf1 0 TD0.4017 Tc3.81719 TwSet
P2

possívãage

P2

Cada coluna $j \in N$, contém seu conjunto próprio de atributos que podem tanto ser obtidos como dado de entrada do problema (custo real de uma coluna) quanto determinados no transcorrer da execução do algoritmo (custo reduzido lagrangeano; determinado após a execução do Método de Otimização Subgradiente, custo reduzido do simplex; setado ou atualizado a cada vez que uma coluna participa do subconjunto de colunas para a qual se deseja obter uma solução viável – ***Problema Core – PC***).

O algoritmo aqui apresentado, após determinar um vetor de multiplicadores próximos do ótimo por meio do Método de Subgradiente, seleciona colunas de ***R*** e reaproveita outras de iterações anteriores com o objetivo de compor um conjunto de colunas a fazer parte do ***PC***.

Essa seleção é efetuada escolhendo-se no máximo ***t*** colunas de ***R*** e reaproveitando-se ***k*** dentre aquelas que já participaram, pelo menos uma vez, do ***PC***. Assim, sendo ***nc*** uma constante definida como o número de colunas a estarem presentes no ***PC*** a partir da segunda iteração do algoritmo, define-se também: ***número de iterações do algoritmo (iter)*** = *n div t*, *se n mod t = 0*; *(n div t) +1*, caso contrário. ***k*** = número de colunas a serem reaproveitadas = *nc – t*

Selecionadas as colunas a fazerem parte do ***Problema Core***, as mesmas são submetidas a um algoritmo que conjuga o método de otimização simplex e informações provenientes do vetor de multiplicadores próximo do ótimo calculado para o subproblema em questão, com a pretensão de obter uma solução viável para o mesmo. Sumariamente, o algoritmo pode ser divido em 3 partes:

FASE 1) Fase do Subgradiente: apresentando como entrada todas as colunas e linhas do problema, essa fase objetiva, além de encontrar um limite inferior global para o problema, obter um vetor de multiplicadores lagrangeanos próximo do ótimo que irá contribuir para a caracterização do quão importante cada uma das colunas é para o problema na Fase 2 do Algoritmo. Byun, em [11], apresenta uma nova abordagem para o cálculo do Limite Inferior baseado no Método de Subgradiente – "Subgradient bundle method" (*SBM*). Nessa abordagem, o cálculo de ***s(u)*** leva em consideração não apenas o vetor subgradiente da iteração corrente, como também aqueles determinados em iterações anteriores. Tal método, experimentalmente, mostrou-se tão eficiente quanto o clássico obtendo, em alguns casos, Limites Inferiores cujos valores superam aqueles obtidos pela abordagem clássica em até 4%.

FASE 2) Seleção de Colunas: Essa fase pretende, a cada iteração do algoritmo, determinar um novo conjunto de colunas a ser utilizado na Fase 3 do Algoritmo. Seja ***u**** o melhor vetor de multiplicadores lagrangeanos determinado na fase 1 e ***csimplex*** um vetor de custos reduzidos inicialmente setado em $+\infty$ para todas as colunas do problema e atualizado após a execução do método simplex. A partir de ***u**** é possível obter, para cada coluna $j \in N$, o seu respectivo custo reduzido ***cju(j)*** por meio de (7). Logo, para todo $j \in N$, obtém-se a seguinte fórmula que caracteriza o quão boa é cada coluna para a solução do problema: $c_{LS}(j) = cju(j) + csimplex(j)$. Assim, na primeira iteração do algoritmo, ***t*** colunas são retiradas do repositório ***R*** passando a compor o ***Problema Core*** e, obviamente, nenhuma é reaproveitada. Essas colunas serão submetidas à fase heurística, fase em que o valor de ***csimplex*** de todas as colunas pertencentes ao ***PC*** serão atualizados. Nas ***iter-1*** posteriores iterações, novamente, no máximo ***t*** colunas serão retiradas do repositório e irão compor o novo ***PC***. Entretanto, ***k*** colunas serão reaproveitadas e passarão, também, a compor o ***PC***, através da ordenação do vetor $\boldsymbol{c_{LS}}$ e posterior seleção das ***k*** primeiras melhores colunas dentre aquelas presentes no mesmo.

FASE 3) Fase Heurística: A Fase Heurística objetiva encontrar uma solução viável para o subproblema da iteração corrente do algoritmo. Para tal, a seguinte seqüência de passos é executada:

Execução do Método Simplex – esse método de otimização é executado tendo como entrada as colunas presentes no ***PC*** e *P1*. Ele, por sua vez, retorna uma solução **(*lp*)** que nem sempre apresenta variáveis cujos valores são inteiros. Logo, a partir da solução determinada pelo

Simplex, o procedimento *branch-and-bound* é executado objetivando encontrar uma solução cujas variáveis apresentam apenas valores inteiros (0 ou 1, no caso do *Set Covering*).
Sendo o *branch-and-bound* um procedimento de busca em árvore, a melhor solução procurada pode demandar um intervalo de tempo muito grande para ser encontrada. Logo, no presente algoritmo, o número de iterações para o qual esse procedimento tentará obter a solução ótima do ***PC*** foi limitada em Δ.

Heurística Lagrangeana – se após Δ iterações do procedimento *branch-and-bound* a melhor solução inteira encontrada por esse método exceder ***lp*** em mais de três unidades, a fase 1 é novamente executada (nessa situação, apenas as colunas pertencentes ao ***PC*** serão levadas em consideração) objetivando a obtenção de um vetor multiplicadores lagrangeanos capaz de caracterizar a importância das colunas pertencentes ao subproblema ***PC***. Esse vetor, por sua vez, é passado como parâmetro a um procedimento guloso com o objetivo de obter soluções de melhor qualidade. O procedimento implementado utiliza a metodologia proposta por Caprara e outros em [3]. No entanto, a função *Score* por ele definida, responsável por calcular a importância de cada uma das colunas para o subproblema considerado no presente algoritmo, leva também em consideração os custos reduzidos simplex anteriormente calculados.

As fases 2 e 3 são iteradas até que não mais haja colunas a serem buscadas do repositório.

Observações adicionais sobre o algoritmo

Como trabalha-se sempre com um subconjunto pequeno de colunas, testes experimentais mostraram ser a ***Fase Heurística*** bastante eficiente tanto computacionalmente quanto na tentativa de ser obter melhores soluções. Limitando o *branch-and-bound* a executar, no máximo ***4*m*** iterações, nota-se que o procedimento guloso, passa a ser executado com maior freqüência quando, aproximadamente, 60% das colunas do repositório já foram inseridas. Em especial, ele mostrou-se extremamente eficiente quando a sua execução foi requerida havendo, freqüentemente, melhoras na melhor solução disponível. A fase ***Seleciona Colunas***, por sua vez, embora simples, mostrou-se eficaz no propósito dado a ela: selecionar colunas do repositório e reaproveitar outras de iterações anteriores. Observações na evolução do valor do limite superior após findada cada uma das iterações mostra que esses valores são sempre decrescentes ou estáveis. Isso acontece porque colunas que compõem a melhor solução do problema para uma dada iteração, apresentam baixo custo reduzido tendendo, portanto, a permanecer no conjunto de colunas que compõem o ***PC***.

2.4. Resultados computacionais para o problema de Set Covering

Nesta seção serão apresentados os resultados computacionais obtidos ao submeter instâncias reais[1] e da literatura[2] ao algoritmo anteriormente apresentado. Além disso, os resultados obtidos são também comparados com aqueles obtidos pelos algoritmos *BeCh* de Beasley e Chu (1996), *Be* de Beasley (1990), *BaCa* de Balas e Carrera (1996), *CFT* de Caprara e outros e *Par* de M. Solar.

Tabela 1 mostra os resultados obtidos quando as instâncias da Classe 4, 5, 6, A, B, C e D de [13], para o qual a solução ótima é conhecida, são submetidas ao nosso algoritmo. *Nome*, *tamanho*, *custo*, *dens* (*densidade*) e *opt* (solução ótima) são as propriedades de cada uma das instâncias; *sol* corresponde à solução encontrada pelo algoritmo descrito nesse artigo e GAP é a diferença entre *sol* e *opt* em porcentagem.

[1] Instâncias reais provenientes da competição "FASTER Competition" ocorrida na Itália que comprende instâncias cujo número de linhas pode alcançar valor superior a 5000 e um número de colunas superior a 1000 000.
[2] OR Library - J.E. Beasley

Nome	Tamanho	Dens	Custo	Opt.	Sol.	Be	BaCa	BeCh	CFT	Par.	GAP
4,1	200 x1000	2%	1 - 100	429	429	429	429	429	429	432	0,00%
4,2	200 x1000	2%	1 - 100	512	512	512	512	512	512	517	0,00%
4,3	200 x1000	2%	1 - 100	516	516	516	516	516	516	x	0,00%
4,4	200 x1000	2%	1 - 100	494	494	495	494	494	494	513	0,00%
4,5	200 x1000	2%	1 - 100	512	512	512	512	512	512	x	0,00%
4,6	200 x1000	2%	1 - 100	560	560	561	560	560	560	x	0,00%
4,7	200 x1000	2%	1 - 100	430	430	430	430	430	430	x	0,00%
4,8	200 x1000	2%	1 - 100	492	492	493	492	492	492	x	0,00%
4,9	200 x1000	2%	1 - 100	641	641	641	641	641	641	x	0,00%
4.10	200 x 1000	2%	1 - 100	514	514	514	514	514	514	x	0,00%
5,1	200 x 2000	2%	1 - 100	253	253	255	254	253	253	271	0,00%
5,2	200 x 2000	2%	1 - 100	302	302	304	307	302	302	x	0,00%
5,3	200 x 2000	2%	1 - 100	226	226	226	226	228	226	x	0,00%
5,4	200 x 2000	2%	1 - 100	242	242	242	243	242	242	x	0,00%
5,5	200 x 2000	2%	1 - 100	211	211	211	211	211	211	x	0,00%
5,6	200 x 2000	2%	1 - 100	213	213	213	213	213	213	x	0,00%
5,7	200 x 2000	2%	1 - 100	293	293	294	293	293	293	x	0,00%
5,8	200 x 2000	2%	1 - 100	288	288	288	288	288	288	x	0,00%
5,9	200 x 2000	2%	1 - 100	279	279	279	279	279	279	x	0,00%
5.10	200 x 2000	2%	1 - 100	265	265	265	265	265	265	x	0,00%
6.1	200 x1000	5%	1 - 100	138	138	141	140	138	138	x	0,00%
6.2	200 x1000	5%	1 - 100	146	146	146	147	146	146	x	0,00%
6.3	200 x1000	5%	1 - 100	145	145	145	145	145	145	x	0,00%
6.4	200 x1000	5%	1 - 100	131	131	131	131	131	131	x	0,00%
6.5	200 x1000	5%	1 - 100	161	161	161	163	161	161	x	0,00%
A.1	300 x 3000	2%	1 - 100	253	253	255	258	253	253	258	0,00%
A.2	300 x 3000	2%	1 - 100	252	252	256	254	252	252	x	0,00%
A.3	300 x 3000	2%	1 - 100	232	232	234	237	232	232	x	0,00%
A.4	300 x 3000	2%	1 - 100	234	234	235	235	234	234	x	0,00%
A.5	300 x 3000	2%	1 - 100	236	236	237	236	236	236	x	0,00%
B.1	300 x 3000	5%	1 - 100	69	69	70	69	69	69	x	0,00%
B.2	300 x 3000	5%	1 - 100	76	76	77	76	76	76	x	0,00%
B.3	300 x 3000	5%	1 - 100	80	80	80	81	80	80	82	0,00%
B.4	300 x 3000	5%	1 - 100	79	79	80	79	79	79	x	0,00%
B.5	300 x 3000	5%	1 - 100	72	72	72	72	72	72	x	0,00%
C.1	400 x 4000	2%	1 - 100	227	227	230	230	227	227	x	0,00%
C.2	400 x 4000	2%	1 - 100	219	219	223	220	219	219	230	0,00%
C.3	400 x 4000	2%	1 - 100	243	243	252	248	243	243	x	0,00%
C.4	400 x 4000	2%	1 - 100	219	219	224	224	219	219	x	0,00%
C.5	400 x 4000	2%	1 - 100	215	215	217	217	215	215	x	0,00%
D.1	400 x 4000	5%	1 - 100	60	60	61	60	60	60	x	0,00%
D.2	400 x 4000	5%	1 - 100	66	66	68	68	66	66	x	0,00%
D.3	400 x 4000	5%	1 - 100	72	72	75	75	72	72	x	0,00%
D.4	400 x 4000	5%	1 - 100	62	62	64	63	62	62	65	0,00%
D.5	400 x 4000	5%	1 - 100	61	61	62	61	61	61	x	0,00%

Table 1: Results das instâncias das classes 4, 5, 6, A, B, C and D.

Em todas as instâncias apresentadas na tabela acima a melhor solução foi alcançada em um pequeno e aceitável intervalo de tempo. Essas instâncias foram executadas considerando-se um ***CP*** de aproximadamente 500 colunas e uma taxa de reaproveitamento de colunas de aproximadamente trinta porcento. Observações experimentais mostraram que o uso de ***CP*** e porcentagem de reaproveitamento pequenos resultam em uma maior eficiência do algoritmo. Além disso, experimentos com diferentes

Método Subgradiente Clássico x Método Bundle

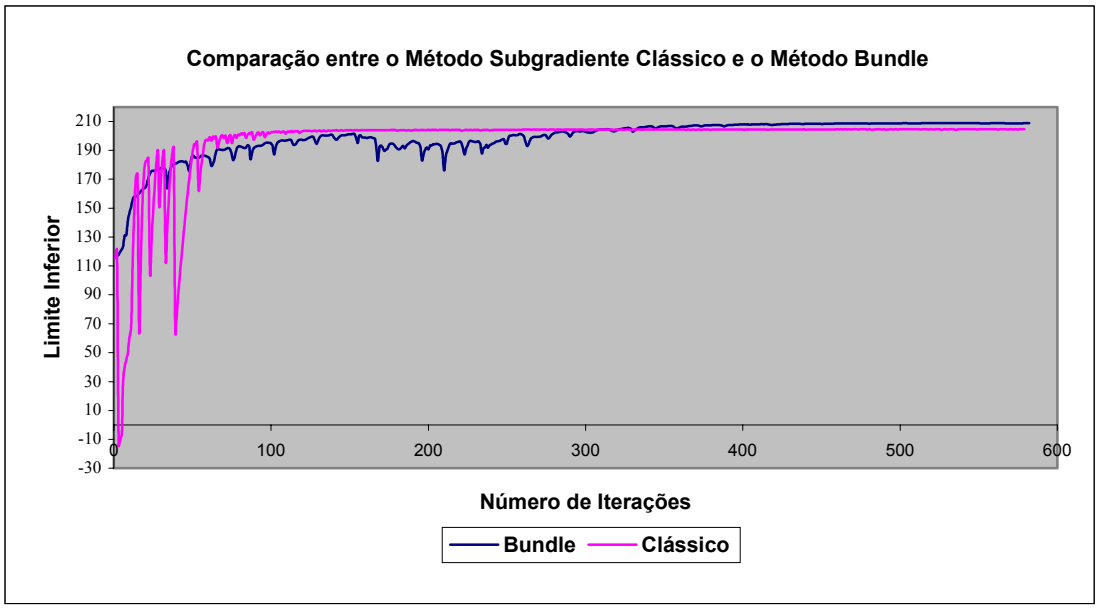

Figura 2 – Evolução dos Limites Inferiores segundo os métodos SBM e SCM.

A figura 2 mostra um gráfico que compara o métodos Clássico (*SBM*) e o Método Bundle (*SBM*) quando os mesmos são executados objetivando a obtenção de limites inferiores e multiplicadores lagrangeanos para o ***CP*** de uma determinada iteração. Ela mostra que o Método Clássico, além de sofrer de grande instabilidade nas iterações iniciais; o que pode resultar, em alguns casos, em uma lenta convergência do mesmo, tende a obter um limite inferior de pior qualidade. Entretanto, excetuando situações de anormalidade que tendem a degradar a performance do *SCM*, nota-se que, em poucas iterações; 150 para a instância analisada[3], o Limite Inferior encontra-se praticamente estabilizado sem haver comprometimento da qualidade dos multiplicadores lagrangeanos, o que ocorre apenas a partir da iteração de número 400 para o *SBM*. Logo, a escolha do método a ser implementado é uma questão de compromisso do algoritmo: ambos geram multiplicadores lagrangeanos capazes de caracterizar, com perspicácia, a importância de uma coluna quando a mesma é imposta na solução do problema ou subproblema a ser resolvido. No entanto, o *SBM* tende a gerar um limite inferior de melhor qualidade.

3. Crew Rostering

Recebendo como entrada as jornadas anteriormente obtidas por meio da solução do problema de *Set Covering*, o problema de *Crew Rostering* ***(CRP)*** objetiva a criação de *rosters* (seqüência de jornadas) possíveis de serem executados por uma única tripulação dentro de um determinado intervalo de tempo. Além disso, para a construção de *rosters*, há restrições trabalhistas e operacionais de uma dada empresa de transporte que afetam a forma com que a alocação pode ser feita. A descrição do problema, a especificação de atributos e regras trabalhistas são baseados em uma competição (FARO – Ferrovie Airo Rostering Optimisation) promovida pela AIRO (Italian Association of Operational Research) e Ferrovie dello Stato Spa (Italian State Railways, briefly FS) objetivando a resolução do ***CRP*** para companhias de redes ferroviárias. O algoritmo a ser apresentando, por sua vez, é baseado no trabalho de Caprara e outros com algumas modificações relatado em [6].

[3] Teste efetuado com a instância RAIL582 (582 x 55515) disponível em OR-BEASLEY. O método clássico obteve um limite inferior igual a 204 e o Método Bundle obteve 210.

3.1 – Descrição geral do problema

São dados como entrada um conjunto de jornadas que devem ser executadas segundo uma dada periodicidade. Cada uma dessas jornadas apresenta uma série de atributos: início, s_i, $0 \leqslant s_i < 1440$ (= 24 horas); fim, e_i, $0 \leqslant e_i < 1440$ (= 24 horas); duração, p_i; tempo de trabalho, $w_i \leqslant p_i$, tempo efetivo de trabalho de uma tripulação; tempo a ser pago, $a_i \geqslant w_i$, soma do tempo de trabalho e todos os outros possíveis intervalos de tempo adicionais. Uma jornada também pode classificada como: *jornada com descanso externo* – jornada que inclui descanso fora do depósito da tripulação; *jornada longa* – jornada que não apresenta descanso externo cujo tempo de trabalho (w_i) é maior que 8 horas e 5 minutos; *jornada noturna* – jornada cujo período de trabalho encontra-se no intervalo entre as 0 e 5 horas da manhã; *jornada noturna pesada* – jornada noturna, sem descanso externo que requer mais de 1 hora e trinta minutos de trabalho entre as 0 e 5 horas da manhã[4].Além disso, um dia completo é chamado *vago* se nenhuma jornada ou parte de uma jornada é executada durante o mesmo. Caso contrário é ele chamado de *cheio*. Uma semana é, por convenção, constituída por um grupo de k dias consecutivos sem uma relação direta com os dias Domingo – Sábado de uma semana sendo o k-ésimo dia vago. Um roster consiste de um subconjunto de jornadas que são executadas no decorrer de uma ou mais semanas. O tamanho de um roster é definido em função de seu número de dias (um inteiro múltiplo de k). Tipicamente um limite superior q no tamanho do roster é imposto. Além disso, a seguinte constante é definida:

$$\alpha = k \,.\, 1440 = \text{número de minutos em uma semana} \qquad (10)$$

Obviamente, todo roster, para ser viável, deve incluir descansos semanais para as tripulações que o executa. Assim, além do k-ésimo dia já anteriormente definido como vago, caracteriza-se também dois outros descansos: *simples* e *duplos*. Os mesmos diferenciam-se pelo seu tamanho mínimo. Descansos duplos são, geralmente, maiores que os simples (esse tamanho está condicionado à parâmetros a serem seguidos por cada empresa em particular). Se duas jornadas não são seqüenciadas por meio de descansos semanais simples ou duplos é dito que as mesmas são diretamente intercaladas. Para a construção de um roster viável restrições devem ser obedecidas. As restrições existentes nessa aplicação podem ser divididas em duas categorias: ***regras seqüenciais*** – são regras gerais aplicáveis entre o fim de uma jornada e o início de uma subseqüente. Estão elas relacionadas ao tempo mínimo que deve ser considerado ao se intercalar jornadas diretamente ou por meio de descansos simples ou duplos; ***regras operacionais*** – são regras impostas a uma semana ou, de uma maneira mais geral, a um roster, condicionadas à configuração de jornadas até aquele momento já seqüenciadas (regras condicionadas ao estado de um roster).

3.2 – Modelo para o Problema de Crew Rostering (CRP)

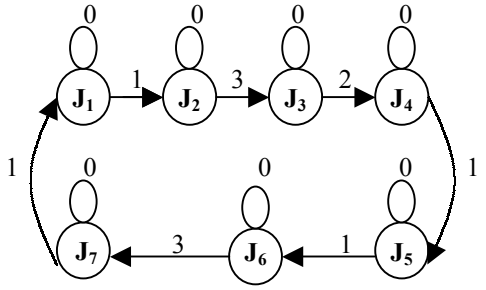

Figura 3: Um grafo correspondente a um roster viável.

[4] Os valores apresentados para caracterização de cada classe de atributos de jornadas variam entre implementações. Logo é sugerido que sejam eles parametrizados.

A figura 3 mostra um grafo *G* que corresponde a um roster viável construído respeitando-se as regras seqüenciais e operacionais acima descritas. Nesse grafo G = (V, A U L), onde V = {1, ..., n} é o conjunto de vértices, *A* é um conjunto de arcos e *L* é um conjunto de loops. A cada vértice uma jornada está associada. Os arcos que saem de um vértice *i* e entram em um vértice *j* correspondem ao tipo de seqüenciamento (direto; jornadas diretamente intercaladas, simples; jornadas intercaladas por meio de descansos simples ou duplo; jornadas intercaladas através de descansos duplos) aplicados ao se efetuar uma intercalação entre jornadas. Ou seja: o conjunto *A* pode ser subdividido em outros três disjuntos – A_1, A_2 e A_3. Arcos pertencentes a A_1 são chamados arcos diretos, arcos pertencentes a A_2 e A_3 são definidos, respectivamente, como arcos de descanso simples e arcos de descanso duplo. Para cada par de vértices $i, j \in V$ há um arco direto $(i, j) \in A_1$, cujo custo c^1_{ij} é o mínimo tempo, em minutos, entre o começo da jornada *i* e o início da jornada *j* quando essas são diretamente seqüenciadas. Em outras palavras, $c^1_{ij} = (s_j + h \,.\, 1440) - s_i$, onde *h* é o número mínimo de dias completos que resulta em um seqüenciamento direto. Raciocínio semelhante pode ser aplicado a arcos $(i, j) \in A_2$ e A_3.

Matrizes c_1, c_2 e c_3 podem ser calculadas a partir dos dados de entrada de acordo com as regras seqüenciais. Além disso, para toda jornada *i* é também definido $c^1_{ii} = c^2_{ii} = +\infty$ e $c^3_{ii} = \alpha$ (o que possibilita a criação de um roster que apresenta uma única jornada). O conjunto de loops *L* corresponde ao número de dias vagos a serem inseridos entre jornadas de modo que as regras operacionais impostas sejam cumpridas. Esse conjunto é particionado em $\delta + 1$ subconjuntos, L_0, L_1, ... L_δ; onde δ é um limite superior para o número de dias vagos entre duas jornadas. Assim, para cada vértice $i \in V$ e para $t = 0, ..., \delta$; tem-se um loop $(i,i) \in L_t$, cujo tamanho $d^t_i = t \,.\, 1440$ representa um descanso de *t* dias vagos entre *i* e a jornada subseqüente no roster.

Abaixo é dado um *Modelo de Programação Linear* para o ***CRP*** baseado nas formulações acima descritas para o grafo teórico apresentado. Nesse modelo as regras seqüenciais são implicitamente impostas por meio das matrizes c_1, c_2 e c_3. As regras operacionais, por sua vez, são modeladas via desigualdades que previnem contra a existência de soluções inviáveis.

Para cada arco $(i, j) \in A_t$, i, j = 1, ..., n, t = 1, 2, 3, é introduzida uma variável binária x^t_{ij} igual a 1 se o arco $(i, j) \in A_t$ está na solução, e 0 caso contrário. Similarmente, para cada loop $(i, i) \in L_t$, i = 1, ..., n, t = 0, ..., δ, é introduzida uma variável binária y^t_i igual a 1 se o loop $(i, i) \in L_t$ está na solução ótima, e 0 caso contrário. Finalmente, seja o conjunto $\boldsymbol{P}$ definido como o conjunto mínimo de soluções inviáveis:

$$v(CRP) = \min \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{t=1}^{n} c^t_{ij} x^t_{ij} + \sum_{i=1}^{n}\sum_{t=0}^{\delta} d^t_i y^t_i \tag{11}$$

$$\sum_{i=1}^{n}\sum_{t=1}^{3} x^t_{ij} = 1, \qquad j = 1,...,n \tag{12}$$

$$\sum_{i=1}^{n}\sum_{t=1}^{3} x^t_{ij} = 1, \qquad i = 1,...,n \tag{13}$$

$$\sum_{t=0}^{\delta} y^t_i = 1, \qquad i = 1,...,n \tag{14}$$

$$\sum_{t=1}^{3}\sum_{(i,j)\in A_t \cap P}\left(1 - x^t_{ij}\right) + \sum_{t=0}^{\delta}\sum_{(i,i)\in L_t \cap P}\left(1 - y^t_i\right) \geq 1, \qquad P \in \mathrm{P} \tag{15}$$

$$x^t_{ij} \in \{0,1\}, \qquad i, j = 1,...,n;\ t = 1, 2, 3 \tag{16}$$

$$y^t_{ij} \in \{0,1\}, \qquad \mathrm{i, j} = 1,...,\mathrm{n};\ \mathrm{t} = 0..\delta \tag{17}$$

Restrições (12) e (13) impõem que cada nodo apresenta, respectivamente, apenas um arco entrando e um arco saindo do mesmo. A restrição (14) garante que a solução apresenta, exatamente, um loop incidente em cada vértice. A inexistência de soluções inviáveis é garantida por meio da restrição (15). Essa restrição estipula que pelo menos uma variável associada aos arcos / loops de $\boldsymbol{P}$ deve ser setada em 0.

O modelo acima será estendido por meio da inserção de variáveis e desigualdades que são redundantes para o modelo já apresentado (a restrição 15 engloba toda e qualquer outra que, porventura, resulte em uma solução inviável), mas torna-se útil para a relaxação do ***CRP***. Essas novas variáveis e desigualdades impõe que o total de descansos semanais deve ser, no mínimo, igual ao número de semanas que compõem um roster e que o total de descansos duplos deve ser, no mínimo, igual a β vezes o número de descansos semanais. Essas restrições são modeladas através da introdução de duas variáveis inteiras: *w*, representando o número mínimo de arcos simples ou duplos na solução e *z*, representando o número mínimo de arcos duplos.

$$w \geq \frac{\sum_{i=1}^{n}\sum_{i=1}^{n}\sum_{t=1}^{3} c_{ij}^{t} x_{ij}^{t} + \sum_{i=1}^{n}\sum_{t=0}^{\delta} d_{i}^{t} y_{i}^{t}}{\alpha} \tag{18}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\left(x_{ij}^{2} + x_{ij}^{3}\right) \geq w \tag{19}$$

$$z \geq \beta . w \tag{20}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij}^{3} \geq z \tag{21}$$

$$w, z \geq 0 \qquad \textit{inteiro.} \tag{22}$$

3.3 – Limite Inferior

Caracterizamos aqui os limites inferiores que podem ser obtidos através das propriedades e formulações do ***CRP*** de modo a se obter uma estimativa da qualidade das soluções viáveis obtidas.

a) ***limites triviais***: podem ser obtidos pela simples imposição de um limite superior no total de jornadas com um dado atributo *j* ou no tempo total de trabalho (w_i) ou a ser pago (a_i) em uma semana ou em um conjunto cíclico de *m* dias em um roster. Para a determinação dos limites inferiores triviais foram considerados cinco atributos que podem vir a caracterizar uma jornada: *tempo a ser pago*, *jornada longa*, *jornada noturna*, *jornada com descanso externo* e *tempo de trabalho*. A cada um desses atributos está, respectivamente, associada uma expressão para cálculo do limite inferior:

- $L_1 = \alpha . 5 . \sum_{i=1}^{n} \frac{a_i}{170.60}$;
- $L_2 = \alpha$. número de *jornadas longas*;
- $L_3 = \alpha$. (número de *jornadas noturnas*) / 2;
- $L_4 = \alpha$. 5 . (número de *jornadas com descanso externo*) / 7;
- $L_5 = \alpha \ . \ \sum_{i=1}^{n} \frac{w_i}{36.60}$

Os limites inferiores 1, 4 e 5 são calculados baseados nos atributos das *n* jornadas fornecidas como dado de entrada para o problema. Os limites 2 e 3, por sua vez, são recalculados a cada término de semana em um roster.

b) ***limites inferiores obtidos através da Relaxação Lagrangeana:*** utilizando-se técnicas de Relaxação Lagrangeana [7] obtém-se o seguinte problema:

$$\min \quad \lambda_1 w + \lambda_2 z + \sum_{t=1}^{3}\sum_{i=1}^{n}\sum_{j=1}^{n} \tilde{c}_{ij}^{t} x_{ij}^{t} \tag{23}$$

$$w \geq \frac{\lambda_1 w + \lambda_2 z + \sum_{t=1}^{3}\sum_{i=1}^{n}\sum_{j=1}^{n} \tilde{c}_{ij}^{1} x_{ij}^{t}}{\alpha} \tag{24}$$

$$\alpha.w \geq \lambda 1 w + \lambda 2 \lceil \beta.w \rceil + v(AP), \quad w \geq 0 \ \ inteiro; \tag{25}$$

Onde os valores $\tilde{c}_{ij}^{1} = c_{ij}^{1}, \quad \tilde{c}_{ij}^{2} = c_{ij}^{2} - \lambda_1, \quad \tilde{c}_{ij}^{3} = c_{ij}^{3} - \lambda_1 - \lambda_2$; são os custos lagrangeanos associados, respectivamente, às variáveis x_{ij}^{1}, x_{ij}^{2} *e* x_{ij}^{3} e λ_1 e λ_2 são multiplicadores lagrangeanos que, para a modelagem apresenta pode assumir dois conjuntos de valores: (λ_1^* , λ_2^*) = (1440, 1440) ou (λ_1^* , λ_2^*) = (1440, 0). A partir dessas restrições define-se $z = \lceil \beta.w \rceil$. Além disso, identifica-se $\sum_{t=1}^{3}\sum_{i=1}^{n}\sum_{j=1}^{n} \tilde{c}_{ij}^{t} x_{ij}^{t}$ como *v(AP) – Problema de Assinalamento*. Maiores detalhes em [8].

3.4 – O Algoritmo e a Heurística

Nesta seção é descrita uma heurística para ***CRP*** que faz uso das informações obtidas do problema relaxado já descrito baseado no trabalho de Caprara e outros descrito em [6]. Ela constrói um *roster* por vez escolhendo, consecutivamente, jornadas a serem inseridas no mesmo segundo um dado critério. Uma vez que o *roster* tenha sido completado, todas as jornadas que o compõe são removidas do problema e o processo é iterado para aquelas remanescentes até que não mais haja jornadas a serem seqüenciadas.

Seja *(u, v)* vetores que representam uma solução dual ótima para o *PA* correspondente ao melhor limite inferior *LB*. Nesses vetores, v_i é a variável dual ótima associada à i-ésima restrição do tipo (3) e u_i representa a variável dual ótima associada à i-ésima restrição do tipo (4), para i = 1, ..., n. Para cada par *(i, j)* $\in V$ e para t = 1, 2, 3, o custo reduzido do arco (i, j) $\in A_t$ é $\tilde{c}_{ij}^{t} = \tilde{c}_{ij}^{t} - u_i - v_j \geq 0$. Esse valor representa um limite inferior para o crescimento da função *v(PA)* quando o arco (i, j) $\in A_t$ é imposto na solução.

Experimentalmente, a utilização de $\tilde{c}_{ij}^{t}$ ao invés de c_{ij}^{t} culmina em resultados de qualidade superior. Logo, a construção de uma coleção de *rosters* que utilizam apenas arcos cujo custo reduzido apresenta o valor zero é desejada.

Descrição geral da Heurística

Inicia-se o roster selecionado-se sua jornada inicial (i_0). Uma vez selecionada, uma seqüência de iterações é executada:

a) a melhor jornada a ser seqüenciada após a última já inserida no *roster* é escolhida;
b) o limite inferior *LB* e os limites triviais são parametricamente atualizados;
c) a possibilidade do roster corrente ser fechado é considerada. Esse procedimento é iterado até que não mais exista *rosters* melhores que o corrente parando de qualquer forma se o seu tamanho atinge o limite superior estipulado (q = 60).

O procedimento de construção de um *roster* termina quando uma das duas condições especificadas em ***(c)*** é atingida. Assim, o roster corrente é adicionado à solução geral do problema ***CRP*** e um novo é construído se há jornadas ainda não cobertas.

Detalhamento da Heurística

Escolha da jornada inicial - A jornada inicial (i_0) é escolhida como a jornada *i* que apresenta o melhor *score* que leva em consideração o número de arcos com custo reduzido zero que incidem no vértice *i*. Em particular, como essa jornada será escalonada, geralmente, no primeiro dia do *roster* dá-se prioridade:

a) a jornadas tendo um número pequeno de arcos incidentes em seu vértice pertencentes a A_1;
b) a jornadas que apresentam um grande número de arcos incidentes em seu vértice que pertencem a A_2 e A_3.

Além disso, como i_0 deverá ser seguida por outras na mesma semana, jornadas que não apresentam arcos diretos que saem de seu vértice com custo reduzido zero são fortemente penalizadas.

Escolha da próxima jornada - A jornada *j* a ser seqüenciada após a corrente *i* é escolhida da seguinte forma:

1. Para cada jornada candidata *g* é considerado o seqüencimento de *g* após *i* através de 3 possíveis movimentos: movimentos diretos (μ_g^1), movimentos de descanso simples (μ_g^2) e movimentos de descanso duplo (μ_g^3). Esses correspondem, respectivamente a arcos (i, g) pertencentes a A_1, A_2 e A_3.
2. Ao movimento μ_g^l (l = 1, 2, 3) é atribuído um *score* ∂_g^l que leva em conta o quanto o limite inferior global *LB** vai crescer (θ_g^l) quando o arco (i, g) $\in$ A_1 é imposto na solução. Esse valor, por sua vez, é avaliado como a soma entre c_{ig}^l e o número de dias vagos que deve ser inserido entre *i* e *g* para o movimento ser viável[5].
3. Além disso, buscando atender uma das regras operacionais pré-estabelecidas, *score* ∂_g^2 é penalizado se o número de descansos duplos já impostos no *roster* é menor que β vezes o número corrente de semanas (possivelmente incompletas) já inseridas no mesmo. Da mesma forma, *score* ∂_g^3 é penalizado se o número de descansos duplos é β vezes maior que o número de semanas (possivelmente incompletas) já inseridas no *roster*.
4. Termos adicionais são inseridos nos *scores* das jornadas objetivando "quebrar" possíveis empates. Os mesmos levam em consideração tanto o número de arcos entrando e saindo de um nodo *g* quanto os atributos de cada jornada. O compromisso estabelecido é o de seqüenciar, em primeira instância, jornadas que são críticas (Ex.: jornadas com um número pequeno de arcos incidentes e que saem de seu nodo com custo reduzido zero, jornadas noturnas, pesadas, com descanso externo e com tempo de trabalho e a ser pago grandes). A importância de cada termo desse *score* (peso) é avaliado como uma função entre o quociente do correspondente limite trivial e o limite inferior global *LB**.
5. Assim, para cada jornada *g*, é assinalado um *score* $\partial_g = \min\left\{\partial_g^1, \partial_g^2, \partial_g^3\right\}$ e, a jornada *j* a ser seqüenciada após a *i* é escolhida como aquela que apresenta *mínimo score*.

Atualização do Limite Inferior – Após a inserção de cada jornada em um *roster*, é necessário efetuar o cálculo do novo limite inferior global *LB**. Tal pode ser feito da seguinte maneira:

- Para cada par de jornadas (i, j) já seqüenciadas, o arco correspondente é imposto na solução através do assinalamento de + ∞ a todo $k \in \{1, ..., n\}$ e $k \neq j$.
- A partir da inicialização acima descrita, o novo *PA* é resolvido parametricamente e LB é calculado como em (14).

[5] A viabilidade de um movimento está condicionada ao respeito de regras seqüenciais e operacionais impostas para o problema.

Fechamento de um Roster – Dada uma jornada *j* já selecionada para compor um determinado *roster*, é considerada a possibilidade de fechamento de um ciclo de trabalho precedendo *j* por meio de um movimento $c^t_{ji_0}$ (t = 1, 2, 3). Para tal, é necessário que haja a checagem da viabilidade de se executar tal operação. Essa, por sua vez, pode ser feita através de testes que verifiquem se as regras operacionais pré-estabelecidas (ver seção 1.4) são respeitadas quando o *roster* é fechado.

Havendo a possibilidade de se executar o fechamento de um roster por meio de mais de um movimento, deve ser selecionado aquele cuja soma $c^t_{ji_0} + \kappa$ seja a menor possível.[6]

3.5. Resultados para o problema de Crew Rostering

O algoritmo descrito na seção anterior foi testado utilizando instâncias reais providas pela Ferrovie dello Stato SpA e aplicadas na competição FARO e instâncias artificiais obtidas por meio da combinação de instâncias reais.

Name	*n*	*ext. rest*	*long*	*overnight*	*heavy*	*p*	*w*	*a*
F021	21	5	3	7	2	575	446	575
F033	33	6	3	14	8	525	437	525
F069	69	13	2	18	2	507	411	507
F134	134	24	6	50	23	510	420	514
F164	164	34	13	59	27	514	409	522
F360	360	92	52	136	49	587	456	610
F386	386	100	22	171	66	560	429	576
F525	525	149	29	240	89	579	439	595

Table 3: Característica das instâncias submetidas ao algoritmo

Tabela 3 ilustra as características das instâncias reais provenientes da competição FARO. Para cada instância a tabela mostra:

name	Nome da instância;
N	Número de jornadas;
Jornada externa	Número de jornadas com descanso externo;
Jornada longa	Número de jornadas longas;
Jornada Noturna	Número de jornadas noturnas;
Jornada Pesada	Número de jornadas noturnas pesadas;
P	Tempo médio da duração das *N* jornadas (em minutos);
W	Tempo de trabalho das *N* jornadas(em minutos);
A	Tempo médio pago pelas *N* jornadas (em minutos);

			A
21*	7	8	12.500
33*	10	14	28.571
54*	18	23	21.739
69*	24	28	14.286
90	30	35	14.286
102	34	40	15.000
134*	42	54	22.222
155	48	59	18.644
164*	53	64	17.188
167	54	66	18.182
197	65	80	18.750
203	68	81	16.049

[6] κ caracteriza o número de dias vagos a serem inseridos entre *j* e i_0 para garantir a viabilidade do roster.

233	69	76	9.211
298	97	117	17.094
360*	118	152	22.368
385	125	159	21.384
386*	118	184	35.870
393	131	165	20.606
407	133	195	31.795
419	136	200	32.000
429	143	177	19.209
455	147	210	30.000
494	165	207	20.290
520	170	243	30.041
524	170	211	19.431
525*	165	253	34.783
546	172	261	34.100
550	182	255	28.627
558	177	267	33.708
594	190	279	31.900
659	211	312	32.372
689	220	320	31.250
746	247	340	27.353
911	288	434	33.641

Tabela 4: Resultados para o problema de Crew Rostering

A tabela acima mostra os resultados obtidos pelo algoritmo descrito para resolução do problema de Crew Rostering. Para cada uma das instâncias a tabela reporta o número de jornadas a ser seqüenciada, o limite inferior (LB) e o limite superior (UB) encontrados em semanas e o GAP entre o limite inferior e o superior obtido por meio da seguinte fórmula: GAP = (UB-LB)/UB.

A análise dos resultados acima apresentados mostra soluções com ***GAP*** variando entre 9.2 e 35.8 %. Ou seja: em nenhum caso chegou-se a uma solução ótima. Pode-se dizer que o resultado apresentado pelo algoritmo implementado não foi satisfatório em virtude do ***GAP*** alto obtido em quase todas as instâncias submetidas ao algoritmo. Entretanto, melhores resultados podem ser obtidos caso outras políticas, que exploram e incluem maiores detalhes a serem considerados na construção da heurística, sejam adotadas:

- A determinação da próxima jornada a ser seqüenciada em um *roster* pode levar em consideração técnicas de ***look ahead (k)*** onde *k* é o número de passos adiante a serem levados em consideração na obtenção da melhor jornada. Ou seja: para cada jornadas *j* ainda não seqüenciada, além do *score* a ser calculado para a mesma, determina-se o melhor *score* da jornada a ser seqüenciada após *j* caso essa fosse escolhida para compor o *roster*. Indubitavelmente, esse procedimento demandaria muito tempo para instâncias cujo número de jornadas a serem seqüenciadas é alto. Logo, poderia ser aplicada essa técnica quando um o número de jornadas ainda não seqüenciadas em *roster* atingir um determinado valor.
- Estipulação de restrições não formais que priorizem o assinalamento de descansos simples e duplos coincidentes com o dia vago garantido a qualquer tripulação em uma semana.
- O processo de construção de um roster, ao invés de ser aplicado uma única vez, poderia ser aplicado múltiplas vezes até que um determinado intervalo de tempo seja alcançado ou quando a solução ótima fosse obtida.
- Finalmente, a possibilidade de utilização de Programação por Restrições é considerada na modelagem das restrições operacionais.

* Instâncias reais.

4. Resultados obtidos quando Set Covering e Crew Rostering são executados consecutivamente

Na introdução desse artigo a abordagem utilizada para se resolver o problema de *Crew Scheduling* foi especificada: dividir o problema em partes de modo que a saída de um servisse como dado de entrada para o outro. Assim, , e foram consecutivamente executados. O resultado final é um conjunto de rosters que serão assinalados a uma única tripulação.

Abaixo é mostrado o resultado obtido quando quatro instâncias passaram pelo processo acima descrito. Nessa tabela ***Dimensão*** representa o tamanho da matriz de restrições (m x n) fornecida como entrada para o problema de *Set Covering* (*m* é o número de tarefas e *n* o número de jornadas viáveis), ***LBSC*** e ***UBSC*** são, respectivamente, o limite inferior e superior obtidos para o problema ***SCP***. O problema anterior seleciona um número ***n*** de jornadas de mínimo custo a serem submetidas ao problema de ***Crew Rostering*** (***NJ*** na tabela abaixo). Da mesma forma, ***LBCR*** e ***UBCR*** correspondem, respectivamente, ao limite inferior e superior para o problema de ***Crew Rostering*** em semanas; ***GAP*** é calculado segundo fórmulas apresentadas em cada um dos problemas na seções anteriores e ***UBB*** é obtido somando-se os ***GAP'S*** dos problemas de *Set Covering* e *Crew Rostering*. Esse valor representa o quão afastado os algoritmos propostos, quando considerados em conjunto, estão da solução ótima no pior caso.

Set Covering				*Crew Rostering*				
Dimensão	**LBSC**	**UBSC**	**GAP (%)**	**NJ**	***LBCR***	**UBCR**	**GAP(%)**	**UBB**
200 x 1000	521	521	0	61	19	23	21.0526	21.0526
200 x 1000	560	560	0	65	25	33	32	32
200 x 2000	253	253	0	63	22	26	18.1818	18.1818
400 x 4000	227	227	0	82	28	31	10.7142	10.7142
516 x 47311	182	182	0	158	51	64	25.4902	25.4902

Tabela 5: Resultados obtidos executando o Set Covering e o Crew Rostering um após o outro.

Conclusão: Nesse artigo foram apresentados algoritmos que objetivavam resolver um problema bastante comum originado em sistema de trânsito de massa (ônibus, aviões e trens) – o *crew scheduling problem*. Nossa abordagem consistiu em dividi-lo em fases resolvendo cada uma delas em seqüência de modo que a saída de uma delas servisse como dado entrada para uma fase posterior. Para esse propósito revisamos as características e modelos dos dois principais algoritmos envolvidos nesse processo – *Set Covering* e *Crew Rostering*. Além disso, também foi proposto nesse artigo uma abordagem simples, mas eficiente para resolver o problema de *Set Covering* e a abordagem descrita em [6] foi utilizada objetivando-se resolver o problema de *Crew Rostering*. Também foram sucintamente citados algoritmos e métodos de otimização comumente utilizados na Literatura (Relaxação Lagrangeana, Método Simplex, Problema de Assinalamento, Método Subgradiente Clássico e Bundle). Por fim, foram mostrados os resultados computacionais dos problemas *Set Covering* e *Crew Rostering* quando os mesmos são resolvidos isoladamente e em conjunto. Além disso, também foi explicitada uma expectativa de GAP no pior caso quando os problemas são resolvidos em conjunto.

5. Bibliografia

[1] B.M. Smith, "A bus crew scheduling system using integer programming", *Mathematical Programming* 42 (1988) 181-187.

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

[3] A. Caprara, M. Fischeti, P. Toth, "A heuristic algorithm for the set covering problem", Relatório Técnico, DEIS, Universidade de Bologna, Outubro de 1995.

[4] S. Ceria, P. Nobili, A. Sassano, "A lagrangian-based heuristic for large-scale set covering problems", Relatório Técnico, IASI-CNR, Roma, Abril de 1995.

[5] REEVES, C.R. "Modern heuristic thechniques for combinatorial problems". John Wiley & Sons, Inc., 1993.

[6] A. Caprara, M. Fischeti, P. Toth, "Modeling and solving the crew rostering problem", Relatório Técnico, DEIS, Universidade de Bologna, Outubro de 1995.

[7] FISHER, M. L., "An applications oriented guide to lagrangian relaxation", *Interfaces* 15:2, March-April 1985, pg 10-21.

[8] Mateus, G. R., "Otimização em redes", Relatório Técnico, DCC, Universidade Federal de Minas Gerais, Julho de 1996.

[9] J. E. Beasley, P.C. Chu, "A genetic algorithmn for the set covering problem", *European Journal of Operation Research 9*4 (1996) 392-404.

[10] A.V. Eremeev, "A genetic algorithmn with a non-binary representation for the set covering problem", *OR-98* pg. 175-181.

[11] BYUN, Chul-Young, "Lower bound for large-scale set partitioning problems", Master-Thesis, Technischen Universität Berlin, Janeiro de 2001.

[12] M. Solar, V. Parada, R. Urrutia, "A parallel genetic algorithm to solve the set-covering problem", *Computers & Operation Research* 29 (2002) 1221-1235.

[13] OR-Library: distributing test problems by electronic mail. http://www.ms.ic.ac.uk/info.html