ALGORITMO BASEADO EM GRAFO MULTIPARTIDO PARA ESCALONAMENTO DE PESSOAL EM EMPRESA DE TRANSPORTE

Ademir Aparecido Constantino
Universidade Estadual de Maringá
ademir@din.uem.br
Rogério Calvi
Universidade Estadual de Maringá
calvi@din.uem.br
Silvio Alexandre de Araujo
Universidade Estadual Paulista
saraujo@ibilce.unesp.br
Cândido Ferreira Xavier de Mendonça Neto
Universidade de São Paulo
cfxavier@usp.br

RESUMO

O Problema de Escalonamento de Tripulação consiste em encontrar um conjunto de jornadas que cubra a escala diária de ônibus de uma empresa de transporte público minimizando os custos e satisfazendo um conjunto de leis trabalhistas e regulamentações da empresa. Neste trabalho é proposto um novo algoritmo heurístico de duas fases para resolução do problema de escalonamento de tripulação utilizando múltiplos problemas de designação que surge de um modelo baseado em grafo multipartido. A primeira fase do algoritmo constrói uma solução inicial. Na segunda fase o algoritmo tenta melhorar a solução encontrada na primeira fase modificando as matriz de custos do problema de designação. Nesta fase são propostos dois modelos diferentes. Testes computacionais são realizados usando dados do mundo real de duas empresas de transporte rodoviário urbano e instâncias aleatórias extraídas de dados reais. Os resultados obtidos são comparados com os resultados de um modelo de cobertura de conjunto e com limites inferiores calculados para as instâncias. O algoritmo apresentou bons resultado comparando a qualidade das soluções e em tempo computacional.

Palavras-chave: problema de escalonamento de tripulação, grafo multipartido, problema de designação.

ABSTRACT

The Crew Scheduling Problem consists in finding a set of duties which covers the bus daily schedule of a public transport company, minimizing costs and satisfying a set of work laws and company regulamentations. In this work it is proposed a new two phase heuristic algorithm to solve the crew scheduling problem using multiples assignment problems that arises from model based on multi-partite graph. In the first phase the algorithm constructs an initial solution. In the second phase the algorithm tries to improve the initial solution modifying the matrix of cost of the assignment problem. In this phase two different models are proposed. Computational tests are realized using real world data of two companies of urban bus and randomized instances extracted of real data. Results reached are compared with results from a resolution using the set covering model and with lower bonds calculated for instances. The algorithm reached good results comparing the quality solutions and computational time

Key-words: crew scheduling problem, multi-partite graph, assignment problem.

1 Problema de Escalonamento de Tripulação

O Problema de Escalonamento de Tripulação (PET), consiste em gerar uma escala de trabalho para a tripulação, ou seja, um conjunto de jornadas de trabalho de tal forma que toda uma escala de ônibus predefinida seja contemplada, minimizando, assim, os custos. Além disso, é necessário que as restrições legais estabelecidas em lei e contratos com o sindicato da categoria sejam obedecidos, bem como regulamentações da companhia, o que aumenta ainda mais a complexidade do problema.

Algumas definições básicas relacionadas ao PET, são: ***Bloco***: uma seqüência de viagens que um veículo tem de realizar em um dia, geralmente, começando e terminando na garagem. O conjunto de blocos forma a *escala de ônibus*; ***Oportunidade de troca***: é um local e horário onde são permitidos que uma tripulação abandone ou assuma um veículo; ***Tarefa***: a porção de trabalho entre duas oportunidades de trocas consecutivas. Portanto, durante a realização de uma tarefa não é possível que haja troca de tripulação. As tarefas são os dados de entrada para resolução do PET; ***Fração de trabalho:*** ou simplesmente ***fração***, contém um número de tarefas consecutivas; ***Jornada***: o trabalho diário a ser realizado por uma tripulação e compreende uma ou mais frações de trabalho; ***Tripulação***: geralmente envolve um motorista e um cobrador, mas em alguns casos pode referir-se a apenas um motorista;

Entre os modelos mais utilizados para resolução do problema de escalonamento de tripulação estão o Problema de Cobertura de Conjunto (PCC) e o Problema de Partição de Conjunto (PPC) que são expressos como problemas de programação linear inteira (PLI).

Sejam n o número de potenciais jornadas (colunas), m o número de frações de jornada (linhas) a serem cobertas e c_j o custo associado a jornada j. Considerando, ainda, as constantes inteiras a_{ij} assumindo valor 1 se a jornada j cobre a fração i e, caso contrário, o valor 0, então pode-se formular o seguinte modelo do PCC:

$$\text{Min} \quad \sum_{j=1}^{n} c_j x_j \tag{1}$$

$$\text{Sujeito a:} \quad \sum_{j=1}^{n} a_{ij} x_j \geqslant 1 \,,\; i \in \{1,2,\ldots,m\} \tag{2}$$

$$x_j \in \{0,1\},\; j \in \{1,2,\ldots,n\} \tag{3}$$

A função objetivo (1) minimiza o custo total. A restrição (2) garante que cada linha (fração de trabalho) seja coberta ao menos por uma coluna (jornada), e a restrição de integralidade (3) garante que somente jornadas completas serão consideradas.

Alterando-se a restrição (2), para uma restrição de igualdade, passamos a ter o modelo conhecido como Problema de Partição de Conjunto.

Os dois modelos são bastante semelhantes, exceto pela substituição da desigualdade no modelo de cobertura de conjunto pela igualdade no modelo de partição. Porém, o número total de possíveis jornadas (colunas da matriz) é usualmente muito grande. Isso deu origem a algumas técnicas heurísticas para reduzir o número de colunas (jornadas), ou, alternativamente, o problema tem sido decomposto em vários subproblemas e resolvidos separadamente. Tanto o PCC como o PPC são problemas NP-difíceis (KARP, 1975 e KARP, 1976 *apud* GOLDBARG e LUNA, 2000).

Para a resolução destes modelos, na literatura podem ser encontrados diversos trabalhos, tais como: Beasley (1987) que propôs um algoritmo exato para o PCC; Beasley e Chu (1996) apresentaram um algoritmo genético; Caprara, Fischetti e Toth (1999) propuseram uma heurística baseada em relaxação lagrangeana; Wren e Wren (1995) usaram um algoritmo genético com um novo operador de cruzamento para resolver o escalonamento diário de condutores; Mayerle

(1996) propôs um algoritmo genético; Fortsyth e Wren (1997) aplicaram a meta-heurística *ant system* para resolver o escalonamento de tripulação; Ohlsson, Peterson e Söderberg (2001) desenvolveram uma rede neural artificial com realimentação de campo médio para resolver o PCC; Li e Kwan (2000) apresentaram um algoritmo *evolução simulada fuzzy* (similar aos algoritmos genéticos); Mauri e Lorena (2004) descreveram um metodologia interativa baseada na aplicação do algoritmo de treinamento populacional (ATP) juntamente com programação linear (PL).

Apesar da maioria dos trabalhos encontrados na literatura tratarem de métodos que utilizam os modelos de cobertura ou partição de conjunto para resolver o problema de escalonamento de tripulação (PET), existem alguns trabalhos que utilizam métodos heurísticos para resolução do PET sem utilizar tais modelos. Shen e Kwan (2000) desenvolveram o algoritmo HACS que utiliza uma busca tabu para resolver o problema de escalonamento de tripulação; Souza, Cardoso e Silva (2003) abordaram o problema de escalonamento de tripulação utilizando as meta-heurísticas têmpera simulada, busca tabu e uma hibridização dessas duas técnicas; Siqueira (1999) resolve o PET através do problema de emparelhamento em grafo não bipartido, sendo que, as jornadas são formadas por, no máximo, duas frações.

2 Algoritmo Proposto

O algoritmo proposto baseia-se na resolução consecutiva de problemas de designação. O problema de designação é equivalente ao emparelhamento perfeito de custo mínimo em um grafo bipartido. Dada uma matriz de custos $[c_{ij}]$ de dimensão n x n, o problema de designação consiste em associar a cada linha uma coluna diferente de tal maneira que a soma dos custos seja mínima. Usando uma váriavel binária $x_{ij} = 1$ se a linha i está associada a coluna j, o problema pode ser formulado, conforme abaixo:

$$\text{Min} \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{8}$$

Sujeito a:

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad (i = 1,2,\ldots,n) \tag{9}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad (j = 1,2,\ldots,n) \tag{10}$$

$$x_{ij} \in \{0,1\} \qquad (i, j = 1,2,\ldots,n) \tag{11}$$

O algoritmo divide-se um duas fases: 1) construção de uma solução inicial; e 2) melhoramento da solução. Em ambas as fases o modelo de designação acima é utilizado, conforme descrito a seguir.

2.1 Construção da Solução Inicial

O algoritmo inicia construindo um grafo multi-partido. Considere o grafo $G = (T, A)$, onde T é o conjunto de vértices que representam as tarefas as quais deverão ser alocadas as tripulações e A é o conjunto de arestas a_{ij} que indicam a possibilidade de uma mesma tripulação executar a tarefa j após a tarefa i. Considere ainda que os vértices estão dispostos em camadas e que cada uma destas camadas é composta por vértices que representam tarefas que podem ser realizadas como seqüência de alguma tarefa na camada imediatamente anterior, e que não podem ser realizadas após tarefas localizadas na mesma camada ou camadas posteriores.

Portanto, o conjunto de vértices T é particionado em subconjuntos, T_1, T_2, ..., T_K , que representam os vértices (tarefas) de cada camada, assim, $T_2 \cup T_2 \cup ... \cup T_K = T$, sendo K o número de camadas (ou partições).

A construção da solução inicial consiste em resolver um problema de designação para cada camada. Cada problema de designação definido pela matriz quadrada $C = [c_{ij}]$, de ordem $|T_K| + |J|$, sendo J o conjunto de jornadas. O tamanho do conjunto de jornadas J deve ser suficiente para que nenhuma tarefa fique sem ser alocada. A matriz $C = [c_{ij}]$ pode ser dividida em quatro blocos conforme mostrados na Figura 2, sendo que seus elementos são definidos da seguinte forma:

- **I**. Para $i = 1, ..., |J|$ e $j = 1, ..., |T_K|$, se for possível alocar a tarefa j à jornada i, temos $c_{ij} = f(i, j) + CTL$, onde $j \in T_k$, a função f define o custo de alocar-se à tarefa j à jornada i, e CTL assume um valor que penaliza a troca de linha de viagem na jornada se a linha de viagem relacionada à tarefa j for diferente da linha da última tarefa da jornada i e, caso contrário, valor zero. Se não for possível alocar a tarefa j à jornada i então $c_{ij} = \infty$;
- **II**. Para $i = 1, ..., |J|$ e $j = |T_K| + 1, ..., |T_K| + |J|$, c_{ij} é o custo da jornada i sem receber tarefa adicional;
- **III**. Para $i = |J| + 1, ..., |T_K| + |J|$ e $j = 1, 2, ..., |T_K|$, c_{ij} corresponde a um valor suficientemente alto para que nenhuma tarefa fique sem alocação;
- **IV**. Para $i = |J| + 1, ..., |T_K| + |J|$ e $j = |T_K| + 1, ..., |T_K| + |J|$, $c_{ij} = 0$;

	Tarefas	Tarefas Fictícias
Jornadas	**I** $c_{ij} = f(i,j) + CTL$ ou $c_{ij} = \infty$	**II** c_{ij} = custo da jornada i sem tarefa adicional
Jornadas Fictícias	**III** $c_{ij} = \infty$	**IV** $c_{ij} = 0$

Figura 1 - Estrutura da matriz de custos C.

No início da aplicação do algoritmo, todas as jornadas estão vazias e as tarefas lhe são alocadas em cada iteração de acordo com a resolução do problema de designação da camada corrente. Ao final do procedimento as jornadas que permanecerem vazias são eliminadas.

Para formação do Bloco I pode-se utilizar uma das três funções abaixo para $f(i, j)$:

i) $f_1(i, j)$ = custo da jornada i em minutos pagos, considerando a inclusão da tarefa j na jornada i;

ii) ou, $f_2(i, j)$ = minutos ociosos na jornada i, considerando a inclusão da tarefa j na jornada i;

iii) ou, $f_3(i, j)$ = minutos ociosos na jornada i + minutos extras na jornada i * $AdHE$ considerando a inclusão da tarefa j na jornada i, onde $AdHE$ é o adicional sobre horas extras definido pela legislação como sendo de 50%.

Em suma, o procedimento para construção da solução inicial pode ser descrito pelo algoritmo da figura 2 a seguir.

Entrada: Conjunto de tarefas;
Passo 1: Faça $k = 1$;
Passo 2: Forme a camada k contendo as tarefas que não podem ser realizadas na seqüência de tarefas da mesma camada ou camadas posteriores;
Passo 3: Monte a matriz de custos C;
Passo 4: Resolva o problema de designação descrito pela matriz C e aloque as tarefas às jornadas de acordo com o resultado obtido.
Passo 5: Faça $k = k + 1$ e retorne ao Passo 2 até que todas as tarefas estejam alocadas.

Figura 2 - Procedimento para construção da solução inicial.

2.2 Melhoramento da Solução

A fase de melhoramento consiste na aplicação consecutiva de dois procedimentos denominados M1 e M2. Primeiramente, é aplicado o procedimento M1 que divide as jornadas construídas em duas partes, formando as jornadas parciais. Cada divisão (ou corte) separa as tarefas de uma jornada formando duas jornadas parciais, uma contendo as tarefas à esquerda do corte e outra contendo as tarefas à direita do corte. Em seguida, estas jornadas parciais são recombinadas com a resolução um problema de designação. A Figura 3 ilustra esta operação.

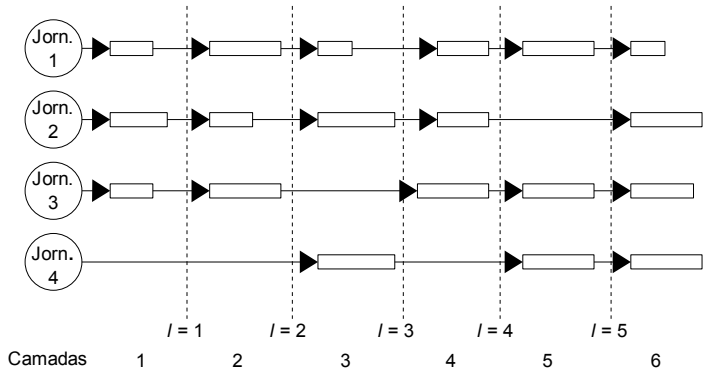

Figura 3 - *A solução obtida após a fase de construção e seu possíveis cortes. Os retângulos representam as tarefas e as setas o seqüenciamento das mesmas em uma jornada. Os cortes possíveis estão representados pelas linhas tracejadas.*

Uma iteração do procedimento M1 consiste em realizar K-1 cortes e resolver um problema de designação para cada corte. Em seguida, calcula-se os custos para recombinar as jornadas parciais à esquerda do corte com jornadas parciais à direita do corte. Estes custos formam a matriz quadrada $D = [d_{ij}]$ que representa um problema de designação. A resolução deste problema determina como as jornadas parciais devem ser recombinadas. A Figura 4 exemplifica este procedimento.

Seja *NJE* o número de jornadas parciais à esquerda do corte e *NJD* o número de jornadas parciais à direita do corte. Como regra geral, para $i = 1, ..., NJE$ e $j = 1, ..., NJD$, d_{ij} será igual ao custo em minutos pagos da jornada que combina as jornadas i e j (se for possível combiná-las) + *CTL*, onde *CTL* assume um valor pequeno que penaliza a troca de linha de viagem na jornada se a linha de viagem da última tarefa da jornada i for diferente da linha da primeira tarefa da

jornada *j* e, caso contrário, valor zero. Se não for possível combinar as jornadas *i* e *j*, $d_{ij} = \infty$.

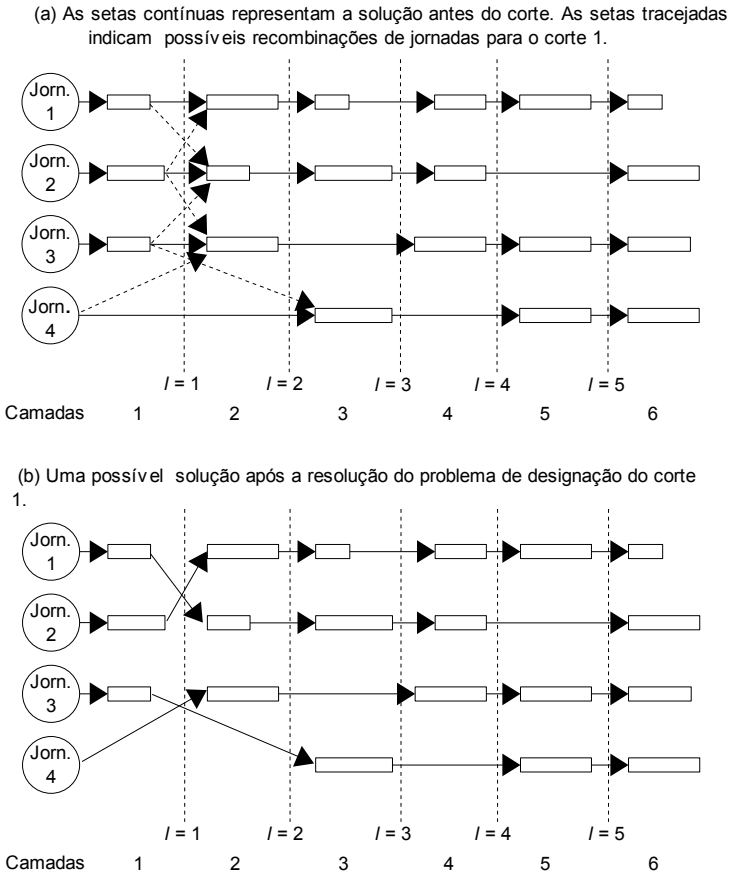

Figura 4 - Exemplo de execução do procedimento M1.

Pode ocorrer que *NJE* seja diferente de *NJD*, então para tornar a matriz *D* quadrada serão adicionadas jornadas fictícias conforme ilustrado na Figura 5. O Bloco I segue a definição original de d_{ij}, enquanto o Bloco II é definido conforme o caso:

- Caso 1. Se *NJE* > *NJD*, o Bloco II é definido como: para *i* = 1, ..., *NJE*, e j = *NJD* + 1, ..., *NJE*, d_{ij} = custo em minutos pagos da jornada *i*;
- Caso 2. Se *NJE* < *NJD,* o bloco II é definido como: para i = *NJE* + 1 , ..., *NJD*, e *j* = 1, ..., *NJD*, d_{ij} = custo em minutos pagos da jornada *j*;

O procedimento M2 realiza um corte na solução atual que consiste em separar a última tarefa das jornadas com horas extras e resolver o problema de designação para este corte. A idéia deste procedimento é diminuir o custo da solução por retirar tarefas das jornadas com horas extras e realocá-las em jornadas que estão com tempo ocioso. Após o corte calcula-se os custos para recombinar as jornadas parciais à esquerda do corte com jornadas parciais à direita do corte. Estes custos formam a matriz quadrada $D = [d_{ij}]$ que representa um problema de designação e que tem a mesma estrutura já definida para o procedimento M2. A Figura 6 exemplifica este procedimento.

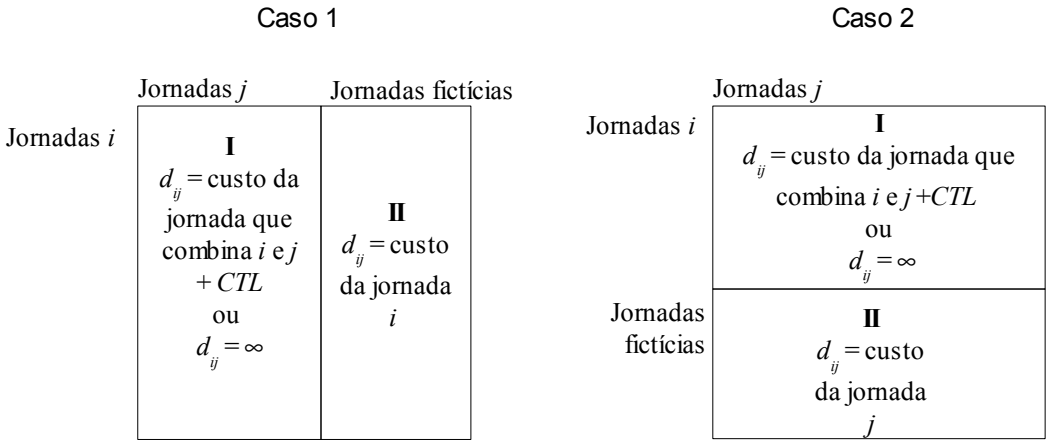

Figura 5 - Estrutura da Matriz D.

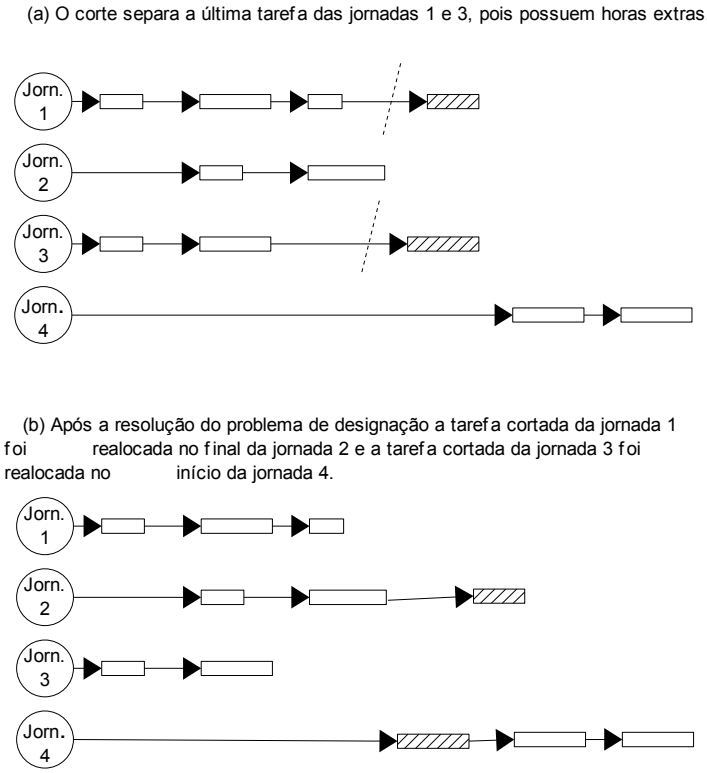

Figura 6 - Exemplo de execução do procedimento M2.

Os procedimentos M1 e M2 são executados repetidas vezes até que não haja melhora no custo da solução por *NumIter* iterações. A Figura 7 apresenta o esquema geral da fase de melhoramento.

3 Resultados e Análise dos Resultados

Para a resolução do problema de designação foi implementado o algoritmo proposto por Carpaneto e Toth (1987) que combina o método Húngaro e o método do Menor Caminho Aumentante. Para resolução dos modelos de PLI ou PL foi utilizado o resolvedor *lp-solver* de Markus Weidenauer (2003). Todos os testes computacionais foram realizados em um Pentium III

de 900 Mhz com 512 MB de memória RAM e executando sobre o sistema operacional Windows 2000.

```
Entrada: Solução inicial e NumInter;
Início
  Repita
    Repita
       Execute M1;
    Até que não haja melhora na solução por NumIter iterações.
    Repita
       Execute M2;
    Até que não haja melhora na solução por NumIter iterações.
  Até que não haja melhora na solução por NumIter iterações.
Fim.
```

Figura 7 - Procedimento para melhoramento da solução

3.1 Características das Instâncias de Teste

Duas das instâncias utilizadas nos testes são instâncias reais e as demais são instâncias aleatórias baseadas em uma instância real. Portanto, apesar de instâncias geradas de forma aleatórias, elas possuem características oriundas de instâncias reais. As instâncias são descritas abaixo, sendo que a parte numérica do nome indica o número de tarefas (*e.g.* CV412 contém 412 tarefas):

- CV412: corresponde aos dados de uma empresa local que realiza o transporte coletivo metropolitano entre cidades ;
- CC2313: corresponde aos dados de uma empresa local que realiza o transporte urbano;
- CC130, CC251, CC512, CC761, CC1000, CC1253, CC1517 e CC2010: são instâncias obtidas à partir da instância CC2313 sorteando-se *blocos* da mesma.

3.2 Resultados Obtidos

A Tabela 1 apresenta os resultados obtidos pelo algoritmo proposto (AP) usando na construção da solução inicial as funções f_1, f_2 e f_3, sendo que para o parâmetro *CTL* que penaliza as trocas de linhas foi utilizado o valor 1 e para o parâmetro *NumIter* que define o número de iterações sem melhora da solução foi utilizado o valor 4. Para cada instância é mostrado o número de jornadas (NJor) na solução, o custo da solução em minutos pagos e o tempo de execução no formato minutos:segundos. A coluna LIP mostra o valor do limite inferior para o PET calculado de acordo o modelo para escalonamento de pessoal em local fixo (Bodin *et a*l ,1983) e Gap% é o percentual da melhor solução do AP (destacada em negrito) para cada instância em relação a LIP, ou seja, $Gap = (Custo - LIP)/LIP \cdot 100$.

O tempo computacional foi pequeno, sendo o maior tempo igual a 8min8s para a instância CC2313 utilizando-se a função f_1. Os melhores resultados foram obtidos utilizando-se as funções f_2 e f_3, que obtiveram o menor custo em 4 e 6 instâncias respectivamente. Os percentuais de Gap são aparentemente elevados, porém, o valor de LIP é muito baixo uma vez que o problema da seção 3.3 é uma proposta bastante simplificado em relação ao PET.

3.3 Comparação entre os Procedimentos de Melhoramento

A Tabela 2 mostra o quanto o custo da solução inicial pode ser reduzido utilizando-se

isoladamente cada um dos procedimentos e com a combinação de ambos os procedimentos de melhoramento proposto juntamente com a função f_3.

Tabela 1 – Resultados obtidos pelo algoritmo proposto

	AP usando f_1			AP usando f_2			AP usando f_3				
		Custo da			Custo da			Custo da			
Instância	NJor	Solução	Tempo	NJor	Solução	Tempo	NJor	Solução	Tempo	LIP	Gap%
CC130	19	8451,3	00:38	19	8472,5	01:09	19	**8389,4**	00:45	8057	4,13
CC251	40	17667,5	00:36	40	17690,0	00:46	40	**17600,0**	00:35	15655	12,42
CV412	69	30810,0	01:05	69	30795,0	00:36	66	**29512,5**	00:48	25427	16,07
CC512	80	35612,5	01:08	79	**35105,0**	00:40	80	35312,5	01:00	30858	13,76
CC761	109	48395,0	02:22	110	48632,5	01:47	107	**47532,9**	01:50	43010	10,52
CC1000	152	67090,0	01:39	147	65019,4	02:10	146	**64873,6**	05:00	57000	13,81
CC1253	191	84580,0	02:29	188	83261,1	02:40	187	**82842,9**	03:36	72261	14,64
CC1517	232	102729,5	02:02	225	**99852,8**	05:29	227	100507,0	02:45	89191	11,95
CC2010	297	131482,5	05:21	290	**128964,2**	07:00	292	129637,8	03:43	116019	11,16
CC2313	339	150649,7	08:08	331	**147215,0**	04:37	340	150522,5	04:04	131800	11,70

Tabela 2 – Comparação entre os procedimentos de melhoramento

Instância	Inicial	M1	Redução	M2	Redução	M1 + M2	Redução
CV412	33394,8	29512,5	11,63%	30897,3	7,48%	**29512,5**	11,63%
CC1000	74637,0	65442,0	12,32%	68550,5	8,15%	**64873,6**	13,08%
CC1253	95622,0	82955,4	13,25%	88486,9	7,46%	**82842,9**	13,36%
CC2313	172660,4	150635,0	12,76%	158088,0	8,44%	**150522,5**	12,82%

Na Tabela 2 nota-se, também, que as soluções obtidas utilizando-se apenas M1 foram melhores que utilizando-se apenas M2. Porém, a combinação dos dois procedimentos gerou soluções melhores para as instâncias CC1000, CC1253 e CC2313. O maior ganho no uso conjunto de M1 e M2 foi para a instância CC1000 que obteve uma redução adicional de 0,76% em relação à redução obtida usando-se apenas M1. Embora este ganho seja percentualmente pequeno deve-se lembrar que seu valor econômico pode ser significativo.

3.4 Resultados com o Modelo de Cobertura de Conjunto

Nesta seção são apresentados resultados para o PET modelado como um PCC. Para as instâncias menores foi possível obter o valor ótimo usando programação linear inteira, ao passo que para as instâncias maiores foram obtidos limites inferiores usando relaxação linear ou relaxação lagrangeana dependendo do tamanho da instância.

A Tabela 3 apresenta resultados obtidos a partir da resolução do PET usando o modelo do PCC. A coluna *MaxPed* mostra o número máximo de frações por coluna, sendo que um asterisco antes do número indica que todas as colunas possíveis foram geradas. Os valores sem asterisco indicam que a instância poderia ter colunas com mais de *MaxPed* frações que não foram geradas a fim de limitar o tamanho do problema numa tentiva de tornar possível sua resolução.

A Tabela 3 também mostra o número Linhas e Colunas obtidas pela combinação das frações. A coluna ZPL traz o valor do problema de programação linear relaxado (PL), ZPLG é o valor do limite inferior Lagrangeano ((BEASLEY; 1987 e REEVES; 1993) e ZPLI é o custo da solução inteira obtida resolvendo-se o modelo de PLI. Para obtenção de ZPL/ZPLI adotou-se um limite de tempo igual a 24 horas. Para o cálculo de ZPLG não houve limite de tempo. O valor Gap% é obtido pela fórmula $Gap = (Custo - LIP)/LIP \cdot 100$, onde *Custo* é o valor de ZPL para as instâncias em que foi possível obtê-lo e para as demais é o valor de ZPLG.

Tabela 3 – Resultados obtidos para o PCC

Instância	MaxPed	Linhas	Colunas	ZPLI	ZPL	ZPLG	LIP	Gap%
CC130	*3	36	172	9530,60	9530,60	9528,90	8057	18,29
CC251	*4	77	1483	18691,20	18691,20	18587,40	15655	19,39
CV412	*4	136	6730	30455,00	30455,00	30228,10	25427	19,77
CC512	*5	162	16875	37112,50	37112,50	36468,69	30858	20,27
CC761	*5	234	47552	51752,50	51752,50	50242,08	43010	20,33
CC1000	*6	315	176734	67971,50	67971,50	65821,81	57000	19,25
CC1253	*6	398	314149	86349,00	86349,00	84052,23	72261	19,50
CC1517	*6	480	579666	105178,50	105178,50	102315,24	89191	17,93
CC2010	4	623	1206504	138035,40	138035,40	133055,62	116019	18,98
CC2313	4	715	1610242	-	-	149648,34	131800	13,54

Nota-se na Tabela 3 que foi possível obter o valor de ZPLI para quase todas instâncias, com exceção à instância CC2313 devido ao elevado número de colunas e o elevado tempo computacional sem obtenção da solução (mais de 100 horas de espera) . Os valores da solução inteira (ZPLI) para estas instâncias são ótimos, pois são iguais ao valor do custo da solução do problema relaxado (ZPL). Lembrando que o valor obtido para *ZPLG* é no máximo igual ao valor obtido pelo modelo de PL, portanto *ZPLG* deve ser usado apenas se a resolução do modelo de PL for impraticável

3.5 Comparação dos Resultados

Para uma comparação direta dos resultados obtidos pelo algoritmo proposto (AP) e pela resolução do PET usando o modelo do PCC, na Tabela 4 é mostrado o Afastamento Percentual Relativo (APR) do custo da melhor solução obtida por AP retirado da Tabela 1, em relação aos valores de ZPL (quando disponível) obtidos pelo PCC conforme Tabelas 3. Além disso, para todas as instâncias o valor obtido por AP é comparado com LIP. O APR é dado pela fórmula $APR = (Custo1 - Custo2)/Custo2 \cdot 100$, onde *Custo1* é o custo da solução de AP e *Custo2* é o valor de ZPL ou LIP, conforme a coluna da tabela.

A Tabela 4 mostra que os custos das soluções AP em todas as instâncias são menores do que os custos das soluções do PCC . As soluções do AP ficaram acima do limite inferior do problema (LIP) entre 4,13% e 16,07%.

Tabela 4 – Comparação dos resultados do AP com os resultados do PCC

Instância	ZPL	LIP
CC130	-11,97%	4,13%
CC251	-5,84%	12,42%
CV412	-3,09%	16,07%
CC512	-5,41%	13,76%
CC761	-8,15%	10,52%
CC1000	-4,56%	13,81%
CC1253	-4,06%	14,64%
CC1517	-5,06%	11,95%
CC2010	-6,57%	11,16%
CC2313	-1,63%	11,70%

4 Conclusões

A concepção do algoritmo foi resultado de uma combinação de conhecimentos de projeto e análise de algoritmos, pesquisa operacional, programação matemática, grafos e heurísticas. Sem esta integração seria muito difícil chegar os resultados apontados. Pode ser provado que este algoritmo tem complexidade assintótica $O(n^4)$, sendo n o número de tarefas. Portanto, um algoritmo de complexidade polinomial.

O desempenho computacional o algoritmo proposto (AP) foi muito satisfatório tanto em relação a qualidade das soluções obtidas quanto ao tempo computacional. Superando os resultados obtidos pelo modelo clássico PCC.

O algoritmo proposto resolve tanto problemas pequenos quanto problemas de grande dimensão em um tempo computacional bastante reduzido e com uma qualidade aceitável para as grandes instâncias, visto que seria difícil alcançar soluções melhores para estes casos, usando programação matemática, em tempo computacional aceitável.

Uma característica importantíssima desse algoritmo é que ele pode ser aplicado tanto para resolver o problema de escalonamento estático (as tarefas não mudam durante do dia) como o dinâmico (quando as tarefas são alteradas por eventos não previstos).

Agradecimento: os autores agradecem ao CNPq pelo suporte parcial conforme processo 305534/2004-1.

5 Referências Bibliográficas

BEASLEY, J. E. An algorithm for set covering problem. *European Journal of Operational Research*, v. 31, n. 1, p. 85-93, jul. 1987.

BEASLEY, J. E; CHU, P. C. A genetic algorithm for the set covering problem. *European Journal of Operational Research*, v. 94, n. 2, p. 392-404, out. 1996.

BODIN, L.; GOLDEN, B.; ASSAD, A.; BALL, M.. Routing and scheduling of vehicles and crews - The stare of the art. *International Journal of Computers and Operations Research*, v. 10, n. 2, p. 63-211, 1983.

CALVI, R. *Um Algoritmo para o Problema de Escalonamento de Tripulação em Empresas de Ônibus*. 2005. Dissertação (Mestrado em Ciência da Computação) – Universidade Estadual de Maringá.

CAPRARA, A.; FISCHETTI, M.; TOTH P. A heuristic method for the set covering problem. *Operations Research*, v. 47, n. 5, p. 730-743, set./out. 1999.

CARPANETO, G.; TOTH, P. Primal-dual algorithms for the assignment problem. *Discrete Applied Mathematics*, n. 18, p. 137-153, North-Holland, 1987.

CASTRO, E. C. de; CONSTANTINO, A. A.; MENDONÇA NETO, C. F. X. de; ARAUJO, S. A. de. Problema de Escalonamento de Condutores: Uma Metodologia Heurística para Construção do Problema de Cobertura de Conjuntos. In: XXXVII SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL-SBPO, *Anais* ... 2005, Gramado. . v. 1 , pp. 1-5.

CONSTANTINO, A. A.; REIS, P.; MENDONÇA NETO, C. F. X. de; FIGUEIREDO, M. F. Aplicação de Algoritmos Genéticos ao Problema de Cobertura de Conjunto. In: XXXV

SBPO - SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL, 2003, Natal. Anais ....v. 1, p. 1-10.

CORMEN, T. LEISERSON, C.; RIVEST, R.; STEIN, C. *Algoritmos - Teoria e Prática.* Editora Campus, 2002.

GOLDBARG, M. C.; LUNA, H. P. L. Otimização combinatória e programação linear: modelos e algoritmos, 4.ed. Rio de Janeiro: Campus, 2000.

HEIS, A. *Uma Heurística para o problema de corte de estoque unidimensional inteiro.* 2006. Proposta de Dissertação (Mestrado em Ciência da Computação – Universidade Estadual de Maringá.

MAURI, G. R.; LORENA, L. A. N. Método interativo para resolução do problema de escalonamento de programação de tripulações. In: SBPO, n. 36, 2004, São João Del Rei. *Anais*... São João Del Rei: UFMG. v.1, pp. 1- 10.

MARTELOZZI, M. R. *Otimização da Distribuição/Alocação de Turmas aos Alunos: Um Estudo de Caso*. 2004. Dissertação (Mestrado em Ciência da Computação) - Universidade Estadual de Maringá.

MAYERLE, S. F. *Um sistema de apoio à decisão para o planejamento operacional de empresas de transporte rodoviário urbano de passageiros*. 1996. Tese (Doutorado em Engenharia de Produção) – Universidade Federal de Santa Catarina, Florianópolis.

OHLSSON, M.; PETERSON, C.; SÖDERBERG, B. An efficient mean field approach to the set covering problem. *European Journal of Operational Research*, v. 133, n. 3, p. 583-595, set. 2001.

REEVES, C. R. Modern heuristic techniques for combinatorial problems, Oxford: Blackwell Scientific Publicantions, 1993.

SHEN, Y.; KWAN, R. S. K. *Tabu search for driver scheduling*, Report 2000.10, Research Report Series, School of Computer Studies, University of Leeds, 2000. Disponível em: <http://www.comp.leeds.ac.uk/research/pubs/reports/2000/2000_10.ps.gz>. Acesso em: 14 jun. 2004.

SIQUEIRA, P. H. *Aplicação do algoritmo do matching no problema da construção de escalas de motoristas e cobradores de ônibus.* 1999. Dissertação (Mestrado em Ciências) – Universidade Federal do Paraná.

NETTO, C. A. de S. *Problema de Escalonamento de Pessoal na Área de Atendimento Telefônico*. 205. Trabalho de Conclusão de Curso (Informática) – Universidade Estadual de Maringá.

SOUZA, M. J. F.; CARDOSO, L. X. T.; SILVA, G. P. Programação de tripulações de ônibus urbano: uma abordagem heurística. In: SBPO, n. 35, 2003, Natal. *Anais*... Natal: UFRN, 2003. p. 1285-1294.

VILELA JUNIOR, A. *Planejamento e Programação da Produção numa Fundição Automatizada de Grande Porte*. 2005. Proposta de Dissertação (Mestrado em Ciência da Computação) – Universidade Estadual de Maringá.

WREN, A.; WREN, D. A genetic algorithm for public transport driver scheduling. *Computers Ops Res.*, v. 22, n. 1, p. 101-110, 1995.