Pré-processamento do problema de cobertura de conjunto aplicado ao escalonamento de condutores

Ademir Aparecido Constantino[1]
Edilson Costa de Castro[1]
Silvio Alexandre de Araujo[2]
Cândido Ferreira Xavier de Mendonça Neto[3]

Resumo: o problema de escalonamento de condutores (PEC) consiste em distribuir de maneira eficiente o quadro de viagens de uma empresa de transporte coletivo entre os condutores disponíveis. Esse problema é comumente modelado como um problema de cobertura de conjunto — PCC (*set covering problem*). Nesse caso, um bom resultado para o PEC depende de uma boa construção e resolução do PCC. Porém, a maior parte da bibliografia relacionada trata apenas da resolução das instâncias do PCC, sem avaliar a influência dos procedimentos com sua construção, aqui denominado de pré-processamento. Este trabalho propõe-se e investigar metodologias heurísticas baseadas em *Simulated Annealing* para o pré-processamento de instâncias do PCC, cujas características possibilitem os algoritmos de resolução obterem melhores resultados para o PEC. Nos testes efetuados, conseguiu-se uma redução de até 8% no custo das soluções apresentadas, em comparação com a resolução de instâncias geradas por um método clássico de geração do PCC**.**

Palavras-chave: Escalonamento de condutores. Heurística. *Simulated Annealing*.

Abstract: *the crew scheduling problem consists on efficiently distribute trips of an urban transport company to available drivers. Generally it is modeled as a set covering problem (SCP). In this case, good results for the crew scheduling problem depend on good generation and resolution to the SCP. Most of works in this area proposes only SCP resolution methods, but does not include the SCP generation in their matters. This paper investigates and proposes heuristic methodologies based on Simulated Annealing to generate SCP instances, whose characteristics allow solution algorithms to obtain better results. In executed tests, costs were reduced up to 8%, in comparison with a classical method of SCP generation.*

Keywords: *Driver scheduling. Heuristic. Simulated Annealing.*

1 Introdução

A resolução do problema de planejamento de escala de veículos e condutores consiste em determinar um plano de distribuição de viagens sobre um conjunto de veículos e motoristas envolvidos na operação, ao longo de um período de planejamento. Como resultado desse processo, para cada veículo é alocado um conjunto de viagens a ser realizado e cada viagem é alocada a um condutor.

[1] Universidade Estadual de Maringá, Departamento de Informática, Av. Colombo, 5.790, Maringá (PR) — Brasil
{ademir,eccastro@din.uem.br}
[2] Universidade Estadual Paulista, Departamento de Ciências da Computação e Estatística, São José do Rio Preto (SP) — Brasil
{saraujo@ibilce.unesp.br}
[3] Universidade de São Paulo, Escola de Artes Ciências e Humanidades, São Paulo (SP) — Brasil
{cfxavier@usp.br}

http://dx.doi.org/10.5335/rbca.v8i2.5131

O planejamento de escala de veículos e condutores responde a uma série de questões que são comuns em uma empresa de transporte rodoviário de passageiros, por exemplo: número de veículos necessários; viagens que devem ser realizadas por determinado veículo; decisões sobre o abastecimento dos veículos; número de condutores necessários; início e fim de cada turno de trabalho; dias de trabalho e de folga de cada condutor; viagens que cada condutor deve cumprir para cada dia; dente outras.

Este trabalho trata apenas do *problema de escalonamento de condutores* (PEC), que pode ser dividido em dois subproblemas: o escalonamento diário (enfoque deste trabalho) e o escalonamento semanal/mensal. No escalonamento diário (ou *crew scheduling*), a carga diária de trabalho é distribuída em escalas diárias de condutores, determinando-se, assim, quantos condutores serão necessários a cada dia e quais viagens fazem parte de cada escala. O escalonamento semanal/mensal (*crew rostering*) define os condutores que cumprirão cada uma dessas escalas diárias ao longo de uma semana ou mês, determinando, por exemplo, as folgas semanais de cada condutor.

O custo com pessoal em uma empresa de transporte representa aproximadamente 50% dos custos da empresa [1], portanto, a atividade de escalonamento de condutores não é a única atividade responsável por reduzir os custos da empresa. De acordo com [2] o problema operacional geral da empresa é dividido em pelo menos três subproblemas resolvidos sequencialmente: Escalonamento de veículos, escalonamento (diário) de condutores (*crew scheduling*) e escalonamento (de longo prazo) de condutores (*crew rostering*). É de conhecimento que a resolução ótima isolada de cada um desses problemas não garante a solução ótima do problema geral. Sendo assim, existem pesquisas direcionadas para a resolução integrada do PEC e do problema de escalonamento de veículos [3]. Dada a complexidade do problema integrado e da sua resolução, as pesquisas ainda são insipientes.

Uma revisão bibliográfica de métodos de escalonamento de condutores pode ser encontrada em [4], cujos métodos são classificados em três categorias: métodos heurísticos e métodos exatos baseados em programação matemática. Alguns trabalhos nessa linha são: [5] (sistema TRACS), [6] (sistema INTERPLAN), [7] (sistema HOT) e [8] (sistema COMPACS). Segundo [4], embora muitos desses sistemas ainda estejam sendo usados em empresas do mundo todo, em termos de pesquisa acadêmica, essas metodologias já foram abandonadas. Com a melhoria dos recursos computacionais disponíveis, tornou-se possível a utilização da programação matemática como metodologia para o escalonamento de condutores, especialmente a partir dos anos 1980. Com essa nova abordagem, modelos matemáticos começaram a ser usados para esse problema, sendo mais comumente usado o modelo do problema de cobertura de conjunto (PCC), apresentado na próxima seção. Por ser um problema que não admite resolução em tempo polinomial, surgiram muitos trabalhos que resolvem o PCC usando algoritmos heurísticos [9], [10] e [11]. Além disso, algumas meta-heurísticas também foram aplicadas, tais como: algoritmos genéticos e variantes [9], [12], [13], [14] e [15]; *Ant System* [16]; Busca Tabu [17] e [18]. Métodos exatos para o PCC relacionado ao PEC também podem ser encontrados na literatura. O trabalho de [1] apresenta um algoritmo híbrido baseado em programação matemática e programação por restrição. Nesse caso, os dados de cada linha de ônibus são separados e o PEC é resolvido para cada uma dessas linhas individualmente. São consideradas duas linhas de ônibus distintas, uma com 125 e outra 266 viagens com dois pontos de possíveis trocas de condutores. O trabalho resolve o PEC de cada linha da rota de ônibus usando o PCC de custo unitário, pois o objetivo é minimizar o número de condutores.

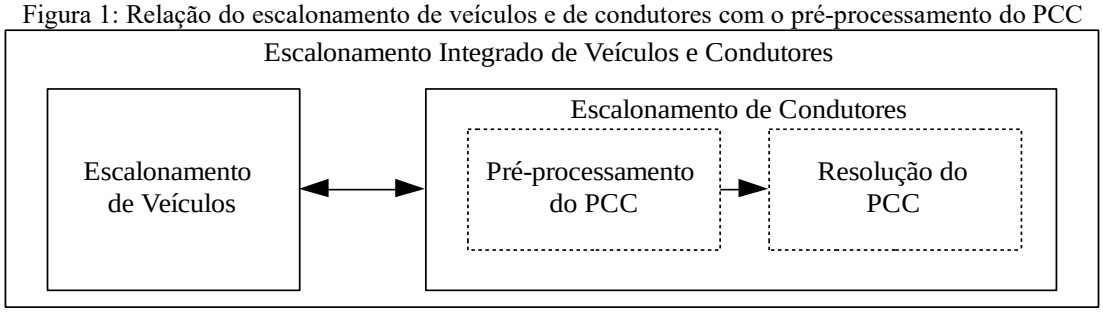
Apesar de o PCC ser um dos modelos mais empregados na tentativa de resolução do PEC, os trabalhos encontrados na literatura não têm explorado técnicas de como gerar boas instâncias do PCC antes da sua resolução, aqui denominado de *pré-processamento* do PCC. Não foi encontrado outro trabalho na literatura que investigasse esse pré-processamento do PCC aplicado ao PEC. Este trabalho mostra que essa atividade tem um papel importante para melhorar a solução do PEC via PCC. Assim, o principal objetivo deste trabalho é contribuir para o preenchimento dessa lacuna na literatura, apresentando o estudo de duas versões de um algoritmo heurístico para uma fase intermediária entre o escalonamento de veículos e de escalonamento de condutores (utilizando o PCC como modelo de resolução). A Figura 1 ilustra essa integração e a relação com o problema de pré-processamento, em que o pré-processamento compreende a geração de instância do PCC para que a seja resolvida na fase seguinte.

Para a resolução do PEC foram consideradas três instâncias de dados de duas empresas distintas. As duas maiores instâncias apresentam 602 e 657 viagens cada uma. Além disso, todos os pontos finais são considerados como possíveis pontos de troca de condutores, como também são consideradas as possibilidades de os

condutores serem alocados em diferentes linhas de ônibus. Esses dois aspectos tornam o PEC difícil de ser resolvido por algoritmo exato.

O presente artigo está dividido nas seguintes seções: além da Introdução, tem-se, na seção 2, uma descrição do problema de cobertura de conjunto. Na seção 3 analisa-se o pré-processamento para o problema de cobertura de conjunto. Na seção 4 são apresentados os métodos de resolução do problema de escalonamento de condutores. Os resultados computacionais são apresentados na seção 5 e, na seção 6, têm-se as conclusões e algumas propostas de trabalhos futuros.

Figura 1: Relação do escalonamento de veículos e de condutores com o pré-processamento do PCC



Fonte: elaborado pelos autores

2 Problema de cobertura de conjunto

Grande parte dos artigos encontrados na literatura modela o problema de escalonamento de *condutores* como um problema de cobertura de conjunto (PCC). A formulação matemática do PCC é a seguinte:

$$Min \sum_{j \in J} c_j x_j \tag{1}$$

$$\text{s. a.} \quad \sum_{j \in J} a_{ij} x_j \geq 1 \qquad i \in I$$
$$x_j \in \{0,1\} \qquad j \in J \tag{2}$$

Sendo I o conjunto de linhas a serem cobertas; J é o conjunto de colunas para cobertura das linhas; c_j é o custo da coluna; j, a_{ij}= 1 se a coluna j cobre a linha i e zero caso contrário. No caso do escalonamento de condutores, as linhas da matriz representam as viagens a serem executadas e cada coluna representa uma – possibilidade de – escala de trabalho de um condutor, que cobre (executa) uma pequena quantidade de linhas (viagens) do problema. Essas escalas são geradas a partir da enumeração completa de todas as combinações entre as linhas, que respeitem as restrições adotadas (sobreposição de horários, leis trabalhistas, regras da empresa e outros indicadores). Assim, a mesma linha pode ser coberta por várias colunas no modelo, sendo que a melhor solução consiste no subconjunto de colunas que cobre cada uma das linhas ao menos uma vez, com o menor custo possível.

O custo c_j, normalmente, corresponde ao valor pago, pela empresa, pelo cumprimento da escala, podendo variar, por exemplo, de acordo com a quantidade de horas extras contidas na escala. Outras variáveis podem entrar no cálculo de c_j, dependendo da modelagem que se queira adotar. Há casos em que o objetivo é reduzir o número de condutores, assim, nesses casos, considera-se custo unitário [1], ou seja, c_j=1, $\forall\, j \in J$.

A enumeração completa de todas as escalas possíveis, pela combinação das viagens, pode resultar em um número muito grande de colunas para o PCC. Uma estratégia encontrada na literatura para reduzir a dimensão do problema consiste em agrupar as viagens em partições de trabalho [8][19][2]. Por definição, uma *partição de trabalho* é um conjunto de viagens a serem executadas pelo mesmo condutor e com o mesmo veículo. Assim, o conjunto I passa a representar um conjunto de partições de trabalho em vez de um conjunto de viagens. Essa estratégia reduz o número de linhas da matriz $[a_{ij}]$ e, consequentemente, o número de colunas. Tais partições de trabalho são resultados da combinação de conjunto de viagens extraídos das escalas de veículos. Algumas estratégias para a formação dessas partições são apresentadas na seção 3.1.

3 Pré-processamento do problema de cobertura de conjunto

A resolução do PEC modelado como um PCC compreende duas fases: pré-processamento, responsável pela construção das instâncias do PCC, e resolução das instâncias do PCC. Este artigo constata que um bom resultado para o PEC depende tanto de uma boa construção como da solução do PCC. Porém, a maior parte da bibliografia relacionada com o assunto trata apenas da segunda fase, ou seja, da investigação de técnicas computacionais para a resolução das instâncias do PCC, enquanto que a fase de construção das instâncias do PCC é negligenciada. São raros os casos da literatura que relatam o procedimento de construção do PCC. Quando relatam, utilizam procedimentos heurísticos bastante simples, por exemplo, limitam empiricamente o número de colunas geradas que cobrem determinadas linhas (partições), cujo propósito é simplesmente limitar a dimensão do PCC gerado para reduzir o esforço computacional na resolução. Este artigo investiga um procedimento heurístico para a fase de construção do PCC, aqui denominada de pré-processamento do PCC. Essa fase pode ser dividida em duas etapas: a divisão das escalas de veículos em partições de trabalho (seção 3.1) e a combinação dessas partições em possibilidades de escalas de condutor (seção 3.2).

3.1 Divisão das escalas de veículos em partições de trabalho

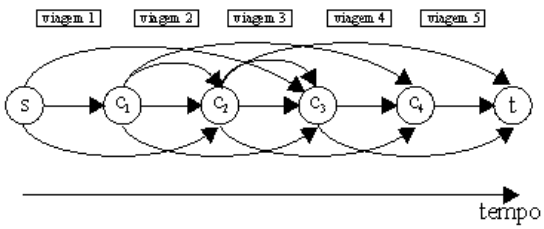
O desafio é descobrir qual é o melhor conjunto de partições de trabalho a ser extraídas da sequência de viagens diárias de cada veículo. O trabalho de [9] utiliza, para cada escala de veículo, um grafo de formação de partições de trabalho $H(I,C)$, em que o conjunto I de vértices corresponde aos intervalos de tempo (possivelmente nulos) entre cada par de viagens consecutivas que formam a escala, além dos pontos s e t, que representam, respectivamente, o início e o fim da mesma. Esta metodologia se assemelha a uma proposta encontrada em [19]. Este procedimento é uma forma heurística de combinar (agrupar) as viagens e com isso reduzir o número de colunas do PCC. Mas como se trata de um procedimento determinístico, este artigo utiliza essa ideia para criar um procedimento guloso randômico com o propósito de gerar alternativas que permitam uma exploração pelo algoritmo *Simulated Annealing*.

O grafo formado é acíclico, com um vértice de origem e um vértice terminal. A existência de uma aresta $(i_x, i_y) \in C$ indica a possibilidade de se formar uma partição de trabalho com as viagens entre os intervalos i_x e i_y. O custo c_{xy} associado a cada arco é relativo ao intervalo de trabalho t_{xy} compreendido entre os intervalos i_x e i_y. Assim, c_{xy} é dado por:

$$c_{xy} = \begin{cases} P_{\min}, \text{se } t_{xy} < P_{\min} \\ \infty, \text{se } t_{xy} > P_{máx} \\ t_{xy}, \text{nos demais casos} \end{cases} \quad (3)$$

em que P_{min} e $P_{máx}$ são, respectivamente, os limites mínimo e máximo de comprimento das partições de trabalho, estabelecidos pelo usuário. Observe-se que é permitido que uma partição de trabalho tenha um comprimento menor que P_{min}; embora, sua utilização nas escalas é dificultada pelo artifício de lhes ser atribuídas um custo maior do que seu comprimento (no caso, o próprio parâmetro P_{min}). A Figura 2 apresenta um exemplo do grafo de formação de partições de trabalho para uma escala de cinco viagens.

Figura 2: Exemplo de um grafo de formação de partições de trabalho



Fonte: [9]

Encontrar a melhor divisão de partições de trabalho para a escala em questão seria equivalente a encontrar o caminho de custo mínimo no grafo H, iniciando no vértice s e finalizando no vértice t. Esse é um problema clássico em grafos e um algoritmo mais conhecido para esse caso é o algoritmo de Dijkstra [20]. O objetivo

principal é dividir as escalas de veículos em partições de trabalho, ou seja, particionar as escalas de veículos nos pontos de maior intervalo entre as viagens, evitando, assim, que esses intervalos tornem-se períodos ociosos dentro das escalas de condutores.

Nos experimentos realizados por esse procedimento para construir o PCC, notou-se que algumas partições de trabalho eram cobertas por um número muito reduzido de colunas. Inclusive, se não for dada a oportunidade de formação de escalas de condutor com apenas uma partição de trabalho (o que ocorre em [9]) algumas partições poderão não ser cobertas por nenhuma coluna, inviabilizando o processo de escalonamento.

Assim, considerando que o algoritmo de Dijkstra é determinístico, este artigo propõe o uso de um algoritmo guloso randomizado que será utilizado na próxima seção em conjunto com o *Simulated Annealing*. O algoritmo, denominado *ParticaoRandomica*, é descrito a seguir:

ALGORITMO 1: *ParticaoRandomica*

Entrada: escala de um veículo;

Saída: sequência de vértices (que definirá conjunto de partições de trabalho);

Construir o grafo de formação de partições *H*;

v = *s*; //Iniciar com o vértice *s* do grafo *H;*

Enquanto o vértice *v* for diferente de *t* **faça**

v = vértice adjacente a *v,* selecionado aleatoriamente com probabilidade inversamente proporcional à distância.

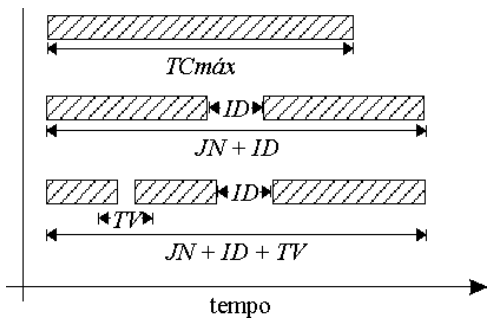
A sequência de vértices escolhidos definirá a divisão da escala de um veículo, gerando um conjunto de partições de trabalho. Essas partições servirão de solução inicial, que passarão por melhorias por meio de um algoritmo de busca local que será descrito na seção 4.

3.2 Combinação das partições em escalas de condutores

Com as partições definidas, o próximo passo é combiná-las, formando possíveis escalas de trabalho de condutores que representa uma coluna do PCC. O número de escalas possíveis (representadas por colunas no PCC) pode ser muito elevado, tendo em vista que cada escala é uma possível combinação de partições de trabalho. Para reduzir o tamanho das instâncias (linhas e colunas) do problema, alguns trabalhos da literatura aplicam métodos heurísticos para gerar apenas um subconjunto dessas colunas.

A forma mais comum para gerar essas colunas é o agrupamento dessas partições de maneira determinística seguindo algumas regras de formação impostas pela empresa. Normalmente uma escala de condutor é formada pela combinação de uma, duas, três ou até quatro partições, e deve obedecer aos seguintes parâmetros (referentes às leis trabalhistas e/ou às certas regras internas da empresa), ilustrados na Figura 3.

Figura 3: Parâmetros para a formação de escalas de condutores



Fonte: elaborado pelos autores

Sendo *TCmax* (trabalho contínuo máximo) o período de tempo máximo que um condutor pode trabalhar sem ter um período de descanso; *JN* (jornada normal de trabalho) a duração máxima de uma escala de trabalho normal (sem horas extras); ID_{min} e $ID_{máx}$ (mínimo e máximo intervalo de descanso) delimitam o comprimento de um período de descanso (não é contabilizado como parte da jornada de trabalho); *TV* (mínimo intervalo para troca de veículo) o tempo mínimo estimado para um condutor trocar de veículo entre uma partição de trabalho e outra (é contabilizado como parte da jornada de trabalho) e *HE* (máximo período de horas extras) a quantidade máxima de horas extras que uma escala pode conter. Uma escala não pode ter duração maior do que *JN* + *HE*.

4 Algoritmo para o pré-processamento

A meta principal desta pesquisa é investigar algumas formas de melhorar a formação de partições com objetivo de obter soluções melhores para o PEC a partir da resolução do PCC. Para isto, são propostos dois algoritmos para o pré-processamento baseados na meta-heurística *Simulated Annealing* (SA) [21][22].

Os algoritmos propostos são iniciados com uma solução obtida pelo algoritmo *ParticaoRandomica* para cada uma das escalas de veículos. Os algoritmos fazem modificações no conjunto de partições por meio de uma estrutura de vizinhança, buscando otimizar uma função objetivo. Essas modificações geram novas partições de trabalho. Uma descrição geral utilizada para os dois algoritmos é dada a seguir.

Os parâmetros utilizados pelo algoritmo geral são:

MT: tempo de execução utilizado como critério de parada do algoritmo;
T: temperatura inicial;
TL: número de iterações para cada valor da temperatura (*T*) e
CF: fator de resfriamento (decréscimo da temperatura a cada *TL* iterações).

ALGORITMO 2 : *APP*

Construir uma solução *S* usando o *ParticaoRandomica;*
Enquanto não ultrapassado o tempo *MT* **faça**
  Para *int* = 1 até *TL* **faça**
    S' = solução vizinha a *S*; seu custo é dado por *f*(*S'*);
    $\delta = f(S') - f(S)$;
    Se $\delta <= 0$ **faça**
      $S = S'$;
      $S^* = S'$;
    senão
      com probabilidade $e^{-\delta / T}$, faça $S = S'$;
    fim-se
  fim-para
  $T = T \,.\, CF$;
fim-enquanto

A seguir, são apresentados alguns passos importantes para a utilização da meta-heurística *Simulated Annealing* tais como: geração da solução inicial, geração da solução vizinha e cálculo da função *f* (custo da solução).

Considere as seguintes notações: $V = \{v_1, v_2, ..., v_n\}$ o conjunto de viagens, sendo *n* o número de viagens; $U_i \subseteq V$, $i=1, ...,u$, escala do veículo *i* para um número de *u* veículos, sendo que $\bigcap_{i=1}^{u} U_i = \phi$ (as escalas não possuem viagens em comum) e $\bigcup_{i=1}^{u} U_i = V$ (a união de todas as escalas cobre todas as viagens). Considere $P_{ij} \subseteq U_i$, $i=1, ..., u$, $j=1, ..., p_i$, sendo a partição *j* da escala do veículo *i*, de forma que $\bigcap_{j=1}^{p_i} P_{ij} = \phi$ (as partições são distintas) e $\bigcup_{j=1}^{p_i} P_{ij} = U_i$ (a união das partições cobrem todas as viagens do veículo *i*).

Os dados de entrada para o algoritmo *APP* são as escalas de veículos, portanto, foi considerada a existência dessas escalas. Define-se uma solução *S* como sendo um conjunto contendo todas as partições P_{ij} resultantes da divisão de todas as escalas de veículos U_i, i=1, ..., u , j=1, ..., p_i .

4.1 Função de avaliação

Para o cálculo da função de avaliação é necessário construir a instância do PCC correspondente à solução *S* (uma solução *S* é um conjunto de escalas de veículos com todas as suas escalas divididas em partições de trabalho). Por sua vez, para construir a instância do PCC é necessário enumerar todas as alternativas de escala de condutor possíveis usando as partições da solução *S*.

Todas as combinações possíveis de 1, 2, 3 e 4 partições são testadas na geração de escalas de trabalho. Porém, em razão das restrições e da possível sobreposição de horários, nem toda combinação de partições é capaz de gerar uma escala.

Depois de gerado o conjunto de alternativas de escalas de condutores (colunas do PCC), o custo de uma solução *S* pode ser calculado, é quando se determina o valor da função de avaliação. Foram implementadas duas variantes para o cálculo da função de avaliação, gerando os algoritmos APP1 e APP2, tomando como base o algoritmo *APP* apresentado anteriormente.

4.1.1 Algoritmo de pré-processamento 1

A primeira variante busca por um PCC cuja linhas estejam cobertas pelo maior número de colunas, e da maneira mais homogênea possível. Assim, as instâncias do PCC que são geradas durante a execução do algoritmo são avaliadas de acordo com essa cobertura e homogeneidade e a função de avaliação é calculada da seguinte forma:

Seja *cob*(*P*) o número de colunas que cobrem a partição $P \in S$. Então:

$$f(S) = col(S) \times \frac{\delta(S)}{\mu(S)^2} \tag{4}$$

sendo *col*(*S*) o número de colunas do problema; $\delta(S)$ e $\mu(S)$ são, respectivamente, a média e o desvio padrão de *cob(P)*.

Ao minimizar o valor de *f*(*S*), o algoritmo APP1 estará procurando por uma instância do PCC, em que cada linha é coberta pelo maior número de colunas possível ($\mu(S)$), de forma homogênea ($\delta(S)$) e sem aumentar desnecessariamente o número de colunas (*col*(S)).

4.1.2 Algoritmo de pré-processamento 2

A segunda variante avalia as instâncias do PCC geradas utilizando um método guloso de resolução, cujo custo da solução obtida é utilizado como uma estimativa da qualidade dessa solução. Assim, o PCC construído é submetido a um algoritmo heurístico construtivo com o objetivo de obter uma estimativa do custo da solução rapidamente. Nesse caso foi utilizado o algoritmo de [23]. Dessa forma, *f*(*S*) é a soma dos custos das colunas que participam da solução construída para o PCC.

4.2 Geração de uma solução vizinha

Em ambas as variantes do algoritmo, para gerar uma solução vizinha *S′*, é necessário, em primeiro lugar, selecionar-se uma partição *P* da solução atual *S* para ser modificada. A partição selecionada é então submetida a sete diferentes heurísticas de alteração, que dão origem a sete novas soluções. Finalmente, uma dessas soluções é selecionada aleatoriamente como sendo a solução vizinha *S*, com probabilidades de seleção inversamente proporcionais aos seus respectivos custos. A seguir, esse procedimento é descrito com mais detalhes.

4.2.1 Seleção da partição

Seguem as formas de selecionar a partição:

- **APP1:** na primeira variante do algoritmo, a seleção da partição P a ser modificada leva em consideração a instância do PCC criado a partir da solução atual S. A seleção é feita com base nos valores de $cob(P)$ para as partições $P \in S$. Selecionam-se aleatoriamente duas partições, das quais aquela com maior $|cob(P) - \mu(S)|$ (abordagem *tournament*) será a partição escolhida para sofrer a modificação.

- **APP2:** para a segunda variante do algoritmo, é selecionada aleatoriamente uma partição com distribuição uniforme.

4.2.2 Modificação da partição

Após a seleção da partição P, ela é submetida a sete heurísticas de modificação. Sendo U_i a escala de veículo que contém a partição P. As heurísticas são as seguintes:

- **Expansão do limite inferior:** caso exista uma partição $P' \in U_i$ anterior a P, a última viagem de P' será transferida para a partição P. O processo se repete até que o local de origem da viagem transferida seja um ponto de troca de condutores.

- **Retração do limite inferior:** caso exista uma partição $P' \in U_i$ anterior a P, a primeira viagem de P' será transferida para a partição P. O processo se repete até que o local de origem da viagem transferida seja um ponto de troca de condutores.

- **Expansão do limite superior:** caso exista uma partição $P' \in U_i$ posterior a P, a primeira viagem de P' será transferida para a partição P. O processo se repete até que o local de origem da viagem transferida seja um ponto de troca de condutores.

- **Retração do limite superior:** caso exista uma partição $P' \in U_i$ posterior a P, a última viagem de P' será transferida para a partição P. O processo se repete até que o local de origem da viagem transferida seja um ponto de troca de condutores.

- **Divisão:** seja k o número de oportunidades de troca existentes dentro da partição P (excluindo-se seu início e fim). Se $k > 1$, divide-se a partição P em duas novas partições no ponto da n-ésima oportunidade de troca. O valor de n é dado $\lfloor k/2 \rfloor$.

- **União:** une-se a partição P a uma de suas partições vizinhas (caso exista o vizinho anterior e posterior, seleciona-se aleatoriamente um deles); caso essa união resulte em uma partição maior do que o permitido, transfere-se para P apenas as viagens possíveis.

- **Reconstrução das partições da escala:** refaz-se toda a divisão em partições da escala U_i, utilizando o algoritmo guloso randomizado descrito no final da seção 3.1.

4.3 Períodos ociosos dentro de escalas

A utilização do algoritmo de Dijkstra para efetuar a divisão das escalas de veículos em partições faz com que essas partições contenham o mínimo possível de períodos ociosos, assim como as escalas de condutor que serão formadas a partir delas. A metodologia aqui proposta, ao procurar por novos conjuntos de partições, acaba efetuando uma relaxação nesta restrição. Para controlar essa relaxação, desenvolveu-se um mecanismo para evitar que períodos ociosos sejam incorporados excessivamente às partições de uma solução: seja S_0 a solução inicial do algoritmo, $t(p)$ a duração de uma partição p qualquer; são descartadas automaticamente as soluções S' que apresentarem:

$$\sum_{p' \in S'} t(p') > \sum_{p' \in S_0} t(p) + \%perda_{\max} \tag{5}$$

sendo $\%perda_{max}$ um parâmetro de entrada definido para o algoritmo.

5 Resultados computacionais

Nesta seção, são apresentados os resultados obtidos com a utilização das duas variantes do algoritmo de pré-processamento do PCC (APP1 e APP2) apresentadas na seção anterior. Para avaliação dos resultados, foram utilizadas três bases de dados distintas:

- As bases de dados de [9], correspondentes aos dados reais de uma empresa de transporte coletivo. Foram utilizados os quadros de viagens de dias úteis (602 viagens) e de domingos e feriados (212 viagens). Essas bases serão designadas por M602 e M212, respectivamente;
- A base de dados de uma empresa do estado do Paraná, com 657 viagens, referenciada como CV657.

Os testes foram efetuados em um computador equipado com processador AMD XP 2.7GH e 4 GB de memória RAM. Tanto APP1 quanto APP2 foram executados a partir de uma solução inicial obtida com o algoritmo *ParticaoRandomica* para efetuar a divisão das escalas de veículo. Assim, para efeito de comparação, os resultados obtidos com a aplicação de APP1 e APP2 serão confrontados com aqueles obtidos com a aplicação do algoritmo de Dijkstra.

As instâncias do PCC geradas pelas três abordagens (algoritmo de Dijkstra, APP1 e APP2) sobre as três bases de dados citadas (M602, M212 e CV657) foram submetidas a dois algoritmos genéticos cedidos por [24] e [25], designados aqui por AG1 e AG2, respectivamente. Esses algoritmos genéticos têm a função de resolver o PCC como uma forma de avaliar o resultado dos algoritmos propostos. Nos experimentos realizados foram utilizados os seguintes parâmetros:

AG1 - Tamanho da população: 1000 indivíduos; taxa de mutação: 3%;
AG2 - Tamanho da população: 400 indivíduos; número de ciclos após convergência: 1000.

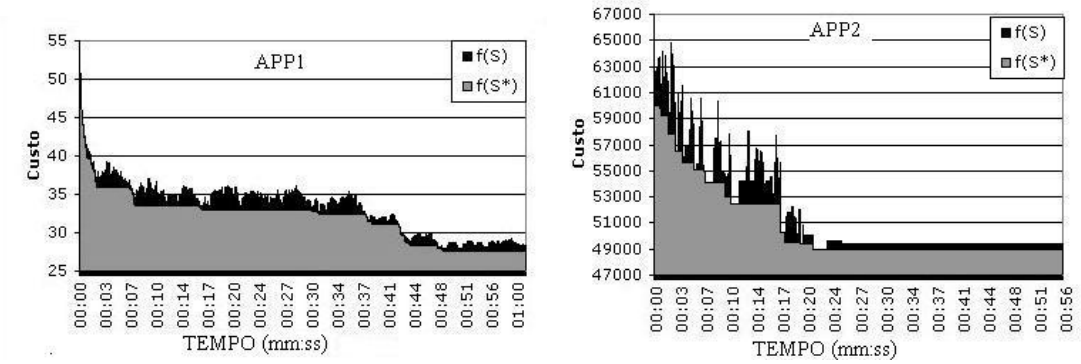
Os resultados para cada uma das três bases de dados são apresentados nas três seções seguintes. Em cada seção apresenta-se inicialmente a geração de uma única instância do PCC pelo algoritmo de Dijkstra e, em seguida, quatro diferentes resoluções do PCC com os algoritmos AG1 e AG2. Posteriormente, tem-se a geração de quatro diferentes instâncias do PCC com o algoritmo APP1, seguida de quatro diferentes resoluções para cada instância do PCC com os algoritmos AG1 e AG2. Por fim, repete-se o procedimento anterior gerando as instâncias do PCC com o algoritmo APP2. Os melhores resultados de cada tabela são destacados em negrito.

5.1 Calibragem dos parâmetros

Os parâmetros utilizados nos algoritmos APP1 e APP2 foram calibrados com base na observação da convergência e exploração do espaço de busca pelos algoritmos. Vários experimentos foram realizados para se chegar ao ajuste desses parâmetros.

A Figura 4 ilustra um exemplo dos algoritmos sobre base de dados CV657. O gráfico mostra uma convergência gradual dos algoritmos (cor cinza), enquanto que continuam explorando o espaço de soluções (cor preta) por meio do mecanismo de aceitação probabilística para soluções vizinhas pior que a incumbente. Esse é um aspecto importante para o desempenho de um algoritmo heurístico baseado em *Simulated Annealing*.

Figura 4: Gráfico de convergência dos algoritmos APP1 e APP2



Fonte: elaborado pelos autores

5.2 Experimento 1 - Resultados sobre a base de dados M602

As tabelas a seguir resumem os parâmetros utilizados e os resultados. A coluna *PCC* significa o rótulo para o problema gerado na tabela, *MT* é o tempo em segundos, *partições* é o número de partições (linhas) do problema gerado, *escalas* é o número de escalas (equivalentes ás colunas do PCC) do problema gerado e $\%perda_{max}$ é o valor para o parâmetro de entrada do algoritmo de pré-processamento.

Tabela 1: Características do PCC gerado para a instância M602

Algoritmo	*PCC*	Parametrização					Resultados	
		T	*CF*	*TL*	*MT*	% perda$_{max}$	*partições*	*escalas*
Dijkstra	0.1	-	-	-	-	-	177	5451
APP1	1.1	1000	0,95	40	60	0,05	179	6276
	1.2	1000	0,95	40	60	0,15	178	5886
	1.3	1000	0,95	120	240	0,25	178	5376
	1.4	1000	0,95	120	60	0,25	177	5494
APP2	2.1	500000	0,95	60	120	0,01	177	5453
	2.2	500000	0,95	100	240	0,01	178	5383
	2.3	500000	0,95	60	120	0,1	177	5616
	2.4	500000	0,95	100	300	0,1	177	5611

Fonte: elaborada pelos autores

As tabelas a seguir resumem resultados obtidos, sendo que *it.* significa o número de iterações efetuadas pelo algoritmo, *cond.* o número de condutores utilizados na solução gerada e *tempo* dado em segundos.

Tabela 2: Resolução do PCC para a instância M602

Algoritmo	*PCC*	AG1				AG2			
		it.	*cond.*	*custo*	*tempo*	*it.*	*cond.*	*custo*	*tempo*
Dijkstra	0.1	54085	90	40048	155	44800	91	**38408**	125
	0.1	55706	90	**39787**	161	43400	91	38435	70
	0.1	53857	90	39912	169	47050	91	38439	112
	0.1	51622	90	39823	149	44350	91	38414	49
APP1	1.1	51596	89	39383	158	56950	90	38007	107
	1.1	50954	89	39065	158	40500	90	38093	114
	1.1	62585	90	39652	189	45450	91	38316	152
	1.1	51574	89	39285	159	54150	90	38075	67
	1.2	39731	90	39731	158	56350	91	38449	150
	1.2	70044	90	39650	198	76750	91	38359	107
	1.2	148124	89	**38478**	402	67700	91	38406	139
	1.2	57103	90	39609	164	66930	91	84405	132
	1.3	49615	89	40016	142	63200	90	38127	106
	1.3	55779	89	39787	158	44250	90	38120	112
	1.3	75717	89	39033	181	55350	91	38337	158
	1.3	50240	89	39760	144	74800	90	38215	140
	1.4	49782	89	39820	139	53800	90	38148	100
	1.4	97099	89	38979	264	55450	89	**37826**	41
	1.4	76299	89	39191	212	54000	90	38059	95
	1.4	64415	89	39562	182	54450	90	38063	80
APP2	2.1	56052	89	39814	160	56950	90	38007	107
	2.1	58494	89	39125	162	40500	90	38093	114
	2.1	82078	90	39297	236	45450	91	38316	152
	2.1	56091	89	39069	165	54150	90	38075	67
	2.2	61149	88	39307	174	56350	91	38449	150
	2.2	70914	89	39156	208	76750	91	38359	107
	2.2	54187	89	39953	165	67700	91	38406	139
	2.2	53054	89	39977	161	66930	91	84405	132
	2.3	51643	90	40153	160	63200	90	38127	106
	2.3	49245	90	39845	146	44250	90	38120	112
	2.3	38837	89	39735	106	55350	91	38337	158
	2.3	63440	90	39491	192	74800	90	38215	140
	2.4	58987	89	39028	161	53800	90	38148	100
	2.4	48002	88	39659	135	55450	89	**37826**	41
	2.4	68944	88	**38886**	189	54000	90	38059	95
	2.4	55507	88	39038	138	54450	90	38063	80

Fonte: elaborada pelos autores

Nos testes sobre a base de dados M602, verifica-se que ambos algoritmos (APP1 e APP2) conseguiram melhorar os resultados obtidos pelos algoritmos de resolução do PCC, AG1 e AG2 — em comparação com as instâncias do PCC geradas pelo algoritmo de Dijkstra. Isso pode ser notado pelas colunas *custo*, na qual estão destacados em negrito os melhores custos obtidos com os algoritmos Dijkstra, APP1 e APP2. O melhor custo do Dijkstra foi 38408 com AG2, enquanto que o menor custo para APP1 e APP2 foi 37826, usando AG2 para ambos. A redução do custo das soluções variou entre 1,5 e 3,3%, além disso, nota-se a redução do número de até dois condutores, sendo noventa o menor número com Dijkstra e 88 o menor número com APP2.

5.3 Experimento 2 - Resultados sobre a base de dados M212

As tabelas 3 e 4 apresentam os resultados alcançados com essa instância.

Tabela 3: Características do PCC gerado para a instância M212

Algoritmo	*PCC*	Parametrização					Resultados	
		T	*CF*	*TL*	*MT*	% perda$_{max}$	*partições*	*escalas*
Dijkstra	0.1	-	-	-	-	-	66	528
APP1	1.1	300	0,99	30	60	0,01	65	496
	1.2	300	0,99	30	60	0,05	64	473
	1.3	500	0,99	40	60	0,1	63	454
	1.4	500	0,99	40	60	0,2	62	412
APP2	2.1	200000	0,99	40	60	0,01	63	449
	2.2	200000	0,99	40	60	0,05	63	444
	2.3	200000	0,99	40	60	0,1	62	411
	2.4	300000	0,99	40	60	0,1	62	434

Fonte: elaborada pelos autores

Tabela4: Resolução do PCC para a instância M212

Algoritmo	*PCC*	AG1				AG2			
		it.	*cond.*	*custo*	*tempo*	*it.*	*cond.*	*custo*	*tempo*
Dijkstra	0.1	6194	33	**14589**	14	9150	33	14492	01
	0.1	5803	33	14922	13	8150	33	14503	01
	0.1	5849	33	14731	13	7300	33	14537	01
	0.1	6195	33	14658	14	14600	33	**14416**	02
APP1	1.1	5402	32	14934	12	13150	33	14547	01
	1.1	**4053**	32	14692	09	17500	33	14453	02
	1.1	4110	32	14377	09	11150	33	14570	01
	1.1	5529	32	14510	13	10200	33	14608	01
	1.2	4000	32	14614	12	20100	32	14426	02
	1.2	5559	32	14708	12	15600	32	14488	02
	1.2	5535	31	**14359**	15	18600	32	14392	02
	1.2	3944	31	14422	04	15400	32	14454	02
	1.3	5077	31	14493	11	11750	33	14543	01
	1.3	5620	31	14715	12	11150	33	14556	01
	1.3	5496	32	14469	12	13800	32	**14362**	02
	1.3	4773	31	14576	10	16850	33	14535	02
	1.4	5252	33	15023	12	13400	34	15076	01
	1.4	7116	33	15163	16	11200	34	15071	01
	1.4	4264	33	15269	10	18950	34	15023	02
	1.4	4014	33	15418	09	10750	34	15072	01
APP2	2.1	4135	31	14005	10	11650	31	**13729**	01
	2.1	3408	31	13902	08	12500	31	13764	01
	2.1	4081	31	**13740**	09	7850	31	13778	01
	2.1	4954	31	14021	12	8650	31	13765	01
	2.2	3691	32	14568	07	14250	32	14151	01
	2.2	4308	31	14025	08	6950	32	14158	01
	2.2	6243	32	14338	07	9650	32	14121	01
	2.2	3406	31	14319	06	10050	32	14118	01
	2.3	6476	31	14042	13	11150	31	13988	01
	2.3	5489	32	14325	11	6200	32	14109	01
	2.3	4603	31	14029	09	8350	31	14018	01
	2.3	3625	31	14257	06	8400	31	14059	01
	2.4	5541	31	14070	12	13450	31	13874	01
	2.4	5566	31	14101	12	14450	31	13842	01
	2.4	4451	31	14380	10	18350	31	13826	02
	2.4	5868	31	14073	11	6900	31	13997	01

Fonte: elaborada pelos autores

Sobre a base de dados M212, verifica-se, também, que ambos os algoritmos (APP1 e APP2) conseguiram melhorar significativamente os resultados obtidos pelos algoritmos de resolução do PCC, AG1 e AG2 – em comparação com as instâncias do PCC geradas pelo algoritmo de Dijkstra. Isso pode ser observado nas colunas *custo*, em que estão destacados em negrito os melhores custos obtidos com os algoritmos Dijkstra, APP1 e APP2. O melhor custo do Dijkstra foi 14416 com AG2, enquanto que o menor custo com APP1 e APP2 foram 14359 (com AG1) e 13729 (com AG2), respectivamente. Portanto, a redução do custo das soluções chegou a 4,7%, utilizando-se até dois condutores a menos como pode se constatar, com os menores valores para as colunas *cond*, sendo 33 com Dijkstra e 32 para APP1 e APP2.

5.4 Experimento 3 - Resultados sobre a base de dados CV657

As tabelas 5 e 6 exibe os resultados alcançados com a presente instância.

Tabela 5: Características do PCC gerado para a instância M657

Algoritmo	*PCC*	Parametrização					Resultados	
		T	*CF*	*TL*	*MT*	% perda$_{max}$	*partições*	*escalas*
Dijkstra	0.1	-	-	-	-	-	125	1196
APP1	1.1	600	0,95	60	60	0,01	139	2607
	1.2	600	0,95	120	60	0,01	144	3473
	1.3	600	0,95	60	60	0,1	144	3210
	1.4	600	0,95	120	60	0,1	143	3227
APP2	2.1	2000000	0,95	40	60	0,01	128	2040
	2.2	2000000	0,95	40	60	0,05	138	3581
	2.3	2000000	0,95	80	60	0,05	130	2840
	2.4	2000000	0,95	60	60	0,1	132	2228

Tabela 6: Resolução do PCC para a instância M657

Algoritmo	*PCC*	AG1				AG2			
		it.	*cond.*	*custo*	*tempo*	*it.*	*cond.*	*custo*	*tempo*
Dijkstra	0.1	18149	87	40090	19	17450	87	**38983**	18
	0.1	20405	87	40220	19	21450	87	38983	20
	0.1	19782	87	40056	36	19800	87	38983	18
	0.1	20274	87	**39964**	37	22550	87	38983	16
APP1	1.1	42155	83	38431	52	47700	83	37117	37
	1.1	35107	83	38032	39	30500	83	37112	30
	1.1	33957	83	38716	31	46300	83	37063	72
	1.1	34137	83	38350	86	41350	83	37067	37
	1.2	34646	81	37982	39	58050	82	36606	145
	1.2	32644	82	37776	51	57150	82	36619	42
	1.2	32799	81	**37193**	58	57150	82	36677	56
	1.2	34426	82	37698	58	74600	82	36628	59
	1.3	29958	81	37576	75	51650	82	36712	75
	1.3	33257	81	37927	91	58100	82	36715	43
	1.3	38911	81	37423	104	55250	81	**36382**	79
	1.3	50550	81	37351	134	53850	81	36384	56
	1.4	33729	81	37659	96	71750	81	36630	104
	1.4	39209	81	37776	104	63100	81	36647	125
	1.4	29074	81	37922	72	65750	81	36683	54
	1.4	28587	81	38094	78	65150	81	36696	63
APP2	2.1	20123	83	37658	47	25700	83	36900	29
	2.1	23365	83	37487	54	18550	83	36952	33
	2.1	30440	83	37488	70	18600	83	36957	44
	2.1	22592	83	37831	53	21700	83	36952	40
	2.2	24809	82	38145	60	21400	82	36738	53

	2.2	24102	82	38214	59	30600	82	36635	68
	2.2	32031	82	37606	77	27600	82	36680	67
	2.2	39510	82	37653	93	28250	82	36694	56
	2.3	19417	81	36833	48	25950	82	36384	59
	2.3	18886	81	37134	47	22250	81	**36074**	25
	2.3	22093	81	37195	55	33600	82	36402	86
	2.3	23031	81	**36677**	57	24900	82	36402	29
	2.4	21906	83	38026	53	24650	83	37235	54
	2.4	24825	83	37779	65	25600	83	37235	40
	2.4	22421	83	38054	51	26100	83	37237	47
	2.4	22111	83	38609	50	26450	83	37234	60

Fonte: elaborada pelos autores

Sobre a base de dados CV657, verifica-se que ambos algoritmos (APP1 e APP2) também conseguiram melhorar os resultados obtidos pelos algoritmos de resolução do PCC, AG1 e AG2 – em comparação com as instâncias do PCC geradas pelo algoritmo de Dijkstra. Considerando os melhores custos obtidos com os algoritmos Dijkstra, APP1 e APP2, destacados, em negrito, na coluna c*usto*, nota-se que o melhor custo do Dijkstra foi 38983 com AG2, enquanto que o menor custo com APP1 e APP2 foram 36382 (com AG2) e 36074 (com AG2), respectivamente. A redução do custo das soluções chegou a 8,2%, utilizando-se até seis condutores a menos, como pode ser notado na coluna *cond*., sendo 87 condutores com Dijkstra e 81 condutores com APP1 e APP2.

Nota-se pelos resultados apresentados que, ao contrário das instâncias geradas para as duas bases de teste anteriores, a quantidade de partições geradas pelo algoritmo de pré-processamento é maior em relação à instância gerada pelo algoritmo de Dijkstra. Em termos percentuais, o acréscimo no número de partições girou em torno de 15%.

6 Conclusões e trabalhos futuros

Considerando que o problema de cobertura de conjunto é um dos modelos mais empregados para a resolução do problema de escalonamento de condutores, este artigo investigou o problema de pré-processamento do PCC que surge na resolução do PEC em empresas de transporte coletivo. Esse pré-processamento é praticamente negligenciado pela literatura, assim, este artigo propõe e investiga uma abordagem inédita para esta questão e constata-se a sua importância para obter soluções melhores para o PEC. Para isto, foram propostas e investigadas duas versões de algoritmos inspirados na meta-heurística *Simulated Annealing*, sendo APP1 e APP2, que utilizam um procedimento guloso randômico para encontrar partições alternativas iniciais e gerar as instâncias do PCC.

Os algoritmos foram testados com dados reais e de empresa de transporte de grande porte. As duas versões dos algoritmos apresentaram resultados melhores comparados com uma técnica da literatura que utiliza a obtenção de caminhos mínimos em grafos para a construção do PCC. A redução dos custos no PEC chegou até 8,2% para a maior instância analisada. Apesar de não haver outros algoritmos na literatura para comparações, os resultados indicam que a investigação do pré-processamento tem um papel importante para a redução dos custos operacionais no escalonamento de condutores quando modelado como um PCC. Considerando que o custo com pessoal em uma empresa de transporte representa aproximadamente 50% dos custos da empresa [1], nota-se que aplicação dessa abordagem de pré-processamento é vantajosa uma vez que o esforço computacional neste processo é baixo (poucos minutos de processamento) frente aos ganhos na qualidade das soluções.

Os algoritmos propostos, APP1 e APP2, utilizaram outros dois algoritmos, AG1 e AG2, que auxiliaram na resolução do PCC. No entanto, eles podem ser substituídos por outros algoritmos heurísticos, ou exatos da literatura, para a resolução do PCC. Considerando a hipótese de substituí-los por algoritmos mais eficientes e efetivos reforça a possibilidade de melhorar ainda mais os resultados alcançados pelos algoritmos de pré-processamento propostos, aumentando a credibilidade sobre os algoritmos APP1 e APP2.

Por fim, destaca-se que não foram encontrados outros trabalhos na literatura que avalie o pré-processamento do PCC aplicado ao PEC. Assim, este trabalho também contribui para a cobertura dessa lacuna na literatura.

Como trabalhos futuros, os resultados obtidos neste trabalho incentivam a pesquisa de novas metodologias para o pré-processamento do PCC na resolução do PEC, como a exploração de outras meta-heurísticas e procedimentos de busca local. Além disso, esta abordagem de resolução com pré-processamento introduz uma nova possibilidade de resolução integrada do problema de escalonamento de veículos e de condutores, usando o PCC como modelo para o PEC, uma vez que pré-escalas de veículos poderiam ser avaliadas pelo pré-processamento.

Referências

[1] YUNES, T. H.; MOURA, A. V.; DE SOUZA, C. C. Hybrid column generation approaches for urban transit crew management problems. *Transportation Science*, Atlanta, GA. v. 39, n. 2, p. 273–288, 2005.

[2] BODIN, L.; GOLDEN, B.; ASSAD, A.; BALL, M. Routing and scheduling of vehicles and crews: the state of the art. *Computers and Operations Research*, Amsterdam, v. 10, n. 2, p. 63-21, 1983.

[3] FRELING R.; HUISMAN, D.; WAGELMANS, A. P. Models and algorithms for integration of vehicle and crew scheduling. *Journal of Scheduling*, New York, v. 6, n. 1, p. 63-85, 2003.

[4] LAYFIELD, C. *A constraint programming pre-processor for duty scheduling.* Tese (Doutorado em Ciência da Computação) — University of Leeds, England, 2002.

[5] PARKER, M. E.; SMITH, B. M. Two approaches to computer crew scheduling. In: A. Wren, *Computer Scheduling of Public Transport,* Amsterdam: Springer Verlag, 1981, p. 193-221.

[6] MOTT, P.;FRITSCHE, H. INTERPLAN — an interactive program system for crew scheduling and rostering of public transport. *Computer-Aided Transit Scheduling*, Berlin,: Springer Verlag*,* v. 308, n. 1, p. 200-211, 1988.

[7] DADUNA, J. R.; MOJSILOVIC, M. Computer-aided vehicle and duty scheduling using HOT programme system. *Computer-Aided Transit Scheduling*, Berlin,: Springer Verlag*,* v. 308, n. 1, p. 133-146, 1988.

[8] WREN, A.; FORES, S.; KWAN, A.; KWAN, R.; PARKER, M.; PROLL L. et al. A flexible system for scheduling drivers. *Journal of Scheduling*, London, v. 6, p. 437-455, 2003.

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

[10] SMITH, B. M. IMPACS — a bus crew scheduling system using integer programming. *Matematical. Programming*, Nova York, v. 42, n. 1, p. 181-187, 1988.

[11] LESSARD, R.; ROUSSEAU, J. M.; DUPUIS, D. HASTUS I: a mathematical programming approach to the bus driver scheduling problem. *Computer Scheduling of Public Transport*, Netherlands North Holland*,* p. 255-267, 1981.

[12] LI, J. *Fuzzy evolutionary approaches for bus and rail driver scheduling.* Tese (Doutorado em Ciência da Computação) — University of Leeds, England, 2002.

[13] MAURI, G. R. *Novas heurísticas para o problema de escalonamento de tripulações*. Dissertação (Mestrado em Computação Aplicada) — Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2004.

[14] KWAN, A. S. K., KWAN, R. S. K.; WREN, A. Driver scheduling using genetic algorithms with embedded combinatorial traits. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED SCHEDULING OF PUBLIC TRANSPORT. *Proceeding….* Boston: MIT, ano1997. p. 81-102

[15] CASTRO, J. R. P. *Geração de escalas de trabalho em transporte urbano de passageiros*: *uma formulação heurística para a formulação do problema de cobertura de conjuntos.* Dissertação (Mestrado em Engenharia de Produção) — Universidade Federal de Santa Catarina, Florianópolis, 2002.

[16] FORSYTH, P.;WREN, A. An ant system for bus driver scheduling. In: International Workshop on Computer-Aided Scheduling *of* Public. *Proceeding*... Boston: MIT, 1997 p. 405-421.

[17] LOURENÇO, H. R., PAIXÃO, J.; PORTUGAL, R. Multiobjective metaheuristics for the bus-driver scheduling problem. *Transportation Science*, Atlanta, GA, v. 35, n. 3, p. 331-343, 2001.

[18] SHEN, Y.; e KWAN, R. S. K. Tabu search for driver scheduling. In: Yindong Shen, Raymond S. K. Kwan, *Computer-Aided Scheduling of Public Transport*, Berlin, Springer-Verlag, 2001, p. 121-135.

[19] FORES, S. *Column generation approaches to bus driver scheduling*. Tese (Doutorado em Ciência da Computação) — University of Leeds, Leeds, 1996.

[20] GIBBONS, A. *Algorithmic graph theory*. New York: Cambridge University, 1985.

[21] KIRKPATRICK, S.; GELATT JR., C. D.; VECCHI, M. P. Optimization by Simulated Annealing. *Science*, Washington, v. 220, n. 4598, p. 671-683, 1983.

[22] CERNY, V. Thermodynamical approach to the Traveling Salesman Problem. *Journal of Optimization Theory and Applications,* New York, US, 45, p. 41-51, 1985.

[23] JACOBS, L. W.; BRUSCO, M. J. A Local-Search Heuristic for Large Set-Covering Problems. *Naval Search Logistics,* Malden, v. 42, n. 7, p. 1129-1140, 1995.

[24] CASTRO, E. C.; CONSTANTINO, A. A. *Escalonamento de veículos e condutores em empresas de transporte urbano rodoviário*. Trabalho conclusão de curso (Graduação em Informática). Universidade Estadual de Maringá, Maringá, 2003.

[25] CONSTANTINO, A. A.; REIS, P. A. MENDONÇA, C. F. X.; FIGUEIREDO, M. F. Aplicação de algoritmos genéticos ao problema de cobertura de conjunto. –In: SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL, 35. *Anais...* Natal: Sobrapo, 2003.