META-HEURÍSTICA VNS APLICADA A UM PROBLEMA DE PLANEJAMENTO OPERACIONAL DE TRANSPORTE PÚBLICO

Rubens Zenko Sakiyama[1], Ademir Aparecido Constantino[2], Wesley Romão[3]

Departamento de Informática – Universidade Estadual de Maringá (UEM)
Avenida Colombo, 5790 – 87020-900 – Bloco C56 – PR – Brasil
[1]rubens.uem@gmail.com, [2]aaconstantino@uem.br, [3]wesley.uem@gmail.com

RESUMO

Este trabalho aborda um problema de escalonamento de motoristas de uma empresa de transporte público, baseado em um caso real de grande porte, o qual tem como objetivo formar jornadas diárias de trabalho que atenda a uma série de restrições e minimize o custo operacional. Este trabalho propõe um algoritmo heurístico baseado na meta-heurística VNS (*Variable Neighborhood Search*). Na fase de busca local são empregados dois procedimentos denominados PCR e K-*Swap*, baseados na resolução de sucessivos problemas de atribuição. Para validar o algoritmo foram realizados testes com instâncias de dados reais e aleatórias, incluindo uma instância real com mais de 2.300 viagens. Em geral, os resultados alcançados superaram os resultados encontrados na literatura.

PALAVRAS CHAVE. Problema de Escalonamento de Motoristas, *Variable Neighborhood Search*, Heurística, Problema de Atribuição.

Área principal: Logística e Transporte, Metaheurística

ABSTRACT

This paper tackles a bus-driver scheduling problem for a public transport company, based on large-scale real-would instances, which aims to construct a set of dairy work duties satisfying a set of constraints and to minimize the operational cost. This work proposes a VNS based heuristic algorithm to solve the problem trying to minimize the solution cost. The algorithm is based on solving successive assignment problem which is split in two phases. In local search phase two procedures are used, named PCR and K-swap, both of them solves successive assignment problems. To validate the algorithm, tests were performed using real and random instances, including real instances with more than 2300 trips. In general, the achieved results had lower costs than those used for comparison, showing the benefits of the presented techniques.

KEYWORDS. Drivers Scheduling Problem, Variable Neighborhood Search, Heuristic, Assignment Problem.

Main area: Logistics and Transport, Metaheuristics

1. INTRODUÇÃO

O transporte coletivo de passageiros é um serviço essencial para a sociedade, pois além de gerar empregos diretos e indiretos, grande parte da população depende deste transporte para se locomover diariamente até seu trabalho.

Os veículos e motoristas são os principais recursos de uma companhia de transporte público, e a forma como são alocados tem impacto direto na qualidade e no custo do serviço (Silva e Cunha, 2010). Além da redução direta de custos, há o benefício proporcionado aos funcionários, que poderiam cumprir jornadas menos exaustivas, aumentando a qualidade do serviço. Além disso, com jornadas de trabalho dentro da legislação evitam-se possíveis custos passivos (futuros) devido a ações trabalhistas (Calvi, 2005).

A importância de uma programação eficiente dos condutores está relacionada ao impacto desta nos custos totais de uma empresa de transporte coletivo, já que veículos e motoristas são os componentes que constituem a maior parte dos custos da empresa (Silva *et al*., 2002; Prata, 2010). De acordo com a URBS (Urbanização de Curitiba S/A), empresa de capital misto da qual a Prefeitura de Curitiba é a maior acionista, os custos com pessoal e encargos representam 47,16% da composição tarifária.

O problema do escalonamento de motoristas (PEM), também conhecido como problema de escalonamento de tripulação (PET) ou problema de programação de tripulação (PPT) é uma das etapas do planejamento do transporte público (Prata, 2010) e consiste em determinar a melhor programação diária dos condutores de forma a obedecer às restrições, minimizar custos operacionais e cumprir a escala de veículos para a tabela pré-determinada de horários.

Devido ao grande número de combinações possíveis para a resolução do PEM e às diversas restrições que devem ser consideradas, o problema é de difícil solução computacional e pertence à classe NP-Difícil (De Leone *et al*., 2011). O uso de métodos exatos ainda é um fator crítico para instâncias de grande porte.

Apesar de encontrarmos trabalhos que utilizam técnicas de programação matemática, as instâncias utilizadas nestes trabalhos não ultrapassam dezenas ou algumas centenas de viagens. Silva *et al*. (2004) apresentam uma metodologia que formula o PEM como um modelo de particionamento utilizando o método *Simplex* para resolvê-lo. Para validar o método proposto foram realizados testes com instâncias reais (11 linhas contendo de 11 a 87 viagens), resolvendo um problema de particionamento para cada linha separadamente, com o uso do pacote LINGO.

Yunes *et al*. (2005), utilizaram uma abordagem baseada no problema de cobertura de conjuntos e geração de colunas com o objetivo de minimizar o número de motoristas, utilizando duas instâncias reais, composta por duas linhas que atendem a região metropolitana de Belo Horizonte, sendo uma com 125 e outra com 246 viagens.

Na literatura encontramos diversas propostas de algoritmos heurísticos para o PEM. Marinho *et al*. (2004) apresentaram uma proposta baseada em Busca Tabu (BT) utilizando duas estruturas de vizinhança caracterizadas por dois tipos de movimentos. O primeiro movimento consiste em realocar uma tarefa de uma determinada jornada a outra jornada e o segundo em trocar tarefas entre duas jornadas. Silva *et al*. (2002) apresentam um algoritmo baseado na meta-heurística *Simulated Annealing* (SA). Souza *et al*. (2003) fazem uma continuidade de Silva *et al*. (2002), utilizando novas estruturas de vizinhança para o SA e para o Método VNS. Marinho *et al*. (2004) e Souza *et al*. (2003) utilizaram dados reais de empresas de transporte público da cidade de Belo Horizonte.

Lourenço *et al*.(2001) propõem meta-heurísticas multiobjetivos usando Busca Tabu (BT) Multiobjetivo e Algoritmos Genéticos (AG) Multiobjetivo, combinados com GRASP (*Greedy Randomized Adaptive Search Procedure*) com a utilização de instâncias reais de seis empresas de transporte público de Portugal, participantes de um projeto de planejamento de transporte denominado GIST. Dias *et al*. (2002) propõem um Algoritmo Genético que trata custos, número de jornadas, porcentagem da escala realizada por jornadas viáveis e duração média das jornadas como características importantes para obter uma solução de qualidade, utilizando 15 instâncias de

duas maiores empresas de transporte urbano portuguesas sendo a menor com 102 viagens e a maior com 251 viagens.

Mauri e Lorena (2004) propõem uma metodologia interativa baseada na aplicação do Algoritmo de Treinamento Populacional (ATP) juntamente com Programação Linear (PL) com instâncias de 25 a 500 viagens geradas aleatoriamente a partir de instâncias reais de empresas de transporte público brasileiras. Para a resolução de instâncias reais do PEM entre 19 e 138 viagens, obtidas de empresa de transporte público da cidade de Belo Horizonte, Santos (2007) apresentou um método que utiliza o Algoritmo Genético (AG) para a resolução dos subproblemas de geração de novas colunas, resolvidos por PL ou por PLI.

De Leone *et al*. (2010) propõem um algoritmo baseado na heurística GRASP para instâncias entre 80 e 400 viagens considerando as restrições seguidas por empresas de transporte italianas. Silva e Cunha (2010) também apresentaram um modelo de resolução do PEM baseado na heurística GRASP com busca local realizada pela técnica de Busca em Vizinhança de Grande Porte (BVGP) utilizando escalas de programação de veículos de uma empresa de transporte público de Belo Horizonte com um mínimo de 392 e um máximo de 945 viagens. Gonçalves (2010) apresentou uma solução baseada na utilização das meta-heurísticas Busca Tabu e *Iterated Local Search* (ILS) com dados reais de uma empresa de transporte público da cidade de Belo Horizonte com número de viagens entre 27 (1 linha) e 515 (13 linhas).

Uma abordagem integrada do problema de escalonamento de veículos (PEV) e de motoristas (PEM) para diversas garagens também é apresentada por De Groot e Huisman (2004) e Huisman *et al*. (2005), no qual são propostos dois diferentes modelos de algoritmos baseados em uma combinação de técnicas de geração de colunas com relaxação Lagrangeana, resolvendo instâncias com 194 a 653 viagens, obtidas de dados reais de empresas de transporte público da Holanda. Laurent e Hao (2008) resolveram um problema PEV + PEM de forma integrada e sequencial utilizando a meta-heurística GRASP com instâncias reais com até 249 viagens.

Calvi (2005) propõe um novo algoritmo heurístico para a resolução do PEM baseado na resolução de sucessivos Problemas de Designação, também conhecido como Problema de Atribuição, que surgem de um modelo baseado em grafo multipartido. O presente trabalho apresenta uma extensão dessa metodologia, apresentando um novo procedimento de busca local chamado k-swap e investiga sua combinação com a meta-heurística VNS (*Variable Neighborhood Search*).

Este artigo está dividido em 4 seções. A Seção 1 apresenta uma introdução ao problema estudado e faz uma revisão de trabalhos relacionados da literatura. A Seção 2 apresenta o algoritmo proposto. Na Seção 3 são apresentados os resultados computacionais. Por fim, as conclusões são apresentadas na Seção 4.

2. Algoritmo Proposto

Para a resolução do Problema de Escalonamento de Motoristas é proposto um algoritmo heurístico baseado na meta-heurística VNS. Nesta seção é apresentada a fase construção da solução inicial e na sequência são apresentados dois algoritmos de busca local, um denominado PCR (Procedimento de Cortes e Recombinações) e o segundo de K-*Swap*.

2.1 Problema de Atribuição

O Problema de Atribuição (PA), também encontrado na literatura como Problema de Designação ou *Assignment Problem*, é um clássico problema de Otimização Combinatória em Pesquisa Operacional. De acordo com Hillier e Lieberman (2010), é um tipo especial de problema de programação linear no qual os designados são indicados para a realização de tarefas, que em nosso trabalho são as viagens realizadas pelas tripulações.

A solução do PA equivale ao emparelhamento perfeito de custo mínimo em um grafo bipartido. Assim, dada uma matriz de custos de dimensões $n \times n$, o problema consiste em associar cada linha i a uma coluna j sob um custo c_{ij}, de modo que a soma dos custos seja a menor possível. O problema pode ser formulado da seguinte maneira:

$$\text{Minimizar} \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} \,.\, x_{ij}; \tag{1}$$

$$\text{Sujeito a:} \quad \sum_{i=1}^{n} x_{ij} = 1; \qquad j = 1,...,n; \tag{2}$$

$$\sum_{j=1}^{n} x_{ij} = 1; \qquad i = 1,...,n; \tag{3}$$

$$x_{ij} \in \{0,1\}; \qquad i = 1,...,n;\ j = 1,...,n. \tag{4}$$

A função-objetivo (1) minimiza o custo da soma das atribuições entre linhas e colunas; a restrição (2) exige que para cada linha haja uma coluna associada; a restrição (3) garante que para cada coluna seja designada uma linha; a restrição (4) garante que as variáveis envolvidas assumam apenas os valores de decisão 0 e 1.

Como será detalhado nas próximas seções, o problema de designação tem diferentes interpretações para sua matriz de custo. Embora seja um problema que tenha solução em tempo polinomial isso não significa que o presente problema PEM tenha solução em tempo polinomial, pois o problema designação é apenas um subproblema que é resolvido pelo algoritmo heurístico proposto.

2.2 Construção da solução inicial

Inicialmente distribuímos todas as viagens em subgrupos de viagens, denominado camadas, de forma que cada camada contenha apenas viagens que não podem ser designadas em sequência, conforme algoritmo de geração de camadas ilustrado na Figura 1.

Algoritmo GeraCamadas

Entrada: o conjunto de tarefas *T*.
Saída: o conjunto de camadas.
Seq(*TP*, T_k) = V (*verdadeiro*) se uma tarefa *TP* pode ser sequenciada com alguma tarefa da camada T_k, caso contrário, *Seq*(*TP*, T_k) = F (*falso*).
MaisCedo(*T*): função que devolve a tarefa que inicia mais cedo em *T*.
CriarCamada: função que cria uma nova camada vazia.

```
Begin
    k=1;
        T_k = CriarCamada; //Cria a primeira camada
    While T≠ø (vazio) do
            TP= MaisCedo(T);
            T=T-{TP};
            i=0;
            Repeat  //procura por uma camada existente para inserir TP.
                    i=i+1;
            Until (i>k) or  Not Seq(TP, T_i)
            If  i>k then
                    k=i;  //nova camada é criada
                    T_k = CriarCamada;
            T_i=T_i + {TP};
End;
```

Figura 1: Algoritmo de geração de camadas

Essas camadas são utilizadas durante toda a resolução, onde serão designadas as viagens para cada motorista na formação de sua jornada (sequência de viagens realizadas por uma mesma tripulação) de trabalho e realizadas as recombinações entre partes ou pedaços de jornadas na fase de melhoramento, através da resolução de PAs.

A construção da solução inicial consiste em gerar um grafo multipartido através da resolução de sucessivos PAs, no qual cada jornada de trabalho corresponde a um caminho da primeira a última camada. Em cada camada é criada uma matriz de custos quadrada $C^k = [c_{ij}^k]$ de ordem n_k, onde n_k corresponde a soma do total de jornadas já criadas, n_jor, com o total de viagens que devem ser designadas na camada k, v_k.

Na criação da matriz de custos é necessária a inclusão de viagens e/ou jornadas fictícias para que a matriz se torne quadrada, com o número de linhas igual ao número de colunas, o que é um requisito para resolver o PA, conforme alerta Pentico (2007). O algoritmo inicia com o valor de n_jor igual a zero e conforme as viagens vão sendo alocadas o valor de n_jor vai sendo incrementado conforme crescem o número de jornadas.

A Figura 2 ilustra a matriz de custos, formada por quatro blocos, com diferentes critérios de custos, conforme descrito a seguir:

	Viagens	Viagens Fictícias
Jornadas	Bloco 1 $c_{ij}^k = f(i,j)$ ou $c_{ij}^k = \infty$	Bloco 2 $c_{ij}^k = f(i,j) + CF$
Jornadas Fictícias	Bloco 3 $c_{ij}^k = CNJ$	Bloco 4 $c_{ij}^k = 0$

Figura 2: Estrutura da matriz de custos na construção da solução inicial

i. Bloco 1: caso a atribuição seja viável, a função $f(i,j)$ retorna o custo de atribuir a viagem j à jornada i, caso contrário a função retorna um custo infinito;
ii. Bloco 2: a função $f(i,j)$ retorna o custo de atribuir um intervalo de folga ou um tempo ocioso à jornada i. Além disso, é acrescido um custo CF que torna a atribuição de viagens fictícias menos atrativas.
iii. Bloco 3:. A função deste bloco é permitir que novas jornadas sejam criadas quando as viagens não puderem ser designadas às jornadas já existentes. Este bloco recebe um custo CNJ, cujo valor é o custo de uma nova jornada, considerando o tempo mínimo pago;
iv. Bloco 4: formado pelas viagens fictícias e pelas jornadas fictícias, sua função é apenas completar a matriz. Assim, este bloco recebe custo zero.

Na criação da matriz de custos foi utilizada a f, para atribuir o custo de uma viagem j a uma jornada i:

$$f(i,j) = CM(i,j) + CO(i,j) + Pen.nRV \quad (5)$$

Na equação (5), $CM(i,j)$ é o custo em minutos trabalhados, $CO(i,j)$ é o custo em minutos ociosos, nRV é o número de restrições violadas nessa atribuição e Pen é o custo em minutos de penalidade devido às violações de restrições. Dentre as penalidades aplicadas, destacamos a penalidade por troca de veículos, por ultrapassar o limite máximo de jornada de treze horas e jornada contínua de seis horas, por jornada possuir horas extras e ultrapassar o máximo de duas horas extras.

2.3 Fase de Melhoramento

A fase de melhoramento é composta por dois procedimentos, o Procedimento de Cortes e Recombinações (PCR) e o K-*Swap*, que são utilizados para minimizar o custo total da solução inicial. O PCR busca reduzir o custo total da solução através da troca de pedaços entre jornadas.

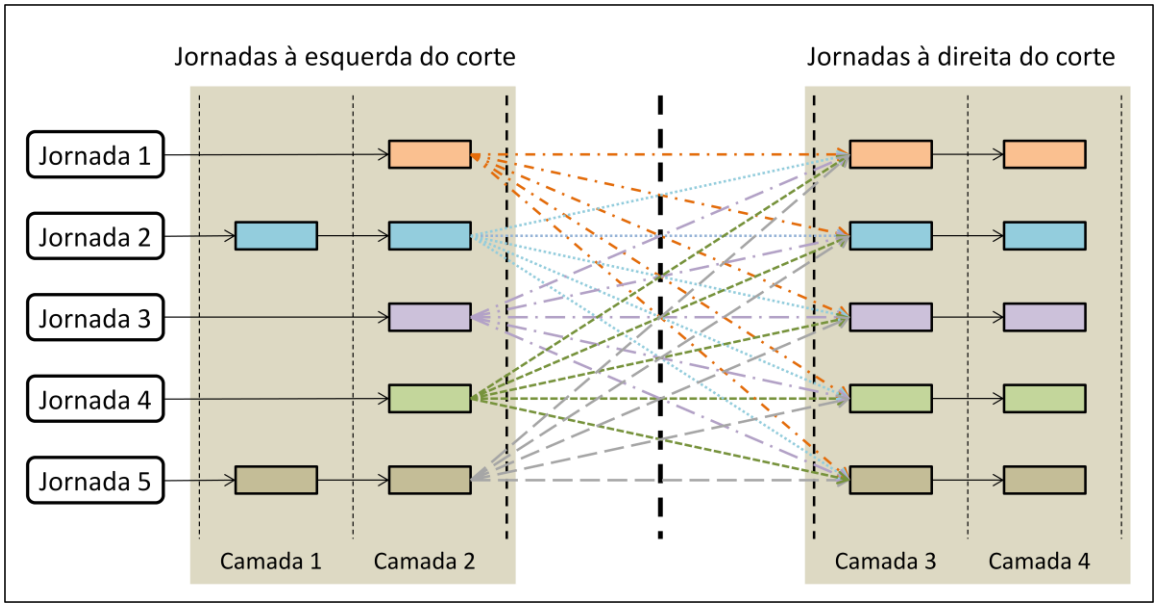

Figura 3: Exemplo de possíveis recombinações do PCR ao resolver a camada 2

Para isso, realiza um corte que divide cada uma das n jornadas em duas jornadas parciais, uma à esquerda e outra à direita do corte, conforme ilustra na Figura 3, o corte realizado entre as camadas 2 e 3.

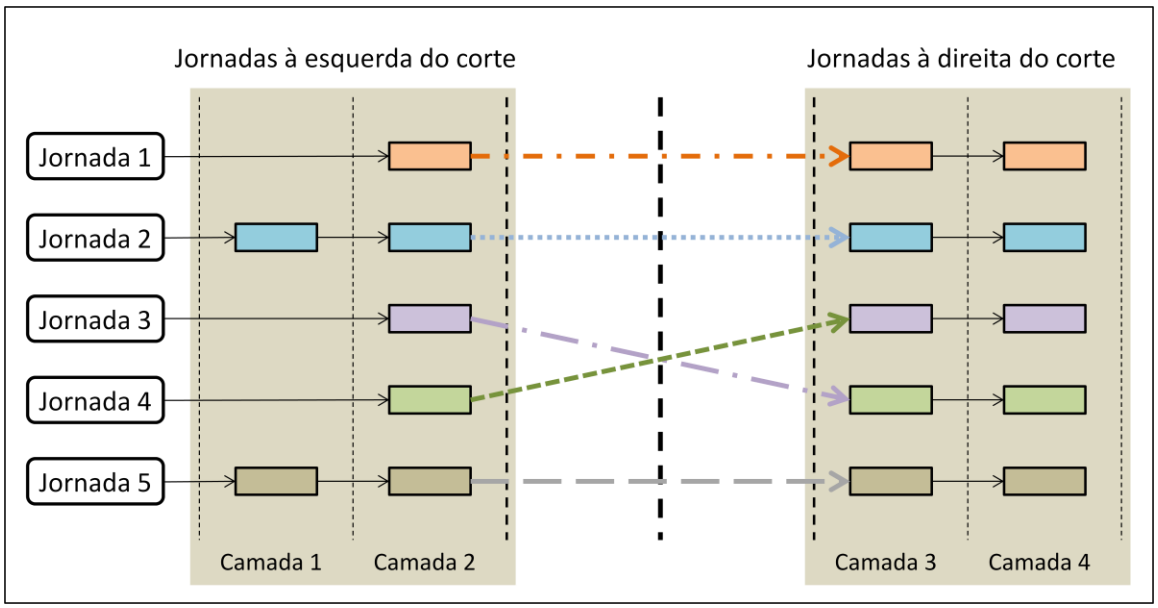

Figura 4: Exemplo de recombinação após execução do PCR na camada 2

Após isso, é calculado o custo de se associar cada um dos n trechos de jornada à esquerda com cada um dos n trechos à direita do corte. Assim, é construída a matriz de custos $D^k = [d_{ij}^k]$, de dimensões $n\ x\ n$, onde n corresponde ao número total de jornadas e o valor de d_{ij}^k

é dado por $g(i,j)$, que corresponde ao custo de associar a jornada à esquerda i com a jornada à direita j, caso possível, mais as possíveis penalidades. Caso essa associação seja inviável, d_{ij}^k recebe custo infinito.

De posse da matriz de custos D^k, é resolvido o PA para a camada correspondente e com base na solução é realizada a recombinação dos trechos, podendo acarretar na formação de novas jornadas.

Uma iteração completa do procedimento consiste em realizar cortes e recombinações antes de todas as camadas do problema, exceto antes da primeira camada por não resultar em nenhuma alteração da solução. A Figura 4 exemplifica como poderia ser realizada a recombinação no PCR. Note que houve uma troca entre as jornadas 3 e 4, permitindo reduzir a duração da Jornada 3 ou da Jornada 4.

O segundo procedimento de melhoramento, o K-*Swap*, é um procedimento de trocas variáveis, e consiste em selecionar K camadas consecutivas e verificar a possibilidade de realizar trocas desses blocos de viagens entre todas as jornadas. Ou seja, o K-*Swap* busca minimizar o custo total da solução através da redistribuição de blocos de viagens entre os motoristas durante K camadas.

O K-*Swap* realiza um corte entre duas camadas, como no PCR, depois percorre K camadas e realiza um novo corte entre camadas. O trecho entre os dois cortes, denominado bloco de viagens, é a parte que poderá ser trocada entre as jornadas. De posse da variável K, é calculado o custo de se associar cada um dos n blocos de viagens com cada uma das jornadas. Assim, na matriz $E^k = [e_{ij}^k]$, referente ao procedimento K-*Swap*, e_{ij}^k é dado por $h(i,j)$, que equivale ao custo de associar o bloco de viagens j com a jornada i, caso possível, incluindo possíveis penalidades. Se a associação for inviável, e_{ij}^k recebe custo infinito.

Na Figura 5 estão representadas, em linhas pontilhadas, as possibilidades de recombinação do 2-*Swap*, K-*Swap* tomando K igual a 2, ao resolver a Camada 2.

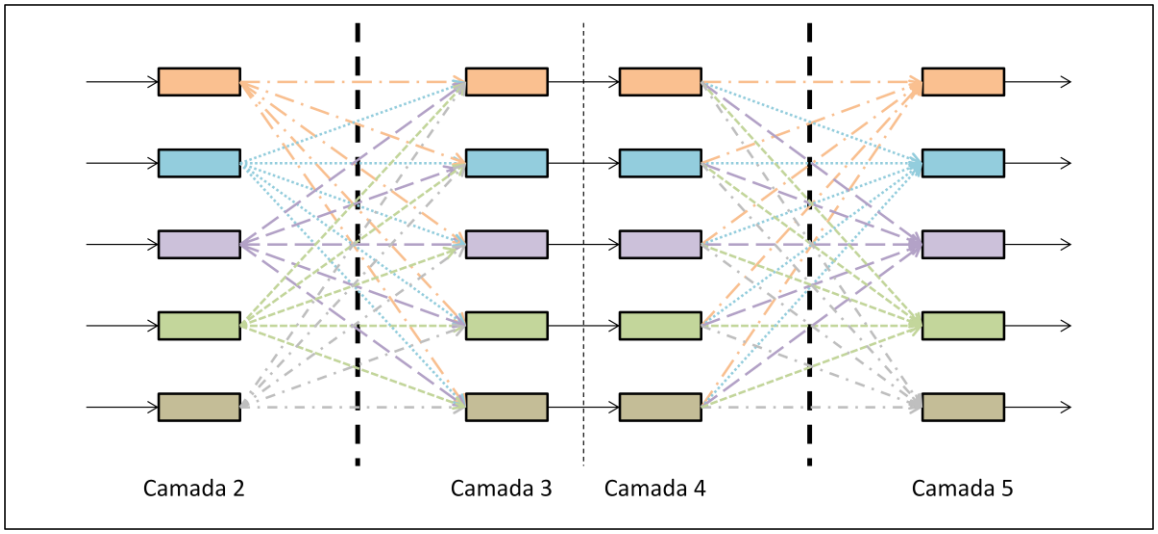

Figura 5: Exemplo de possíveis recombinações do 2-Swap ao resolver camada 2

Construída a matriz de custos E^k, o PA para a camada correspondente é resolvido e com base na solução é feita a recombinação dos blocos de viagens, que no exemplo contém duas viagens devido ao fato de K ser igual a 2. Um exemplo dessa recombinação é apresentado na Figura 6, onde foi realizada uma troca entre os trechos das Jornadas 1 e 2, que gerou um custo de solução menor.

Note que o valor de K pode assumir diferentes valores. Para o presente trabalho foram utilizados os seguintes valores para K=1, 2, 3 e 4, assim temos os seguintes

procedimentos: 1-*swap*, 2-*swap*, 3-*swap*, 4-*swap*. Isso significa que o procedimento K-*swap* resulta em 4 estruturas de vizinhas diferentes.

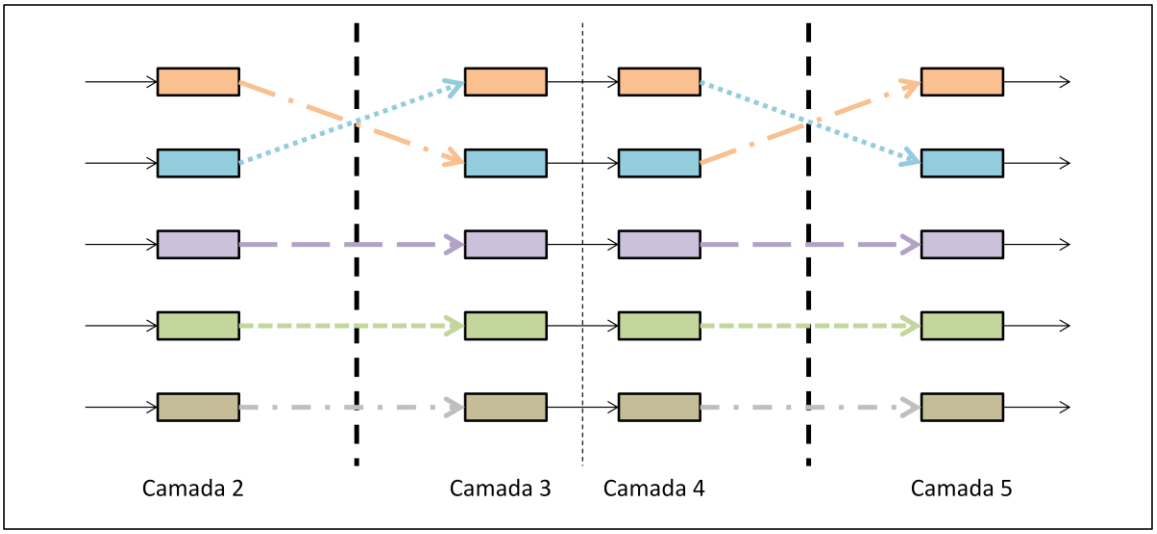

Figura 6: Exemplo de recombinação após execução do 2-Swap na camada 2

2.4 Meta-heurística VNS

A meta-heurística VNS (*Variable Niegboorhood Search* ou Método de Pesquisa em Vizinhança Variável) (Hansen *et al.*, 2010) é um método de uso geral para resolução de problemas de otimização combinatória. É um método de busca local que consiste em explorar o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança com o objetivo de escapar de mínimos locais.

Diferente de outras meta-heurísticas baseadas em métodos de busca local, o método VNS não segue uma trajetória, mas sim explora vizinhanças gradativamente mais distantes da solução corrente e focaliza a busca em torno de uma nova solução se, e somente se, um movimento de melhora é realizado.

Uma versão básica do método VNS é apresentada no pseudocódigo ilustrado na 7, onde *S* representa a solução corrente, z_{max} a quantidade máxima de estruturas de vizinhanças.

Para evitar ciclos, antes de aplicar uma busca local, soluções são selecionadas aleatoriamente pelo procedimento *Shake* (linha 3), que possui como parâmetros a solução corrente e a estrutura de vizinhança selecionada. O método inclui, também, um procedimento de busca local (*Local Search* - linha 4) a ser aplicado sobre soluções vizinhas selecionadas. Esta rotina de busca local também pode usar diferentes estruturas de vizinhança, como por exemplo, aplicar a meta-heurística VND.

```
        VNS(S, z_max);
1       z ← 1
2       repeat
3               S´ ← Shake(S,z)
4               S´´ ← Local Search(S´)
5               if f(S´´) < f(S) then
6                       S ← S´´; z ← 1;
                else
7                       z ← z + 1;
        until z = z_max;
```

Figura 7: Pseudocódigo VNS

O procedimento *Shake* consiste em executar um movimento de modo aleatório para encontrar um vizinho da solução corrente e estabelecer uma nova estrutura de vizinhança.

```
        Shake(S)
        Início
2       S´ = Rd(S);
3       z ← 1;
5       repita
6               caso <z> igual a
7               1: S´ ← 1-swap (S);
8               2: S´ ← 2-swap(S´);
9               3: S´ ← 3-swap(S´);
10              4: S´ ← 4-swap(S´);
11              5: S´ ← PCR(S´);
                fimcaso
12              se (f(S´) < f(S)) então
13                      S ← S´;
14                      z ← 1;
                senão
16                      z ← z + 1;
                fimse
        até (z>5)
        fim
```

Figura 8: Pseudocódigo para *Shake*

A partir da escala de jornadas da solução corrente é selecionada aleatoriamente uma camada onde é realizado um corte. Nesta camada de corte, recombina-se as jornadas à esquerda do corte com jornadas à direita do corte, também selecionadas aleatoriamente, em 10% das jornadas da escala.

O procedimento *Shake* considera *S* a solução corrente e Rd(*S*) uma função que seleciona aleatoriamente uma camada na escala de jornadas e realiza recombinações em 10% das jornadas selecionadas também aleatoriamente. O procedimento é ilustrado pela figura 8. Todos os procedimentos de busca local utilizam a estratégia da escolha do melhor vizinho (*best improviment*) percorrendo as camadas do fim para o início.

Já procedimento *Local Search* foi implementado utilizando somente os procedimentos 1-*swap* e PCR com a estratégia de escolha do primeiro melhor (*first improvement*).

3. Resultados

3.1 Considerações iniciais

Os algoritmos foram implementados em Pascal, no ambiente Lazarus V.1.0.12, e os experimentos realizados em dois ambientes. Os experimentos das implementações dos algoritmos determinísticos foram realizados em um microcomputador pessoal com processador *Intel Core* i5 M480 com clock de 2.67 GHz, 4GB de RAM e sistema operacional *Windows* 7. Os experimentos das implementações dos algoritmos com a meta-heurística VNS foram realizados no *Windows Server* 2008-R2, rodando em uma máquina virtual *KVM*, configurado para ocupar 30GB de RAM e 50 núcleos de um servidor com 4 processadores *Intel Xeon* E7-4860 (24MB de *cache* – 2.26 GHz) com sistema operacional *Linux CentOS* 6.

Foram utilizadas duas instâncias de dados reais, fornecidas por duas empresas de transporte urbano de passageiros do estado do Paraná, e oito instâncias geradas aleatoriamente a partir de uma instância real. Nestas instâncias, a escala de veículos já está definida e segue o planejamento das empresas.

3.2 Instâncias utilizadas

Para validação do algoritmo proposto foram utilizadas as mesmas instâncias utilizadas em Calvi (2005) e Rizzato *et al.* (2012). O conjunto de instâncias é composto por duas das instâncias reais e oito instâncias aleatórias geradas por Calvi (2005) baseadas em uma instância real.

Tabela 1: Instâncias utilizadas

Instância	Número de viagens	Tipo
AL130	130	Gerada aleatoriamente
AL251	251	Gerada aleatoriamente
RE412	412	Problema real
AL512	512	Gerada aleatoriamente
AL761	761	Gerada aleatoriamente
AL1000	1000	Gerada aleatoriamente
AL1253	1253	Gerada aleatoriamente
AL1512	1512	Gerada aleatoriamente
AL2010	2010	Gerada aleatoriamente
RE2313	2313	Problema real

As instâncias são descritas abaixo e representadas na tabela, tendo como o prefixo (RE ou AL) para indicar se é uma instância real ou aleatória e a parte numérica indica o numero de viagens (*e.g.* RE412 é uma instância real e contém 412 tarefas):

- RE412 e RE2313: correspondem aos dados reais de duas empresas de transporte urbano de passageiros do Estado do Paraná;
- AL130, AL251, AL512, AL761, AL1000, AL1253, AL1517 e AL2010: são instâncias aleatórias obtidas a partir da instância RE2313 sorteando-se *blocos* da mesma.

Analisando o tamanho das instâncias dos trabalhos relacionados na seção 1, observa-se que a maior instância foi utilizada por Silva e Cunha (2010) com 945 viagens. Do conjunto de instâncias utilizadas por este trabalho, metade delas contém um número de viagens superior a este valor, sendo que a maior instância possui 2,44 vezes o número de viagens utilizado por Silva e Cunha (2010).

3.3 Resultados obtidos

A Tabela 2 ilustra que os resultados obtidos por este trabalho apresentaram uma redução no número de jornadas da escala assim como uma redução no custo total das jornadas comparado aos resultados obtidos por Rizzato el al. (2012) e Calvi (2005) para todas as instâncias.

Tabela 2: Comparação com resultados de Rizzato e Calvi

Instância	Melhores resultados deste trabalho		Melhores resultados Rizzato (2012)		Melhores resultados Calvi (2005)	
	N.Jor.	Custo	N.Jor.	Custo	N.Jor.	Custo
AL130	**18**	**8.162,10**	19	8.400,93	19	8.389,40
AL251	**37**	**16.475,00**	39	17.190,00	40	17.600,00
RE412	**62**	**28.370,58**	65	29.260,00	66	29.512,50
AL512	**72**	**32.445,00**	76	33.822,50	79	35.105,00
AL761	**102**	**46.137,83**	108	47.932,29	107	47.532,90
AL1000	**140**	**62.553,25**	144	64.097,29	146	64.873,60
AL1253	**180**	**80.257,50**	182	81.047,50	187	82.842,90
AL1517	**220**	**97.895,00**	221	98.283,21	225	99.852,80
AL2010	**281**	**125.462,50**	285	126.747,71	290	128.964,20
RE2313	**315**	**141.343,65**	327	145.701,43	331	147.215,00

A tabela 3 ilustra os *gaps* dos melhores resultados obtidos neste trabalho em relação aos melhores resultados obtidos por Rizzato *el al.* (2012) e Calvi (2005) quanto ao custo da escala de jornadas. *Gaps* negativos significam uma redução nos custos e *gaps* positivos significam um aumento nos custos. O maior *gap* encontrado em relação aos trabalhos citados foi obtido com a instância AL512 e o menor com a instância AL1517.

Tabela 3: *Gap* para os resultados de Rizzato e Calvi

Instância	Melhores resultados deste trabalho	Melhores resultados Rizzato (2012)	*Gap* Rizzato	Melhores resultados Calvi (2005)	*Gap* Calvi
AL130	**8.162,10**	8.400,93	-2,84%	8.389,40	-2,71%
AL251	**16.475,00**	17.190,00	**-4,16%**	17.600,00	-6,39%
RE412	**28.370,58**	29.260,00	-3,04%	29.512,50	-3,87%
AL512	**32.445,00**	33.822,50	-4,07%	35.105,00	**-7,58%**
AL761	**46.137,83**	47.932,29	-3,74%	47.532,90	-2,93%
AL1000	**62.553,25**	64.097,29	-2,41%	64.873,60	-3,58%
AL1253	**80.257,50**	81.047,50	-0,97%	82.842,90	-3,12%
AL1517	**97.895,00**	98.283,21	-0,39%	99.852,80	-1,96%
AL2010	**125.462,50**	126.747,71	-1,01%	128.964,20	-2,72%
RE2313	**141.343,65**	145.701,43	-2,99%	147.215,00	-3,99%

4. Conclusões

Este trabalho apresentou uma proposta para a solução do problema de escalonamento de motoristas (PEM), utilizando a meta heurística VNS combinada com estruturas de vizinhanças denominadas de PCR e *k-swap*. Os experimentos computacionais foram realizados com dados da literatura, incluindo dados de instâncias reais de grande porte. Os resultados alcançados foram superiores a todos os resultados da literatura, mostrando, portanto, a grande potencialidade da metodologia utilizada.

Como continuidade de pesquisas relativas a este trabalho, sugere-se a ampliação da utilização do VNS com outras combinações dos procedimentos PCR e *k-swap*, aplicadas tanto para o procedimento *Shake* como para o *Local Search*.

Agradecimentos: Agradecemos a CAPES e ao CNPq pelo suporte financeiro que apoiaram o desenvolvimento deste trabalho

Referências

Calvi, R., *Um Algoritmo para o Problema de Escalonamento de Tripulação em Empresas de Ônibus.* 2005. Dissertação de Mestrado. Universidade Estadual de Maringá, Maringá.

De Leone, R., Festa, P. e Marchitto, E. (2010), A Bus Driver Scheduling Problem: a new mathematical model and a GRASP approximate solution. *Journal of Heuristics*, vol. 17, n. 4, p. 441-466.

De Groot S, Huisman D. (2004), Vehicle and crew scheduling: solving large real-world instances with an integrated approach. *Econometric Institute Report* EI2004-13, Erasmus University, Rotterdam, The Netherlands; p. 43-56.

Dias, T. G., Souza, J. P. e Cunha, J. F. (2002), Genetic algorithms for the bus driver scheduling problem: a case study. *Journal of the Operational Research Society*, 53, p. 1-12,

Gonçalves, T. L., *Meta-heurísticas para o Problema de Programação de Tripulações.* 2010. Dissertação de Mestrado. Universidade Federal do Rio de Janeiro, Rio de Janeiro.

Hansen, P., Mladenovic, N. e Pérez, J. M.(2010) Variable neighbourhood search: methods and applications. *Annals of Operational Research*, v. 175, n. 1, p. 367-407.
Hillier, F. S. e Lieberman, G. J., *Introdução à Pesquisa Operacional*. 8. Ed. McGraw-Hill, Porto Alegre, 2010, p. 852.
Huisman, D., Freling, R. e Wagelmans, A. P. M. (2005) Multiple-Depot Integrated Vehicle and Crew Scheduling. *Transportation Science*, v. 39, n. 4, p. 491–502.
Laurent, B. e Hao, J. K. (2008) Simultaneous Vehicle and Crew Scheduling for Extra Urban Transports. *Lecture Notes in Artificial Intelligence* 5027: 466–475.
Lourenço, H. R., Paixão, J. P. e Portugal, R. (2001), Multiobjective Metaheuristics for the Bus Driver Scheduling Problem. *Transportation Science*, v. 35, n. 3, p. 331-343.
Marinho, E. H., Ochi, L. S., Drummond, M. A., Souza, M. J. F. e Silva, G. P., (2004), Busca Tabu Aplicada ao Problema de Programação de Tripulações de Ônibus Urbano. *Anais do XXXVI SBPO*, p. 1471-1482.
Mauri, G. R. e Lorena, L. A. N. (2004), Método Interativo para Resolução do Problema de Escalonamento de Tripulações. Anais do XXXVI SBPO.
Pentico, D. W. (2007), Assignment Problems: A golden anniversary survey. *European Journal of Operational Research*, vol. 176, p. 774 – 793.
Prata, B. A., Programação Integrada de veículos e motoristas: uma visão geral. *Revista Eletrônica Sistemas e Gestão*, v. 4, n. 3, p. 182 – 204, set./dez. 2010.
Rizzato, D**. B., Sakiyama, R. Z., Constantino, A. A., e Romão, W.** (2012). Comparação de Algoritmos Heurísticos para um Problema de Planejamento Operacional de Transporte Público. *Anais: XVI CLAIO/ XLIV Simpósio Brasileiro de Pesquisa Operacional*, p. 1856-1867.
Santos, A. G. e Mateus, G. R. (2007), Crew Scheduling Urban Problem: an Exact Column Generation Approach Improved by a Genetic Algorithm. *IEEE Congress on Evolutionary Computation*, p. 1725–1731.
Silva, G. P. e Cunha, C. B. (2010), Uso da técnica de busca em vizinhança de grande porte para a programação da escala de motoristas de ônibus urbano. *Revista Transportes*, v. 8 n. 2 p. 37-45.
Silva, G. P., Souza, M. J. F. e Alves, J. M. C. B. (2002), *Simulated Annealing* Aplicado à Programação da Tripulação no Sistema de Transporte Público. In *XXII Encontro Nacional de Engenharia de Produção*.
Silva, G. P., Souza, M. J. F. e Reis, J. A., (2004), Um método exato para otimizar a escala de motoristas e cobradores do sistema de transporte público. *Anais do XVIII Congresso de Pesquisa e Ensino em Transportes*, p. 340-346.
Souza, M. J. F., Rodrigues, M. M. S., Mapa, S. M. S. e Silva, G. P. (2003), Um estudo das heurísticas Simulated Annealing e VNS aplicadas ao problema de programação de tripulações. *XXIII Entro Nacional de Engenharia de Produção*, 2003.
Yunes, T.H., Moura, A. V., Souza, C.C., 2005. Hybrid Column Generation Approaches for Urban Transit Crew Management Problems. *Transportantion Science*, 39(2), 273-288.