ARTIGO

UM ALGORITMO DE GERAÇÃO DE ARCOS PARA O PROBLEMA DE PROGRAMAÇÃO DE VEÍCULOS

Gustavo Peixoto Silva
Nicolau D. Fares Gualda
Departamento de Engenharia de Transportes
Escola Politécnica da Universidade de São Paulo

RESUMO

Este trabalho retrata a continuidade de uma pesquisa de doutorado voltada para a utilização de modelos de fluxo em redes na resolução de problemas de programação de veículos (de ônibus, em particular). Nesta etapa do trabalho foi aplicada a técnica de Geração de Arcos ao modelo de pseudo-designação para se obter uma representação eficiente de problemas reais, apoiada na utilização do algoritmo out-of-kilter para otimização de fluxo em redes. Com esta abordagem foi possível reduzir o número de arcos da rede, reduzir o tempo de resolução e ainda incluir restrições práticas, tornando o modelo mais adequado à realidade.

A metodologia apresentada foi testada em problemas reais das cidades de Reading – Reino Unido e Sorocaba – Brasil, e os resultados foram comparados com aqueles produzidos pelo sistema heurístico BOOST, desenvolvido pelo grupo de programação de veículos e tripulação da Universidade de Leeds no Reino Unido.

ABSTRACT

This paper is part of a doctoral research which utilises network flow models to solve vehicle scheduling problems(bus, in particular). This stage presents an application of the Arc Generation approach to the quasi-assignment model to obtain an efficient representation for practical problems, with utilization of the out-of-kilter algorithm for

optimization of flows in networks. With this technique it is possible to reduce the number of arcs in the network, to reduce the CPU time to solve the problem and also to introduce some practical side constraints, making the network flow representation more compatible with real cases.

The presented methodology was tested with data from the city of Reading in the UK and from the city of Sorocaba in Brazil, and the results were compared with those provided by the heuristic system BOOST, developed by the vehicle and crew scheduling group, University of Leeds, United Kingdom.

1. INTRODUÇÃO

As recentes privatizações no sistema de transporte público têm forçado as companhias de ônibus a utilizarem de maneira eficiente seus recursos materiais e humanos para que sejam lucrativas sem que haja um comprometimento na qualidade do serviço oferecido. Neste sentido todo o processo de planejamento das operações de ônibus deve ser revisado e otimizado na medida do possível.

O planejamento das operações de ônibus é um trabalho complexo e por isso mesmo é dividido em várias etapas, onde os dados de saída de uma etapa são utilizados como dados de entrada de etapas posteriores. As principais fases deste processo são: a) a definição das rotas, b) a definição das tabelas de horários, c) a definição das programações dos veículos e d) a definição dos turnos de trabalhos das tripulações.

Entre as etapas do planejamento da operação, a definição das rotas e a definição das tabelas de horários normalmente são feitas pelos órgãos de gerência do transporte público, restando aos operadores definir a programação dos veículos e os turnos de trabalho das tripulações. Logo, é nas duas etapas finais do processo que as empresas de transporte público podem aprimorar suas atividades, tornando-se mais eficientes.

Este trabalho aborda a etapa do planejamento relacionada às programações dos veículos. Essas programações devem ser definidas de tal maneira que todas as viagens nas tabelas de horários sejam executadas uma única vez. O problema consiste em encontrar uma programação que utilize o menor número de veículos e cujo custo operacional seja mínimo. Este problema é conhecido como o Problema de Programação de Veículos (Vehicle Scheduling Problem). Vários métodos de resolução são apresentados na literatura e os autores mostram que há uma redução significativa no custo operacional quando métodos de otimização ou métodos heurísticos são aplicados aos casos reais (Wren e Kwan 1999, Wren e Gualda 1999, Löbel 1999, Smith e Wren 1981, Gavish et al. 1978, Wren 1972).

Tentativas de utilização de modelos de fluxo em redes para resolver o problema de programação de veículos são retratadas nos trabalhos de Bodin et al. (1983), de Carraresi e Gallo (1984) e de Daduna e Paixão (1995). Isto se deve pela eficiência de tais modelos em representar o problema e na simplicidade de implementação dos algoritmos de fluxo em redes, os quais na sua maioria apresentam complexidade polinomial. Além disso, uma vez consideradas apenas unidades de fluxo inteiras, a solução ótima será necessariamente inteira, evitando problemas com erros de arredondamento numérico. No entanto, há carência de exploração de técnicas de redução do número de arcos normalmente gerados em problemas deste tipo, para diminuir o tamanho da rede decorrente, de forma a tornar o problema exequível e a modelagem mais eficiente.

Uma primeira exploração de técnicas de redução do número de arcos foi apresentada em Silva et al. (1998). Dando continuidade à pesquisa, o presente trabalho apresenta e explora uma metodologia para resolver problemas reais de programação de veículos com uma única garagem, utilizando o modelo de pseudo-designação proposto por Paixão e Branco (1987) e aplicando a técnica de Geração de Arcos para resolvê-lo de maneira eficiente, com base no proposto por Freling et al. (1995).

A metodologia foi testada com problemas reais das cidades de Reading – Reino Unido e de Sorocaba – Brasil e seus resultados

comparados com aqueles fornecidos pelo sistema heurístico BOOST (Kwan e Rahin (1999)). A comparação dos resultados tem a finalidade de verificar a qualidade das soluções obtidas através do modelo matemático em relação a um sistema heurístico de referência na área de programação de veículos.

Este trabalho está dividido da seguinte maneira: nas seções 2 e 3 são apresentados os modelos básico e de pseudo-designação para o problema, juntamente com a técnica de Geração de Arcos que reduz o tamanho da rede. A seção 4 descreve rapidamente o sistema BOOST, a seção 5 apresenta e discute os resultados dos testes realizados e a seção 6 contém as conclusões do trabalho.

2. REPRESENTAÇÃO BÁSICA DO PROBLEMA

O problema de programação de veículos com uma única garagem (PPVUG) pode ser definido em termos de fluxo em redes como segue. Dado um conjunto de n viagens, $V = \{1,\ldots, n\}$, cada viagem $i \in V$ é representada por:

a) b_i o ponto inicial,
b) e_i o ponto final,
c) d_i o horário de partida de b_i
d) a_i o horário de chegada em e_i.

Para qualquer par de viagens (i, j), t_{ij} representa o *tempo de viagem em trânsito ou viagem morta* de e_i até b_j. A garagem é representada pelos nós r e s, onde r é o nó de partida da garagem e s é o nó de retorno à garagem, ambos referentes ao mesmo local. Assim, t_{ri} é o tempo de viagem morta da garagem até b_i e t_{is} é o tempo de viagem morta de e_i até a garagem. Um par de viagens (i, j) é dito *compatível* se $t_{ij} \leq d_j - a_i$ e o custo do arco (i, j) ligando as viagens i e j é dado por:

$$c_{ij} = \begin{cases} K_1 t_{ij} + Custo\ Capital/2 & \text{se } i = r \text{ ou } j = s, \\ K_1 t_{ij} + K_2\,(tempo\ de\ espera) & \text{se } (i, j) \text{ for um par de viagens compatíveis,} \\ \infty & \text{caso contrário,} \end{cases} \quad (1)$$

onde K_1 e K_2 são pesos definidos pelo usuário e o *tempo de espera* é dado por $d_j - a_i - t_{ij}$. Os valores destes pesos variam de acordo com a

situação operacional, porém K_1 é sempre maior do que K_2 em casos reais. O custo de capital é atribuído à primeira viagem de cada veículo que parte da garagem e à última viagem quando o veículo retorna à garagem, com a finalidade de minimizar o número de veículos na frota.

Seja $G = (N, A)$ o grafo direcionado com nós $N = V \cup \{r, s\}$ e arcos $A = \{(i, j) \mid (i, j)$ é um par de viagens compatíveis $\forall\, i, j \in V\} \cup \{(r, i), (i, s), \forall\, i \in V\}$. Para ilustrar a construção desta rede, considere o conjunto de quatro viagens com seus respectivos horários e locais de partida e chegada contidos na tabela abaixo. Para simplificar a explanação, considere os locais de partida e chegada no mesmo terminal A.

Tabela 1: Conjunto de 4 viagens com horários e locais de partida e chegada.

Viagem i	D_i	b_i	a_i	e_i
1	07:00	Term A	08:30	Term A
2	08:00	Term A	08:50	Term A
3	09:00	Term A	11:00	Term A
4	09:30	Term A	10:00	Term A

É fácil verificar que após executar a viagem 1, o mesmo veículo poderá executar as viagens 3 ou/e 4, visto que estas viagens têm seus horários de partida posteriores ao horário de chegada da viagem 1. Logo, são incluídos arcos ligando a viagem 1 às viagens 3 e 4, com tempos de espera de 30 e 60 minutos respectivamente. Os arcos que partem e chegam à garagem apresentam, além do custo de capital, o custo operacional para o veículo ir da garagem até o terminal e do terminal até a garagem. Estes custos podem diferir, dependendo do sentido da viagem. A figura 1 é a representação gráfica da rede G para as viagens da tabela 1.

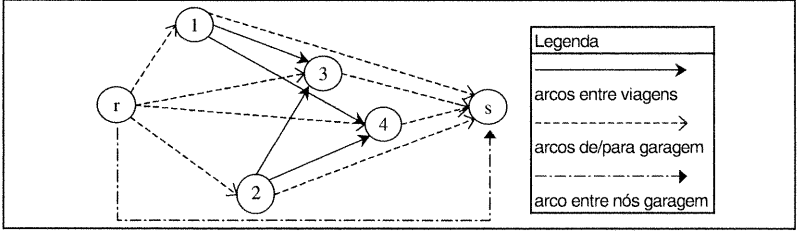

Figura 1: Rede genérica que representa o PPVUG na tabela 1.

2.1 Características da Representação Básica

Esta representação contém um grande número de arcos; portanto, existe a necessidade de descartar arcos que não tomarão parte na solução ótima. Neste sentido é feita a seguinte classificação dos arcos da rede:

a) *arcos longos* são aqueles que ligam duas viagens em que o tempo de espera entre elas é grande o suficiente para que o veículo retorne à garagem, permanecendo lá um intervalo mínimo de tempo antes de executar a próxima viagem. Este "retorno temporário" à garagem é importante do ponto de vista prático, pois permite realizar o abastecimento e serviços rápidos no veículo, além de criar a oportunidade para a troca da tripulação, pausa para descanso e/ou alimentação, execução de outras tarefas, ou mesmo a sua dispensa por um determinado período de tempo.

b) *arcos curtos* são aqueles que ligam viagens cujo tempo de espera entre elas não justifica o retorno temporário do veículo à garagem.

Tomando como referência a rede genérica apresentada na figura 1, define-se o conjunto de arcos longos $A^l \subset A$ como sendo as ligações entre as viagens que satisfazem a seguinte relação:

$$(i, j) \in A^l \Leftrightarrow d_j - a_i \geq t_{is} + \textit{tempo mínimo de garagem} + t_{rj} \quad (2)$$

onde o *tempo mínimo de garagem* é um parâmetro definido pelo usuário. O conjunto dos arcos curtos A^c é dado por $A^c = A - A^l$. Embora a maioria dos arcos na rede seja constituída de arcos longos, apenas uma ínfima parcela destes arcos toma parte da solução ótima. Portanto, o conjunto dos arcos longos é o mais propício a ser reduzido.

A rede apresentada na figura 1 é apenas uma primeira representação para o problema. Na verdade os algoritmos de resolução são aplicados a outras redes que derivam desta.

3. O MODELO DE PSEUDO-DESIGNAÇÃO PARA O PPVUG

O modelo de pseudo-designação atua sobre dois conjuntos disjuntos ligando cada nó de um conjunto a pelo menos um nó de outro conjunto, tal que o custo total das ligações seja mínimo. Portanto, para cada viagem $i \in V$ são definidos os nós i' de início e i'' de final de viagem, com respectivas demanda e oferta de um veículo. A garagem é representada pelo nó r de partida e pelo nó s de retorno, com ofertas e demandas de n veículos, respectivamente.

A divisão natural dos nós da rede em dois conjuntos disjuntos é dada pela definição dos seguintes conjuntos: $N_1 = \{r\} \cup \{i'' \mid i = 1, ..., n\}$ dos nós com oferta de fluxo, e $N_2 = \{s\} \cup \{i' \mid i = 1, ..., n\}$ dos nós com demanda de fluxo. O conjunto de arcos ligando os nós de N_1 a N_2 é definido por A = $\{(r, i') \mid i = 1, ..., n\} \cup \{(i'', j') \mid (i, j)$ é um par de viagens compatíveis$\} \cup \{(i'', s) \mid i = 1, ..., n\}$. O primeiro subconjunto de A liga a partida da garagem até o início de cada viagem; o segundo subconjunto liga os pares de viagens compatíveis; e no terceiro subconjunto, o final de cada viagem é ligado ao retorno à garagem. Assim, tem-se a rede bipartida $G^{pd} = (N_1, N_2, A)$. Um exemplo deste tipo de rede é mostrado na figura 2, na qual os valores entre chaves representam as respectivas ofertas e demandas de veículos, de acordo com o tipo de nó da rede, e os custos nos arcos (i'', j') seguem a expressão dada em (1). O arco que liga as garagens tem custo nulo e capacidade igual ao total de viagens n.

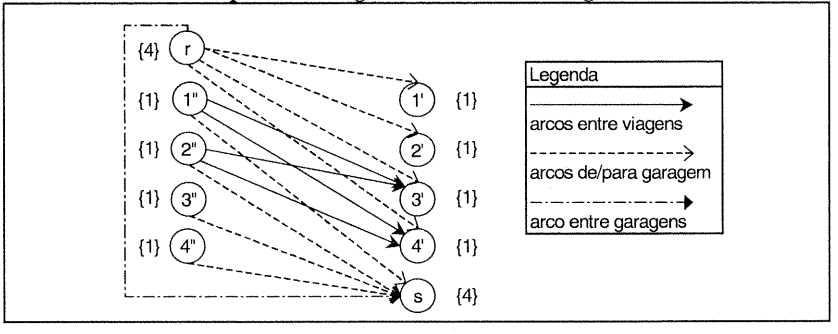

Figura 2: Rede G^{pd} para o problema descrito na tabela de viagens.

A formulação matemática para este modelo de pseudo-designação é dada então por:

$$\text{Min} \sum_{(i,j)\in A} c_{ij} x_{ij} \qquad \text{sujeito a} \qquad (3)$$

$$\sum_{j\in N_2} x_{ij} = 1 \qquad \forall\, i \in N_1 - \{ r \} \qquad (4)$$

$$\sum_{i\in N_1} x_{ij} = 1 \qquad \forall\, j \in N_2 - \{ s \} \qquad (5)$$

$$\sum_{j\in N_2} x_{rj} = n \qquad (6)$$

$$\sum_{i\in N_1} x_{is} = n \qquad (7)$$

$$x_{ij} \in \{0, 1\} \qquad \forall\, (i, j) \in A \qquad (8)$$

onde a variável de decisão $x_{ij} = 1$ se a viagem j for executada pelo mesmo veículo imediatamente após a execução da viagem i, e $x_{ij} = 0$ caso contrário. A expressão (3) minimiza o custo total, as restrições (4) e (5) asseguram que cada viagem será executada uma única vez, enquanto as restrições (6) e (7) estão relacionadas com as partidas e chegadas dos veículos à garagem. A restrição (8) garante a integralidade da solução. A partir da solução deste modelo é possível montar os blocos de viagens para os veículos. Nesta rede, cada caminho saindo de r e chegando em s corresponde a um bloco de viagens a ser executado por um veículo. O conjunto de todos os caminhos disjuntos de r para s, contidos na solução, fornece a programação completa, com o menor número de veículos e custo operacional.

3.1 A Técnica de Geração de Colunas em Problemas de Programação Linear

Para reduzir o tamanho do problema, foi implementada uma versão para modelos de fluxo em redes da conhecida técnica de geração de colunas para a programação linear. A geração de colunas permite solucionar o problema de otimização sem levar em conta todas as colunas da matriz. A idéia consiste em resolver o problema com um subconjunto mínimo da região de soluções viáveis. A partir deste

problema inicial e sua solução, o método tenta gerar novas colunas que permitam melhorar o valor da função objetivo. Cada uma destas novas colunas é obtida através da resolução de um problema de caminho mínimo associado ao problema dual. Quando nenhuma coluna com a capacidade de melhorar o valor da função objetivo puder ser gerada, o processo é finalizado e a solução corrente é ótima. Com isso torna-se possível resolver problemas de grande porte sem comprometer o tempo de processamento. Vários problemas de programação de veículos e de suas tripulações foram resolvidos através desta técnica, incluindo a programação de ônibus, trens, e aeronaves (Crainic e Rousseau (1987), Desrochers e Soumis (1989) e Fores et al. (1999)). Trabalhos mais recentes propõem uma metodologia de *branch-and-price,* que é um processo de *branch-and-bound* combinado com geração de colunas, para resolver problemas de programação inteira mista de grande porte (Löbel (1999) e Barnhart et al. (1998)).

3.2 Técnica de Geração de Arcos em Problemas de Fluxo em Redes

O problema de fluxo em redes é um caso particular de um problema de programação linear, onde a matriz associada à rede, i. e., a *matriz de incidência* da rede, apresenta em suas linhas as informações dos nós e em suas colunas as informações dos arcos. Assim, ao gerar uma nova coluna desta matriz, um novo arco está sendo gerado na rede. As colunas da matriz apresentam as informações dos arcos, por isso elas são compostas por apenas dois elementos diferentes de zero, os quais se referem aos nós nas extremidades do arco. Esta característica da matriz torna a técnica de Geração de Arcos um processo mais simples do que na geração de colunas, visto que o cálculo do custo relativo dos arcos externos não envolve a matriz inversa da base, mas simplesmente os valores duais da solução corrente. Portanto, nenhum esforço computacional extra é necessário para se conhecer os arcos com capacidade de melhorar a solução corrente. Nesta adaptação, os arcos gerados não são obtidos através de sub-problemas de otimização. O conjunto dos arcos que não fazem parte do processo de otimização é armazenado e pesquisado com a finalidade de gerar um novo domínio para o problema.

A estratégia de Geração de Arcos é resolver uma seqüência de problemas, iniciando com um domínio composto de um pequeno conjunto de arcos com grande possibilidade de serem utilizados na solução. A partir dos valores duais fornecidos pela solução do problema corrente é feita uma busca na lista de arcos fora do domínio, identificando aqueles que têm capacidade de melhorar a solução. Se for encontrado algum arco fora do domínio que possa melhorar a solução corrente, então este arco será incorporado à rede e o problema será resolvido novamente. Este processo é repetido até que nenhum arco externo possa melhorar a solução. Quando isto ocorrer, o problema estará resolvido e a solução corrente será ótima. Abaixo é apresentado o esquema que sintetiza a técnica de Geração de Arcos.

Inicialização: Seja S e L conjuntos tais que $S \cup L = A$, onde A é o conjunto de todos os arcos da rede a ser trabalhada.

P1. Resolver o problema de pseudo-designação definido em (3) – (8) tendo como domínio o subconjunto S.

P2. Para cada arco $(i, j) \in L$ faça:

Calcule $\bar{c}_{ij} = c_{ij} - \pi_i - \pi_j$

Se $\bar{c}_{ij} < 0$

Então incluir o arco (i, j) a S e retirá-lo de L.

P3. **Se** algum arco foi incluído em S durante o passo P2
Então retornar ao passo P1,
Senão finalizar o processo. A solução corrente é ótima.

3.3. O Algoritmo de Geração de Arcos para o Problema de Programação de Veículos

A partir dos dados de entrada, que são: 1) o conjunto de todas as viagens a serem executadas e 2) a matriz de viagens mortas entre os p terminais $T = (t_{ij})_{p \times p}$, o PPVUG é formulado como um problema de pseudo-designação e sua rede correspondente àquela encontrada na figura 2 é criada. O conjunto S é inicializado com o conjunto A^c dos arcos curtos e o conjunto L é inicializado com o conjunto A^l dos arcos longos.

Para resolver a seqüência de problemas de pseudo-designação gerada no passo P1 do processo descrito na seção anterior, foi utilizado o algoritmo de fluxo em redes Out-of-Kilter (Fulkerson 1961) e cada problema de re-otimização é inicializado com a solução do problema anterior. Para aplicar o algoritmo Out-of-Kilter, a rede dever ser *fortemente conectada*, isto é, ela deve conter pelo menos um caminho direcionado partindo de qualquer nó para qualquer outro nó da rede. Logo, foram incluídos arcos auxiliares à rede G^{pd} ligando cada nó início de viagens i' ao seu nó de término i''.

3.4. Análise da Técnica de Geração de Arcos

Com a técnica de Geração de Arcos foi possível resolver de forma eficiente problemas de grande porte, assim como tornar o modelo mais adequado à prática operacional no transporte público. Foram implementadas duas restrições adicionais muito encontradas na prática, que serão descritas em maiores detalhes abaixo.

3.4.1. Problema do Retorno à Garagem

Para justificar o retorno de um veículo à garagem no intervalo entre duas viagens, é necessário que o mesmo permaneça no local por um determinado *tempo mínimo de garagem* definido previamente. Durante o tempo em que o veículo permanece na garagem, o custo operacional é tomado como constante, independente do tempo total de estacionamento. Esta consideração implica uma adaptação da função de custo nos arcos longos, que é dada por:

$$(i, j) \in A^l \Rightarrow c_{ij} = K_1(t_{is} + t_{rj}) + K_2(\textit{tempo mínimo de garagem}) \quad (9)$$

Logo, o custo atribuído aos arcos longos depende apenas dos tempos de viagem morta à garagem, sendo constante em relação ao tempo de permanência na garagem. Esta estrutura de custo é adotada na grande maioria das situações práticas e foi incorporada ao modelo.

3.4.2. *Problema de Preferência de Ligações*

Em alguns casos o operador do sistema sabe de antemão que certas rotas devem ter preferência para que suas viagens sejam ligadas às viagens de outro grupo específico de rotas. Isto se deve por razões operacionais, tais como a previsão de atrasos em rotas sobrecarregadas, fluxo de passageiros em um determinado sentido, condições para a programação da tripulação, etc. Nestes casos o operador pode admitir um certo custo extra a fim de priorizar a ligação entre determinados grupos de rotas.

Esta situação de *preferência de ligação* é definida por dois conjuntos de rotas e um *fator de redução* no custo da ligação. As viagens do conjunto das *rotas de chegada* têm preferência para serem ligadas às viagens do conjunto das *rotas de partida*, tendo seu custo reduzido pelo fator de redução no custo da ligação. Uma rota pode ter prioridade de ligação com mais do que uma outra rota, o que é denominado múltipla preferência de ligações. Este tipo de restrição também altera a estrutura da função de custo dos arcos e foi implementada com a ajuda da técnica de Geração de Arcos.

4. O MÉTODO HEURÍSTICO BOOST

O algoritmo de programação de ônibus BOOST (Basis for Object Oriented Scheduling of Transport) pertence à classe das heurísticas conhecidas como "2-opt". O BOOST (Kwan e Rahin 1999) foi desenvolvido tendo como base o sistema VAMPIRES (Wren 1972 e Wren e Smith 1981), o qual usa a mesma heurística 2-opt. Porém, fazendo uso da filosofia de programação orientada a objetos, o BOOST tem alcançado uma melhoria significativa em termos de qualidade de programação e tempo de processamento em relação à última versão do VAMPIRES, provando ser um sistema robusto o suficiente a ponto de suportar extensões necessárias à resolução de programações mais complexas. Este sistema tem sido usando em várias companhias de ônibus no Reino Unido e no Brasil (Wren e Kwan 1999, Wren e Gualda 1999).

O sistema heurístico BOOST forma inicialmente uma programação simplificada, usando um dado número de veículos que pode ser

estimado automaticamente ou fornecido pelo operador do sistema. É possível que a programação inicial tenha algumas ligações infactíveis, isto é, o veículo chegue num determinado terminal depois do seu suposto horário de partida. Neste caso, a programação é então refinada e a cada iteração um determinado par de ligações é quebrado e religado no sentido oposto, caso haja uma redução na infactibilidade total e/ou no custo da programação. O sistema retorna à melhor solução factível encontrada ou, caso o número de veículos especificado no início seja muito baixo, aponta as viagens críticas cujos horários devam ser alterados tal que a programação seja factível. Quando o BOOST é forçado a usar um número de veículos menor do que o mínimo possível, o sistema retorna à melhor solução infactível encontrada.

Nos testes realizados neste trabalho as soluções encontradas pelo sistema heurístico foram comparadas com aquelas fornecidas pelo método exato descrito acima, por isso o BOOST foi forçado a gerar somente soluções factíveis, uma vez que os métodos exatos descartam soluções infactíveis.

5. RESULTADOS OBTIDOS

Para implementar o algoritmo de Geração de Arcos para o PPVUG, descrito nas seções 3.2. e 3.3., foi utilizada a versão do algoritmo Out-of-Kilter em DELPHI, desenvolvida no LPT/EPUSP – Laboratório de Planejamento e Operação de Transportes do Departamento de Engenharia de Transportes da Escola Politécnica da Universidade de São Paulo. Nas tabelas abaixo foram utilizadas as iniciais G. A. como referência ao algoritmo de Geração de Arcos.

Os resultados obtidos com o algoritmo de Geração de Arcos foram comparados com aqueles fornecidos pelo sistema heurístico BOOST e estão resumidos nas tabelas 2 a 4, nas quais os tempos são apresentados no formato horas:minutos. A coluna "Núm. de Retornos à Garagem (duração)" mostra o total de retornos temporários à garagem, com o tempo total de estacionamento temporário mostrado entre parênteses. O Custo Total, com os tempos em minutos, é computado pela expressão:

$$Custo\ Total = 2 \times Viagens\ Mortas + Tempo\ de\ Espera + tempo\ mínimo\ de\ garagem \times número\ de\ returnos \quad (10)$$

Esta expressão é utilizada no sistema BOOST e noutros sistemas de programação de veículos, produzindo resultados de boa qualidade em vários casos reais. Entretanto, no sistema BOOST, esta expressão é complementada com outras regras para tratar de situações nas quais uma função objetivo rígida não consegue avaliar os vários aspectos reais. Em alguns casos a solução ótima não tem qualidades tão apreciáveis quanto soluções quase ótimas.

Uma dificuldade em obter uma função objetivo apropriada para o problema está em encontrar o equilíbrio para o número de retornos à garagem. Os retornos à garagem são importantes na solução, mas não podem ser muito freqüentes, caso contrário a programação terá muitas viagens mortas durante o período e a duração de cada parada temporária será curta. Por esta razão foi adotado o período de 30 minutos para o tempo mínimo de garagem, que é um valor largamente utilizado no Reino Unido (comunicação pessoal, Wren e Kwan, 1999).

Outra dificuldade na definição de uma função objetivo compatível com a realidade é o impacto da programação dos veículos sobre a programação da tripulação. Na programação dos veículos, o tempo de espera nos terminais é mais barato do que o tempo de viagem morta. No entanto, para a tripulação ambos têm o mesmo custo, a menos que o veículo possa ser abandonado pela tripulação quando este estiver estacionado.

Tais dificuldades podem ser melhor trabalhadas nos métodos heurísticos, devido às suas flexibilidades. O sistema BOOST, por exemplo, não considera apenas o custo total na avaliação de um resultado, mas também leva em conta a distribuição do tempo de trabalho entre viagens mortas, tempo de espera nos terminais e número de retornos temporários à garagem. O mesmo não acontece com os métodos exatos, os quais têm como objetivo diminuir o custo total, sem levar em conta as outras características da solução.

Na prática, a qualidade de uma programação depende dos seguintes atributos: a) tempo total de viagens mortas, b) tempo total de espera nos terminais, c) número de retornos à garagem e d) tempo total de estacionamento temporário na garagem. O custo total é, portanto, mais uma referência para verificar a qualidade da solução obtida.

5.1 Testes com dados de Reading – Reino Unido

Este testes foram realizados utilizando dados reais da cidade de Reading no Reino Unido. A seqüência de problema foi criada aumentando-se o número de rotas incluídas na programação, de tal maneira que o número de veículos aumentasse progressivamente.

Ambos os métodos apresentam os mesmos resultados para os três primeiros testes, mostrando que nestes casos é irrelevante qual o método utilizado. Porém, no teste número quatro o algoritmo de Geração de Arcos fornece uma solução com custo total ligeiramente melhor do que aquela fornecida pelo BOOST. Analisando a qualidade desta solução, verifica-se que o algoritmo de Geração de Arcos tem 57 minutos a menos de viagem morta e 5 horas e 38 minutos a mais de tempo de espera nos terminais do que o sistema BOOST. Por outro lado, a solução do BOOST tem 8 retornos à garagem a mais do que o algoritmo de Geração de Arcos, com um total de 4 horas e 40 minutos a mais de tempo de estacionamento temporário.

Tabela 2: Testes com dados da cidade de Reading – Reino Unido.

Teste	Núm. de Veículos	Tempo de Viagens Mortas		Tempo de Espera nos Terminais		Núm. de Retornos à Garagem (duração)		Custo Total	
		G. A.	Boost	G. A.	Boost	G. A.	Boost	G. A.	Boost
1	47	17:53	17:53	57:16	57:16	---	---	5.582	5.582
2	59	21:07	21:07	65:32	65:32	3 (03:45)	3 (03:45)	6.556	6.556
3	68	23:49	23:49	78:20	78:20	3 (03:45)	3 (03:45)	7.648	7.648
4	76	24:28	25:25	71:26	65:48	3 (03:45)	11 (08:25)	7.312	7.328
5	84	26:33	26:25	69:29	69:45	3 (03:45)	3 (03:45)	7.445	7.445

Tendo em vista a realidade operacional Britânica, a solução do sistema BOOST é considerada melhor do que a solução do método de Geração de Arcos, pois a tripulação recebe o mesmo valor/hora de trabalho independente de se tratar de hora de viagem morta ou de espera nos terminais. Sendo assim, a solução heurística tem 4 horas e 51 minutos a menos de pagamento de tripulação por dia do que a solução exata fornecida pelo algoritmo de Geração de Arcos. Esta é uma situação em que a solução heurística quase ótima apresenta uma qualidade melhor do que a solução ótima, por levar em conta a programação da tripulação ao programar os veículos.

5.1.1. *Reading com Preferência de Ligações*

O caso estudado tem um total de 309 viagens distribuídas em 36 rotas, com preferência de ligação entre certos conjuntos de rotas. Inicialmente o problema foi resolvido sem levar em conta as preferências de ligação entre as rotas. Posteriormente as preferências foram ativadas e o problema foi resolvido novamente. Foram considerados onze grupos diferentes de preferências, cada um deles definido através dos conjuntos de chegadas e de partidas que variam de uma a dez rotas. O valor adotado para o redutor do custo de ligação foi de quinze minutos e a tabela abaixo sintetiza os resultados obtidos.

Tabela 3: Soluções sem e com as preferências de ligação (testes 1 e 2 respectivamente).

Teste	Num. de Veículos	Tempo de Viagens Morta		Tempo de Espera nos Terminais		Ligações Preferencias Satisfeitas		Custo Total	
		G. A.	Boost	G. A.	Boost	G. A.	Boost	G. A.	Boost
1	15	05:09	05:11	17:01	16:59	96	97	1.639	1.641
2	15	05:09	05:09	17:08	17:01	105	104	1.646	1.639

No primeiro caso, sem as preferências de ligação, o método de Geração de Arcos produziu uma programação mais barata, embora tenha satisfeito espontaneamente um número menor de ligações preferenciais. Uma vez ativadas as preferências, o algoritmo de

Geração de Arcos efetuou um número maior de ligações preferenciais, embora o custo total tenha sido maior do que no caso do sistema BOOST.

Foi verificado um melhor desempenho qualitativo do método exato que produziu um resultado mais barato quando não existem preferências de ligação, e um resultado com maior número de ligações preferenciais satisfeitas quando estas são requeridas. O sistema BOOST produziu resultados teoricamente incoerentes, pois o problema mais restrito tem custo total menor do que o problema mais relaxado.

5.2 Testes com dados de Sorocaba – Brasil

O conjunto de dados da cidade de Sorocaba é consideravelmente maior do que os da cidade de Reading, tendo sido estudado previamente por Wren e Gualda (1999). O maior teste realizado com os dados de Reading (teste 5 da tabela 7) tem 1.518 viagens com um mínimo de 84 veículos, enquanto o maior teste da cidade de Sorocaba trata de 2.732 viagens que necessitam de no mínimo 111 veículos. A rede montada para representar o maior problema da cidade de Sorocaba é composta por 5.466 nós e 3.379.000 arcos. Para resolver problemas desta magnitude faz-se necessária a aplicação de técnicas de Geração de Arcos, que permitem encontrar a solução sem considerar todos os arcos da rede.

Tabela 4: Testes com dados da cidade de Sorocaba – Brasil.

Teste	Núm. de Veículos	Tempo de Viagens Mortas		Tempo de Espera nos Terminais		Núm. de Retornos à Garagem (duração)		Custo Total	
		G. A.	Boost	G. A.	Boost	G. A.	Boost	G. A.	Boost
1	107	122:08	122:08	124:48	124:48	23 (165:13)	23 (165:09)	22.834	22.834
2	111	94:42	95:33	101:19	100:27	28 (191:52)	28 (192:18)	18.283	18.333

No primeiro problema, as soluções produzidas pelos dois métodos coincidem em custo total e qualidade. No entanto, para o segundo problema, o algoritmo de Geração de Arcos encontra uma solução com menor custo total, com 51 minutos a menos de viagem morta e

52 minutos a mais de tempo de espera nos terminais. As soluções apresentam praticamente mesmo custo com tripulação (viagem morta + espera nos terminais) e igual número de retornos à garagem, porém a solução do sistema BOOST conta com 36 minutos a mais de estacionamento temporário.

5.3 Tempo de Processamento: Algoritmo de Geração de Arcos x Sistemas BOOST

O sistema BOOST é um modulo de um pacote para a programação de veículos e tripulação denominado OpenBus, logo o tempo de processamento para encontrar uma solução para o PPVUG envolve também o tempo de processamento de interface. Entretanto, para se ter uma idéia, o algoritmo de Geração de Arcos exigiu de 1 a 1,5 vezes o tempo requerido pelo BOOST nos testes com os dados de Reading e até 6 vezes para os testes com a cidade de Sorocaba.

6. CONCLUSÕES

A metodologia explorada neste trabalho alcança bons resultados tanto na redução da rede quanto na inclusão de restrições adicionais. Assim, foi possível formular e resolver problemas reais de grande porte sem que houvesse uma degeneração no tempo de processamento.

Na maioria dos casos o método heurístico e o algoritmo de fluxo em redes encontraram a mesma solução, sendo que para alguns problemas de maior porte, o algoritmo de fluxo em redes encontra o ótimo global que o método heurístico não atinge. Considerando somente o custo total definido pela expressão (13), o método heurístico apresenta soluções que diferem do ótimo em 0,22% no caso da cidade de Reading e em 0,27% no caso da cidade de Sorocaba. Embora as diferenças não sejam grandes, deve ser lembrado que a programação é executada diariamente. Logo uma pequena diferença diária pode acarretar em um economia significativa no período de um ano.

Foi mostrado que problemas de diferentes tamanhos e características podem ser resolvidos com o algoritmo de Geração de Arcos em um

razoável tempo de processamento e produzindo soluções de alta qualidade operacional. O bom desempenho do algoritmo de Geração de Arcos possibilita a utilização da metodologia desenvolvida neste trabalho para resolver problemas reais de empresas de transporte público, levando a uma possível economia nos custos de capital e operacional das empresas do setor.

AGRADECIMENTOS

Os autores agradecem o Conselho Britânico e a CAPES por financiarem o intercâmbio acadêmico, que tornou este trabalho possível. Os autores agradecem também o Prof. Anthony Wren e o Dr. Raymond S. K. Kwan da School of Computer Studies, Universidade de Leeds, Reino Unido, e o Dr. Cláudio Barbieri da Cunha do Departamento de Engenharia de Transportes-EPUSP, pelas sugestões, comentários e assistência prestada.

REFERÊNCIAS BIBLIOGRÁFICAS

Barnhart, C.; E.L. Johnson; G.L. Nemhauser; M.W.P. Savelsbergh e P.H. Vance (1998). Branch-and-price: column generation for solving huge integer programs. *Operations Research*, v 46, p. 316-329.

Bodin, L.; B. Golden; A. Assad e M. Ball. (1983) Routing and scheduling of vehicles and crews: The state of the art. *Computers and Operations Research*, v 10, p. 63-211.

Carraresi, P. e G. Gallo (1984) Network models for vehicle and crew scheduling. *European Journal of Operational Research*, v 16, p. 139-151.

Crainic, T e J.M. Rousseau (1987) The column generation principle and the airline crew scheduling problem. *INFOR*, v 25, p. 136-151.

Daduna, J.R. e J.M.P. Paixão (1995) Vehicle scheduling for public mass transport- an overview. *In*: Daduna, J.R.; I. Branco e J.M.P. Paixão (*eds.*) *Computer-Aided Transit Scheduling*, Lectures Notes in Economics and Mathematical Systems. Springer Verlag, Berlin, Alemanha.

Desrochers, M e F. Soumis (1989) A column generation approach to the urban transit crew scheduling problem. *Transportation Science*, v 23, p. 1-13

Fores, S.; L. Proll e A. Wren (1999) An improved ILP system for driver scheduling. In N. Wilson (Ed.), *Computer-Aided Transit Scheduling, Lecture Notes in Economics and Mathematical Systems*, Vol 471, Springer, Berlin, 43-61.

Freling, R.; J.M.P. Paixão e A.P.M. Wagelmans (1995) Models and Algorithms for Vehicle Scheduling. Aviable via WWW at http://kaa.cs.few.eur.nl/few/people/freling/.

Fulkerson, D. (1961) An out-of-kilter method for minimal cost flow problems. *SIAM Journal on Applied Mathematics*, v 9, p. 18-27.

Gavish, B.; P. Schweitzer e E. Shlifer (1978) Assigning buses to schedules in a metropolitan area. *Computers and Operations Research*, v 5, p. 129-134.

Kwan, R.K. e M.A. Rahin (1999) Object orientede bus scheduling - the BOOST system. . In N. Wilson (Ed.), *Computer-Aided Transit Scheduling, Lecture Notes in Economics and Mathematical Systems*, Vol 471, Springer, Berlin, 177-191.

Löbel, A. (1999) Solving large-scale multiple-depot vehicle scheduling problems. . In N. Wilson (Ed.), *Computer-Aided Transit Scheduling, Lecture Notes in Economics and Mathematical Systems*, Vol 471, Springer, Berlin, 193-220.

Paixão, J. e L. Branco (1987) A quasi-assignment algorithm for bus-scheduling, *Networks*, v 17, p. 249-269.

Silva, G.P.; R.S.K. Kwan e N.D.F. Gualda (1998) Vehicle scheduling with network flow models. *Transportes*, v 2, p. 9-27.

Smith, B.M. e A. Wren (1981) VAMPIRES and TASC: two successfully applied bus scheduling programs. *In:* Wren A. *(ed.) Computer Scheduling of Public Transport*. North-Holland, Amsterdam, Netherlands.

Wren, A. (1972) Bus scheduling: and interactive computer method. *Transportation Planning and Technology*, v 1, p. 115-122.

Wren, A. e N.D.F. Gualda (1999) Integrated scheduling of buses and drivers. . In N. Wilson (Ed.), *Computer-Aided Transit Scheduling, Lecture Notes in Economics and Mathematical Systems*, Vol 471, Springer, Berlin, 155-176.

Wren, A. e R.S.K. Kwan (1999) Installing an urban transport scheduling system. A aparecer em: *Journal of Scheduling*.

Endereço dos autores:
Gustavo Peixoto Silva
Nicolau D. Fares Gualda
LPT / EPUSP - Laboratório de Planejamento e Operação de Transportes
Departamento de Engenharia de Transportes
Escola Politécnica da Universidade de São Paulo
Caixa Postal: 61548 CEP: 05424-970 São Paulo SP
E-mail : gpsilva@usp.br
ngualda@usp.br