TEMA Tend. Mat. Apl. Comput., **5**, No. 2 (2004), 357-368.

Metaheurísticas Aplicadas ao Problema de Programação de Tripulações no Sistema de Transporte Público

M.J.F. SOUZA[1], L.X.T. CARDOSO[2], Departamento de Ciência da Computação, Instituto de Ciências Exatas e Biológicas, Universidade Federal de Ouro Preto, 35400-000 Ouro Preto, MG, Brasil.

G.P. SILVA[3], M.M.S. RODRIGUES[4], S.M.S. MAPA[5], Departamento de Engenharia de Produção, Escola de Minas, Universidade Federal de Ouro Preto, 35400-000 Ouro Preto, MG, Brasil.

Resumo. Este trabalho aborda o Problema de Programação de Tripulações (PPT) no Sistema de Transporte Público. Tal problema consiste em atribuir um conjunto de tarefas aos tripulantes de uma dada empresa participante do sistema de forma que todas as viagens das linhas sob responsabilidade desta sejam executadas com o menor custo possível. A solução do PPT é um conjunto de jornadas diárias de trabalho de tripulantes. Neste trabalho, o PPT foi abordado utilizando as metaheurísticas *Simulated Annealing* (SA), Método de Pesquisa em Vizinhança Variável e Busca Tabu (BT). Esses métodos exploram o espaço de soluções utilizando diferentes estruturas de vizinhança, as quais modificam as jornadas de trabalho através de operações realizadas com suas tarefas. Cada solução gerada pelos métodos é avaliada por uma função baseada em penalidades que visa atender a legislação trabalhista, as regras operacionais da empresa, assim como melhorar o aproveitamento da mão-de-obra operacional. Os algoritmos foram testados com dados reais de uma empresa que opera na cidade de Belo Horizonte.

1. Introdução

O Problema de Programação de Tripulações de uma empresa do Sistema de Transporte Público (PPT) consiste em gerar uma escala de trabalho, isto é, um conjunto de jornadas de trabalho para as tripulações que conduzirão a frota em operação. Tais jornadas devem contemplar todas as viagens sob responsabilidade da empresa e satisfazer a um conjunto de leis trabalhistas e regras operacionais da empresa, minimizando o número de tripulações e horas-extras necessárias.

[1]marcone@iceb.ufop.br
[2]leoxt@iceb.ufop.br
[3]gustavo@depro.em.ufop.br
[4]margaridaop@yahoo.com.br
[5]silvinha@engineer.com

A resolução do PPT é de grande importância, uma vez que nos gastos totais de uma empresa, a mão-de-obra operacional representa um dos maiores custos [2], motivo pelo qual esse tema tem sido largamente estudado. A abordagem mais explorada é aquela que formula o PPT como um problema de recobrimento ou de particionamento e utiliza a técnica de geração de colunas para resolvê-lo [20, 5, 8, 1, 9]. A variedade de trabalhos deriva das diferentes maneiras de gerar as colunas e diferentes metodologias para resolver o problema, tais como: *branch-and-bound*, *branch-and-price* e a relaxação lagrangeana.

Dada a NP-completude do problema, os sistemas heurísticos foram os primeiros a serem utilizados na resolução do PPT, os quais consistiam apenas na automação do trabalho antes realizado manualmente [6]. A principal desvantagem de tais sistemas é a sua incapacidade de detectar possibilidades de otimização. Manington e Wren [15] iniciaram a inclusão de procedimentos de otimização neste tipo de sistema, desenvolvendo heurísticas de otimização. Embora tais heurísticas não garantissem a obtenção do ótimo global, elas eram capazes de produzir soluções em baixos tempos computacionais, além de permitirem incluir com facilidade qualquer tipo de restrição. Entretanto, tais heurísticas geravam soluções de baixa qualidade e muitas das vezes infactíveis. Isto acontecia porque elas ficavam presas nos primeiros ótimos locais encontrados. Com o surgimento das metaheurísticas, tais como Algoritmos Genéticos [11], Busca Tabu [10] e *Simulated Annealing* [12], abriu-se um novo horizonte na resolução do PPT. Embora tais métodos também não garantam a obtenção do ótimo global, eles são providos de mecanismos para escapar de ótimos locais. Na literatura podemos destacar nessa linha os trabalhos de Clement e Wren [3], Wren e Wren [24] e Kwan et al. [13] que utilizam Algoritmos Genéticos, enquanto Shen e Kwan [17] utilizam Busca Tabu.

Atualmente, as empresas devem atender outros objetivos conflitantes com a redução dos custos, tais como manutenção da qualidade do serviço e manutenção de determinadas características das jornadas de trabalho. Para abordar tal problema trabalhos recentes fazem uso de otimização multiobjetivo. Lourenço et al. [14] utilizam as metaheurísticas Busca Tabu e Algoritmo Genético para selecionar soluções não dominadas, segundo o conceito multiobjetivo. As colunas de tais soluções, geradas ao longo do processo, são armazenadas para constituir o domínio sobre o qual é aplicado um método exato para o problema de particionamento, quando se tratar de instâncias de pequeno porte. Para instâncias mais elevadas, é aplicado um método GRASP [7] desenvolvido pelos autores para esse fim.

Uma revisão dos principais métodos de solução desenvolvidos para o PPT pode ser encontrada em [23], muitos dos quais publicados em uma série de conferências internacionais sobre o tema (vide, por exemplo, [4, 22, 21]). Trabalhos que abordam casos brasileiros podem ser vistos em [19] e [18], por exemplo.

Neste trabalho, aborda-se o PPT através de modelos heurísticos baseados nas técnicas *Simulated Annealing* (SA), Método de Pesquisa em Vizinhança Variável (VNS) e Busca Tabu (BT). Esses métodos exploram o espaço de soluções utilizando diferentes estruturas de vizinhança, as quais modificam as jornadas de trabalho através de operações realizadas com suas tarefas.

Este trabalho está organizado como segue. A Seção 2. descreve o problema abordado. Na Seção 3. mostra-se a forma de representação de uma solução do pro-

blema, as estruturas de vizinhança desenvolvidas, como uma solução é avaliada e como são geradas as soluções iniciais dos métodos. As Subseções 3.5., 3.6. e 3.7. apresentam os métodos heurísticos desenvolvidos para a solução do PPT. Os resultados computacionais obtidos são apresentados e analisados na Seção 4., enquanto as conclusões se encontram na última seção.

2. Descrição do Problema

No transporte público, usualmente a programação da tripulação é feita após a programação dos veículos. Nesta, as viagens são reunidas em *blocos*. Um bloco representa a seqüência de viagens que um determinado veículo tem que realizar em um dia, começando e terminando na garagem. Cada bloco mostra também as *Oportunidades de Troca* (OT). Uma OT é um intervalo de tempo suficiente, em um ponto apropriado, para haver a troca das tripulações. A partir do bloco de um veículo são criadas as tarefas. Cada tarefa é um conjunto de viagens compreendidas entre duas OT*'s*. Assim, durante sua realização não é possível que haja troca de tripulação.

A programação de uma tripulação é formada por um conjunto de tarefas, chamado de *jornada*. As jornadas são divididas em dois tipos: Pegada Simples ou Dupla Pegada. No primeiro tipo as tarefas são realizadas de uma única vez e os intervalos de tempo entre as tarefas são inferiores a duas horas. Caso ocorra um intervalo maior que duas horas, a jornada é classificada como de dupla pegada. Este intervalo não é contabilizado na remuneração da tripulação.

Ao se reunir as tarefas formando as jornadas, deve-se levar em conta inúmeras restrições operacionais e trabalhistas. As restrições consideradas neste trabalho referem-se àquelas praticadas por uma empresa do Sistema de Transporte Público do município de Belo Horizonte no ano 2002. Essas restrições podem ser classificadas em dois tipos: restrições essenciais (aquelas que obrigatoriamente têm que ser satisfeitas) e restrições não essenciais (aquelas cujo atendimento melhoram a qualidade das jornadas de trabalho mas que, se não satisfeitas, não geram jornadas inviáveis). As seguintes restrições são consideradas essenciais: (a) Uma tripulação não pode realizar mais de uma tarefa ao mesmo tempo; (b) As trocas das tripulações só podem ocorrer nas $OT\acute{}s$; (c) As trocas das tripulações só podem ocorrer entre grupos de linhas predeterminadas, ou seja, grupos de linhas com as mesmas características; (d) Uma tripulação que faz pegada simples tem direito a 30 minutos de descanso/alimentação durante sua jornada diária de trabalho, podendo este período ser fracionado em intervalos menores, desde que um deles seja maior ou igual a 15 minutos; (e) A jornada normal de trabalho diário é de 7:10 horas para as tripulações com pegada simples e 6:40 horas para aquelas com dupla pegada, acrescidas de até duas horas extras; (f) O tempo entre o final de uma jornada diária de trabalho e o seu início no dia seguinte deve ser de, no mínimo, 11 horas; (g) O número de jornadas do tipo dupla pegada deve estar limitado a um certo valor.

As restrições não essenciais consideradas foram as seguintes: (h) O tempo ocioso de uma tripulação deve ser o menor possível; (i) O número de horas extras deve ser minimizado; (j) O número de tripulações deve ser mínimo; (k) O número de vezes que uma tripulação troca de veículo deve ser reduzido; (l) O número de vezes

que uma tripulação com dupla pegada finaliza a primeira parte da jornada em um ponto e inicia a segunda parte em um outro ponto deve ser reduzido.

3. Modelagem do Problema

3.1. Representação

Uma solução do problema é representada por uma lista de jornadas diárias de trabalho, sendo cada jornada constituída por uma lista de tarefas.

3.2. Estruturas de Vizinhança

Dada uma solução s, para atingir uma solução s', onde s' é dito vizinho de s, são usados três tipos de movimentos: Realocação, Troca e *Link*, para definir, respectivamente, três estruturas diferentes de vizinhança, a saber: $N^R(s)$, $N^T(s)$, $N^L(s)$, conforme ilustra a Figura 1.

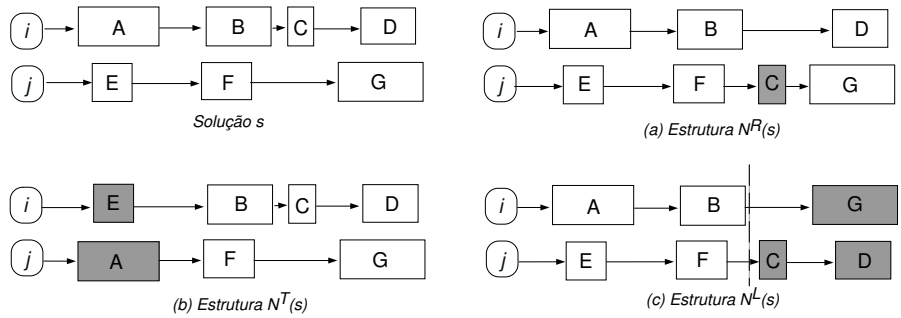

Figura 1: Estruturas de Vizinhança.

Nesta figura, A, $\cdots$, G representam tarefas a serem executadas nas jornadas i e j.

O movimento de realocação consiste em realocar uma tarefa de uma jornada para outra. Em (a), mostra-se que a tarefa C anteriormente pertencente à jornada i é realocada à jornada j. O conjunto de todos os vizinhos de s gerados através de movimentos de realocação define a estrutura de vizinhança $N^R(s)$.

Já um movimento de troca consiste na permuta de tarefas entre duas jornadas distintas. Em (b) mostra-se que as tarefas A e E são trocadas entre as jornadas i e j. O conjunto de todos os vizinhos de uma escala s gerados a partir de movimentos de troca define a estrutura de vizinhança $N^T(s)$.

Finalmente, o movimento *Link* consiste na troca de um conjunto de tarefas entre duas jornadas distintas i e j. Conforme ilustrado em (c), a partir de uma solução s define-se um ponto de corte em uma das jornadas, horário a partir do qual os blocos de tarefas serão trocados. O conjunto de todos os vizinhos de s gerados a partir de movimentos do tipo *Link* define a estrutura de vizinhança $N^L(s)$.

3.3. Função de Avaliação

Uma solução (ou escala) s é avaliada por uma função $f : s \in S \rightarrow \mathcal{R}$, onde S é o conjunto de todas as possíveis soluções. A função de avaliação adotada baseia-se na penalização de cada um dos requisitos essenciais e não essenciais não atendidos. Desse modo, uma escala s é avaliada com base em duas componentes, uma de inviabilidade ($g(s)$), a qual mede o não atendimento aos requisitos essenciais, e outra de qualidade ($h(s)$), a qual mede o não atendimento aos requisitos considerados não essenciais. A soma dessas duas componentes constitui a função de avaliação de uma solução s, que deve ser minimizada, isto é: $f(s) = g(s) + h(s)$.

Deve ser observado que uma solução s é viável se, e somente se, $g(s) = 0$. Nas componentes da função f, os pesos atribuídos às diversas medidas refletem a importância relativa de cada uma delas e, sendo assim, deve-se tomar um valor bem mais elevado para os requisitos essenciais, de forma a privilegiar a eliminação das soluções inviáveis.

3.4. Geração de uma solução inicial

A solução inicial é gerada tomando-se como base a filosofia adotada pela empresa na construção manual de suas jornadas. Esta metodologia baseia-se na repartição dos blocos dos veículos. Cada bloco de um veículo contém um conjunto de tarefas a serem executadas pelo veículo no dia. Este bloco é dividido seguindo dois critérios: sua duração e o fato de o veículo retornar ou não à garagem mais de uma vez durante sua jornada diária. Caso o veículo não retorne à garagem mais de uma vez, a divisão do bloco é feita de acordo com a sua duração. Nesta situação, o bloco é dividido ao meio caso cada metade não ultrapasse 9:10 horas de trabalho, que é o tempo máximo de trabalho permitido por lei, incluindo as 2:00 horas extras. Caso contrário, o bloco é dividido em três partes, sendo as duas primeiras de no máximo 7:10 horas e a última com as tarefas remanescentes. Cada parte de um bloco é alocada a uma jornada diferente.

Caso o veículo retorne à garagem durante sua jornada, a repartição do bloco também se dá conforme sua duração, porém a divisão é feita no intervalo de tempo em que a diferença entre o final da última tarefa e o início da primeira tarefa, excluindo o tempo em que o veículo ficou na garagem, for inferior a 8:40 horas. O tempo em que o veículo permanece na garagem é excluído porque o tripulante que receber essa jornada fará dupla pegada e, portanto, não necessitará dos 30 minutos de intervalo para repouso e/ou alimentação.

Uma das vantagens de se gerar a solução inicial seguindo a filosofia da empresa é que esta não contém sobreposições de tarefas nem trocas de pontos ou linhas proibidas. Todavia, pode ocorrer excesso na duração das jornadas diárias de trabalho.

3.5. *Simulated Annealing* aplicado ao PPT

Simulated Annealing (SA) é uma técnica de busca local probabilística, proposta originalmente em [12], que admite soluções de piora para escapar de ótimos locais. O algoritmo parte de uma solução inicial s gerada conforme descrito na seção 3.4..

Na seqüência, duas jornadas são sorteadas e é gerado um vizinho qualquer s' de acordo com uma certa estrutura de vizinhança $N(s)$, conforme se relata ao final desta seção. Se o valor da função de avaliação $f(s')$ do vizinho for menor que o da solução corrente $f(s)$ e gerar uma variação de energia $\Delta = f(s') - f(s) \leq 0$, este vizinho é aceito e passa a ser a nova solução corrente. Caso contrário, ou seja, se ocorrer um movimento de piora, a solução pode ser aceita com uma probabilidade definida por $e^{-\Delta/T}$, sendo T um parâmetro do método. Este parâmetro, conhecido como temperatura, regula a probabilidade de se aceitar soluções de piora. A temperatura T é inicializada com um valor T_0 elevado. Após um número fixo de iterações *SAmax* (o qual representa o número de iterações necessárias para o sistema atingir o equilíbrio térmico em uma dada temperatura), a temperatura é gradativamente diminuída por uma razão de resfriamento α, tal que $T_n \leftarrow \alpha \times T_{n-1}$, sendo $0 \leq \alpha \leq 1$. Com esse procedimento, existe, no início uma chance maior para escapar de mínimos locais e, à medida que T aproxima-se de zero, o algoritmo comporta-se como o método de descida, uma vez que diminui a probabilidade de se aceitar movimentos de piora ($\mathrm{T} \rightarrow 0 \Longrightarrow e^{-\Delta/T} \rightarrow 0$). O método *SA* é interrompido quando a temperatura T for menor do que um dado valor positivo T_{final}. Para o PPT foram desenvolvidos dois métodos: SA-R e SA-RTL. O primeiro explora o espaço de soluções apenas com a estrutura de vizinhança $N^R(s)$, enquanto o segundo utiliza uma vizinhança $N(s)$ dada pela união de todas as três estruturas de vizinhança definidas na seção 3.2., isto é, $N(s) = N^R(s) \bigcup N^T(s) \bigcup N^L(s)$. Um vizinho s' da solução corrente s é gerado com uma chance de $PercN^R\%$, $PercN^T\%$ e $PercN^L\%$, de pertencer à vizinhança $N^R(s)$, $N^T(s)$ e $N^L(s)$, respectivamente.

3.6. Busca Tabu aplicado ao PPT

De forma semelhante ao método *Simulated Annealing*, Busca Tabu é um procedimento de otimização local que admite soluções de piora. Em sua forma clássica, a cada iteração procura-se um ótimo local selecionando-se o melhor vizinho s' de um subconjunto V da vizinhança $N(s)$ da solução corrente s. Visto que no PPT a análise completa da vizinhança é uma operação custosa computacionalmente, a Busca Tabu foi implementada analisando a cada iteração somente um certo percentual *PercViz* da vizinhança corrente. Isto é, apenas um subconjunto das jornadas, aleatoriamente escolhidas a cada iteração do método, é analisada. De forma a reduzir ainda mais o esforço computacional na exploração dessa vizinhança, interrompe-se a busca quando uma solução de melhora é encontrada. Nesse método, independentemente de $f(s')$ ser melhor ou pior que $f(s)$, s' será sempre a nova solução corrente. Entretanto, apenas esse mecanismo não é suficiente para escapar de ótimos locais, uma vez que pode haver retorno a uma solução previamente gerada. Para prevenir isso, o algoritmo usa o conceito de lista tabu. Esta lista é normalmente constituída por atributos dos movimentos realizados, atributos esses que sejam fáceis de detectar e que possam prevenir o retorno a soluções geradas recentemente. Movimentos que contenham esses atributos ficam proibidos de serem feitos por um certo número $|T|$ de iterações, conhecido como *tempo tabu*. Para o PPT foram desenvolvidos dois métodos: BT-R e BT-RT. O primeiro explora o espaço de soluções trabalhando apenas com a estrutura de vizinhança $N^R(s)$, en-

quanto o segundo considera duas estruturas de vizinhança, $N^R(s)$ e $N^T(s)$, cada qual explorada por $BTMax$ iterações sem melhora, nesta sequência. Quando um movimento é feito, seja ele de realocação ou de troca, armazena-se na lista tabu a dupla $< JOR_{fonte}, TA >$, onde JOR_{fonte} é a jornada de onde foi retirada a tarefa TA para ser realocada a uma outra jornada. No caso de o movimento ser de troca, apenas o índice de uma das duas jornadas envolvidas e a respectiva tarefa escolhida são armazenados. Desta forma, por $|T|$ iterações não será permitido realocar a tarefa TA à jornada JOR_{fonte}. Como a lista tabu pode ser muito restritiva, é comum usar um critério de aspiração, isto é, permitir a realização de um movimento, ainda que contenha um atributo tabu, se satisfazer a um certo critério. Para o problema abordado, consideramos o critério de aspiração por objetivo. Assim, um movimento ainda que tabu é realizado se produzir uma solução de valor menor do que o da melhor solução encontrada até então. Visto que mesmo utilizando uma lista tabu pode ocorrer ciclagem, utilizou-se uma lista tabu de tamanho dinâmico para minimizar essa possibilidade de retorno a soluções anteriormente geradas pelo método. Essa lista varia de tamanho entre $|T_{min}|$ e $|T_{max}|$ a cada $IterT$ iterações. Detalhes adicionais desse método podem ser encontrados em [10].

3.7. VNS aplicado ao PPT

O Método de Pesquisa em Vizinhança Variável, conhecido como VNS (*Variable Neighborhood Search*), proposto em [16], é um método de busca local que consiste em explorar o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança. A idéia do método é a de explorar vizinhanças gradativamente mais 'distantes' da solução corrente, focando a busca em torno de uma nova solução se, e somente se, um movimento de melhora é realizado. O método VNS também inclui um procedimento de busca local a ser aplicado sobre o vizinho da solução corrente.

Para resolver o PPT, foi desenvolvido o método VNS-RTL, o qual utiliza as três estruturas de vizinhança, $N^R(s)$, $N^T(s)$ e $N^L(s)$, nesta sequência, definidas na Seção 3.2.. Esta ordem de exploração foi considerada de forma a minimizar o esforço computacional na exploração das vizinhanças, as quais estão em ordem crescente de complexidade. Para minimizar ainda mais esse esforço e evitar a análise completa de uma dada vizinhança em cada iteração, interrompe-se a busca na vizinhança corrente sempre que ocorrer uma solução de melhora.

O procedimento de busca local implementado foi o Método de Descida em Vizinhança Variável, conhecido como VND (*Variable Neighborhood Descent*), também proposto em [16]. Este método é interrompido quando for atingida a última estrutura de vizinhança e nenhuma melhora na solução corrente for possível. Ao contrário do método VND, o método VNS tem condições de prosseguir a busca quando essa última situação ocorre, uma vez que retorna-se à primeira estrutura de vizinhança e seleciona-se um outro vizinho qualquer até que uma determinada condição de parada seja satisfeita. A condição de parada adotada foi o tempo de processamento.

4. Resultados

Os algoritmos foram desenvolvidos na linguagem C++ usando o compilador Borland C++ Builder 6.0 e testados em um microcomputador com processador Pentium IV, 1.8 MHz, com 256 MB de memória RAM, sob sistema operacional Windows XP. O problema considerado para teste foi relativo a um dia útil de uma empresa de transporte público, situada em Belo Horizonte, responsável por 11 linhas de ônibus, com frota empenhada de 111 veículos.

Os pesos utilizados para penalizar os diversos requisitos da função de avaliação de uma solução foram os mesmos utilizados em [19]. O critério de parada adotado nos testes foi o tempo total de execução requerido para os algoritmos baseados em *Simulated Annealing* congelarem, a saber, 60 minutos.

Cada algoritmo desenvolvido foi submetido a uma bateria preliminar de testes com o objetivo de calibrar os diversos parâmetros, a saber: tamanho mínimo da lista tabu $|T_{min}| = 250$; tamanho máximo da lista tabu $|T_{max}| = 260$; número de iterações em que a lista tabu muda de tamanho $IterT = 150$; percentual da vizinhança analisada a cada iteração dos métodos baseados em Busca Tabu $PercViz = 50\%$; Número de iterações sem melhora com o método de Busca Tabu $BTMax = 500$; temperatura inicial T_0= 1.000.000; temperatura final $T_{final} = 0{,}01$; número máximo de iterações a uma dada temperatura $SAmax = 219 \times 733 = 160.527$ (correspondendo ao número de jornadas utilizadas pela empresa vezes o número de tarefas); taxa de resfriamento $\alpha = 0{,}975$; chance de a estrutura $N^R(s)$ ser escolhida no método SA-RTL $PercN^R = 80\%$; chance de as estruturas $N^T(s)$ e $N^L(s)$ serem escolhidas no método SA-RTL $PercN^T = PercN^L = 10\%$; ponto de corte para o movimento $Link = 6$ horas a partir do início de uma jornada.

Cada algoritmo foi executado 10 vezes, cada qual partindo de uma semente diferente de números aleatórios. A Tabela 1 apresenta, para cada algoritmo testado, o melhor valor obtido e o desvio do valor médio em relação ao melhor valor encontrado, isto é: desvio = ((Valor Médio - Melhor Valor))/(Melhor Valor).

Tabela 1: Desempenho dos Algoritmos.

Método	SA-R	SA-RTL	BT-R	BT-RT	VNS-RTL
Melhor Valor	1.399.400	1.197.820	1.204.240	1.148.380	**1.145.720**
Desvio	28,9%	8,1%	6,8%	4,8%	**1,1%**

A Tabela 1 mostra que o método VNS-RTL obteve o menor valor para a função de avaliação. Além disso, esse método se mostrou o mais robusto, pois foi o que apresentou o menor desvio em relação à melhor solução encontrada. O método BT-RT também alcança soluções finais de boa qualidade, entretanto é menos robusto do que o método VNS-RTL.

A Tabela 2 apresenta algumas das características da solução em uso na empresa estudada, bem como as características das melhores soluções produzidas pelos algoritmos mais eficientes.

Tabela 2: Características das soluções geradas pelos melhores algoritmos.

	Empresa	SA-RTL	BT-R	BT-RT	VNS-RTL
# Jornadas	219	218	218	218	217
Hora extra (hh:mm)	116:00	87:42	87:33	88:34	89:45
Trocas de veiculos	0	33	33	22	24
Trocas de linhas	0	18	25	15	15

Pode-se observar que as soluções obtidas com os métodos heurísticos reduziram o total de horas extras diárias pagas pela empresa bem como o número de jornadas e, portanto, de tripulações necessárias para a operação da frota. Tais reduções foram possíveis devido à flexibilização da operação, com a realização de troca de veículos e de linhas.

A Figura 2 ilustra a evolução típica do valor da melhor solução nos métodos VNS-RTL, BT-RT e SA-RTL ao longo do tempo no processo de busca. Pode-se observar que o método VNS-RTL alcança soluções de boa qualidade mais rapidamente que os outros dois métodos.

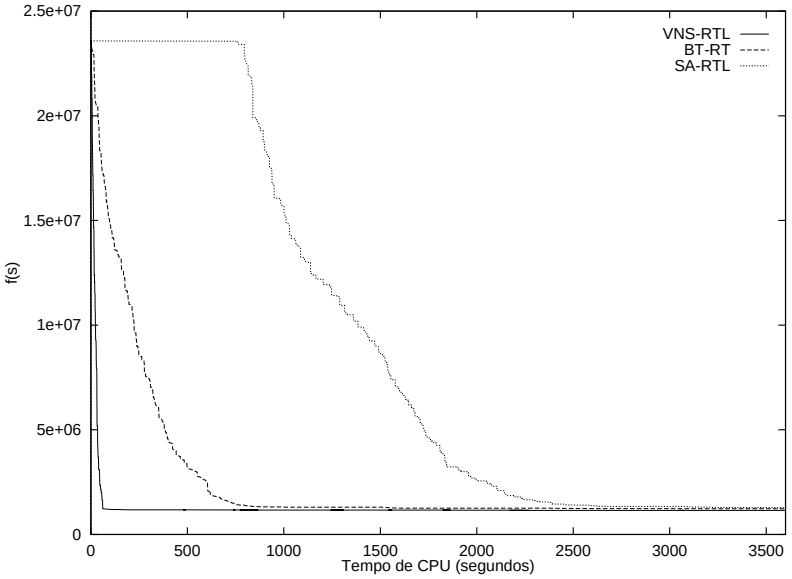

Figura 2: Evolução da melhor solução nos métodos VNS-RTL, BT-RT e SA-RTL.

5. Conclusões

Este trabalho contribui com o desenvolvimento de modelos heurísticos baseados em *Simulated Annealing*, Busca Tabu e Método de Pesquisa em Vizinhança Variável aplicados ao problema de programação de tripulações de ônibus urbano considerando a realidade brasileira. O Método de Pesquisa em Vizinhança Variável, partindo de

uma solução inicial seguindo a filosofia da empresa, mostrou ser o mais eficiente na abordagem do problema. Além de gerar soluções de boa qualidade mais rapidamente, foi o mais robusto dos métodos e o que produziu a solução final de melhor qualidade, com redução no número de tripulações e horas-extras, dois dos principais itens na planilha de custos com a mão-de-obra operacional. Adicionalmente, produziu jornadas com características próximas daquelas adotadas pela empresa, facilitando sua implementação operacional. Uma outra vantagem desse método é que, ao contrário das outras duas metodologias consideradas, ele depende apenas das estruturas de vizinhança desenvolvidas, não tendo parâmetros a calibrar. Ressalta-se, entretanto, que sua superioridade é válida apenas em relação aos outros métodos com as características aqui apresentadas.

Agradecimentos

Os autores agradecem ao CNPq e à Universidade Federal de Ouro Preto pelo apoio dado ao desenvolvimento deste trabalho, bem como à Borland Latin America pela cessão de uma licença de uso do software C++ Builder 6.0.

Abstract. This work deals with the Bus Crew Scheduling Problem (BCSP) related to a company operating in a public mass transit. Such problem consists in assigning the set of all vehicle trips of a given company to a set of drivers with minimal cost. The solution of BCSP is a set of driver duties. In this work the BSCP was solved through the Simulated Annealing, Tabu Search and Variable Neighborhood Search metaheuristics. These methods explore the solution space using different neighborhoods, which modify the driver duties through tasks' movements. Each schedule is evaluated by a function based on penalties that has as goal to satisfy the labor agreement rules, the operational rules of the company, as well as to optimize the use of the crew at work. The algorithms were tested with real data provided by a company operating in Belo Horizonte city.

Referências

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

[2] C.F. Bouzada, "Análise das despesas administrativas no custo do transporte coletivo por ônibus no município de Belo Horizonte", Dissertação de mestrado, Fundação João Pinheiro, MG, 2002.

[3] R. Clement e A. Wren, Greedy genetic algorithms, optimizing mutantis and bus driver scheduling, em "Computer-Aided Transit Scheduling" (J.R. Daduna, I. Branco e J.M.P. Paixão, eds.), pp. 213-235, Springer-Verlag, 1995.

[4] J.R. Daduna, I. Branco e J.M.P. Paixão (eds), "Computer-Aided Transit Scheduling", Proceedings of the 5th International Workshop on Computer-Aided Scheduling of Public Transport, Springer-Verlag, 1995.

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

[6] S.E.G. Elias, The use of digital computers in the economic scheduling for both man and machine in public transport. Technical Report 49, Kansas State University Bulletin, Kansas, EUA.

[7] T.A. Feo e M.G.C. Resende, Greedy randomized adaptive search procedures, *Journal of Global Optimization*, **6** (1995), 109-133.

[8] S. Fores, L. Proll e A. Wren, An Improved ILP System For Driver Scheduling, em "Computer-Aided Transit Scheduling" (N.H.M. Wilson, ed.), pp. 43-61, Springer-Verlag, 1999.

[9] C. Friberg e K. Haase, An exact branch and cut algorithm for the vehicle and crew scheduling problem, em "Computer-Aided Transit Scheduling" (N.H.M. Wilson, ed.), pp. 63-80, Springer-Verlag, 1999.

[10] F. Glover e M. Laguna, "Tabu Search", Kluwer Academic Publishers, 1997.

[11] D.E. Goldberg, "Genetic Algorithms in Search", Optimization and Machine Learning, Addison-Wesley, Berkeley, 1989.

[12] S. Kirkpatrick, D.C. Gellat e M.P. Vecchi, Optimization By Simulated Annealing, *Science*, **220** (1983), 671-680.

[13] A.S.K. Kwan, R.K. Kwan e A. Wren, Driver scheduling using genetic algorithms with embedded combinatorial traits, em "Computer-Aided Transit Scheduling" (N.H.M. Wilson, ed.), pp. 81-102, Springer-Verlag, 1999.

[14] H.R. Lourenço, J.P. Paixão e R. Portugal, Multiobjective metaheuristics for the bus-driver scheduling problem, *Transportations Science*, **35** (2001), 331-343.

[15] B. Manington e A. Wren, A general computer method for bus crew scheduling, em "Workshop on Automated Techniques for Scheduling of Vehicle operators for Urban Public Transportation Services", Chicago, 1975.

[16] N. Mladenovic e P. Hansen, Variable Neighborhood Search, *Computers and Operations Research*, **24** (1997), 1097-1100.

[17] Y. Shen e R.S.K. Kwan, Tabu Search for driver scheduling, em "Computer-Aided Scheduling of Public Transport" (S. Voβ e J. Daduna, eds.), pp. 121-135, Springer-Verlag, 2001.

[18] P.H. Siqueira, "Aplicação do algoritmo do matching no problema da construção de escalas de motoristas e cobradores de ônibus", Dissertação de mestrado, Setor de Tecnologia e de Ciências Exatas, UFPR, PR, 1999.

[19] G.P. Silva, M.J.F. Souza e J.M.C.B. Alves, Resolução do problema de programação diária da tripulação de ônibus urbano via Simulated Annealing, em "XVI Congresso de Pesquisa e Ensino em Transportes", Panorama Nacional de Pesquisa em Transportes, Vol. 2, pp. 95-104, ANPET, 2002.

[20] B.M. Smith e A. Wren, A Bus Crew Scheduling System Using A Set Covering Formulation, *Transportation Research*, **22a** (1988), 97-108.

[21] S. Voβ e J.R. Daduna (eds), "Computer-Aided Transit Scheduling", Proceedings of the 9th International Workshop on Computer-Aided Scheduling of Public Transport, Springer-Verlag, 2001.

[22] N.H.M. Wilson (ed), "Computer-Aided Transit Scheduling", Proceedings of the 7th International Workshop on Computer-Aided Scheduling of Public Transport, Springer-Verlag, 1999.

[23] A. Wren e J.M. Rousseau, Bus driver scheduling - An overview, em "Computer-Aided Transit Scheduling" (J.R. Daduna, I. Branco e J.M.P. Paixão, eds.), pp. 173-187, Springer-Verlag, 1995.

[24] A. Wren e D.O. Wren, A genetic algorithm for public transport driver scheduling, *Computer and Operations Research*, **22** (1995), 101-110.