UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

Programação integrada de veículos e tripulações de ônibus urbano

Marcone Jamilson Freitas Souza /DECOM/UFOP - Coordenador

Anexo do Relatório Técnico Científico Final apresentado ao CNPq, referente ao processo CNPq, 474831/2007-8, desenvolvido no período dezembro de 2007 a dezembro de 2009.

Ouro Preto – Minas Gerais - Brasil
Janeiro de 2010

Problema de programação integrada de veículos e tripulações de ônibus urbano

RESUMO

Este trabalho trata da resolução integrada de duas etapas do planejamento de transportes no Sistema de Transporte Público, no caso, a programação de veículos (PPV) e a programação de tripulações (PPT). O primeiro problema consiste em criar uma rotina diária de operação para uma frota de veículos disponibilizada para uso de uma empresa, a partir de um conjunto de viagens com seus respectivos horários, pontos de partida e chegada, e tempo de viagem. O objetivo do PPV é reduzir o custo operacional de tal atividade, fazendo o melhor aproveitamento da frota. O PPT recebe como dados de entrada a programação dos veículos e retorna como resultado um conjunto de jornadas de trabalho para as tripulações (motoristas e trocadores). Estes problemas são da classe NP-difícil e, assim, ainda não há algoritmos em tempo polinomial para encontrar soluções ótimas. Aqui, estes problemas são tratados conjuntamente e alterações realizadas na programação de veículos têm reflexo na programação das tripulações. Dada a natureza combinatória dos problemas considerados e, em particular, do problema integrado, este é resolvido por meio de métodos heurísticos. Neste trabalho, foi desenvolvido um método híbrido, baseado nas metaheurísticas *Iterated Local Search* (ILS), *Variable Neighborhood Descent* (VND) e Busca Tabu (BT). Para testar o algoritmo proposto são utilizados dados reais das empresas do Sistema de Transporte Público da cidade de Belo Horizonte (MG), que atuam em uma bacia de linhas de tal cidade. Duas funções de avaliação foram consideradas: uma baseada em pesos empíricos e outra, em custos. Os resultados obtidos são comparados com aqueles produzidos por uma metodologia clássica de solução dessas duas etapas de solução do PPVT, que consiste em resolver os problemas PPV e PPT de forma independente e seqüencial. Os experimentos computacionais realizados mostram que o algoritmo proposto é superior, além de robusto.

Índice

Lista de Figuras

Lista de Tabelas

1 Introdução

1.1 O Problema de Programação de Veículos e Tripulações

A Figura 1 representa as relações existentes entre os principais problemas que surgem no processo de planejamento do Sistema de Transporte Público.

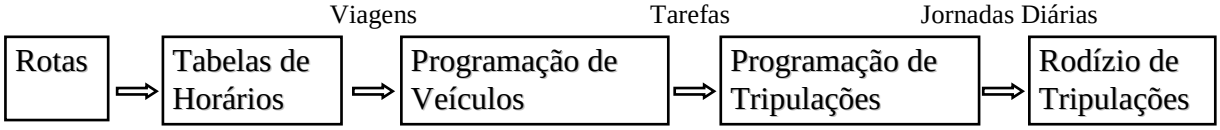

Figura 1.1 – Relação entre as diversas operações no Sistema de Transporte Público

A definição das rotas (linhas) e das tabelas de horários do sistema se baseia na disponibilidade de infra-estrutura, nos serviços requeridos pelos usuários e em aspectos de demanda. Tais decisões não fazem parte da abrangência deste projeto e são consideradas como conhecidas. As viagens de uma mesma rota podem ter suas durações alteradas ao longo do período de planejamento, entretanto, suas durações são consideradas também como conhecidas de forma determinística. As duas próximas etapas do processo são as programações dos veículos e das tripulações, que consistem em atribuir os veículos às viagens e as tripulações aos veículos, respectivamente.

Conhecidas as viagens a serem executadas pela empresa concessionária do serviço de transporte público, o problema seguinte é o da "Programação dos Veículos". Neste, o objetivo é fazer o dimensionamento da frota e a alocação dos veículos às viagens, de forma que todas as viagens programadas para as diversas linhas da empresa sejam executadas. Mais precisamente, o Problema de Programação de Veículos (PPV) pode ser definido da seguinte forma: dado um conjunto de viagens com seus respectivos horários e pontos de partida e chegada, e dado o tempo de viagem entre todos os pares de pontos, encontrar a programação de veículos de custo mínimo tal que: 1) cada viagem seja executada por um único veículo, e 2) a cada veículo seja atribuída uma seqüência exeqüível de viagens. Usualmente, a função de custo é uma combinação dos custos fixos (custo do veículo) e custos variáveis (tempo de viagem fora de operação e tempo parado no terminal). A programação dos veículos resulta em *blocos de viagens*, onde cada bloco consiste na seqüência de viagens realizadas por um veículo entre uma partida e o seu retorno à garagem.

Na abordagem seqüencial assume-se conhecida a programação dos veículos para resolver a programação de tripulações, ou seja, são conhecidos os blocos de viagens dos veículos. Nesta etapa, cada bloco de viagens de um veículo é subdividido em *oportunidades de troca*, que são definidas através de um ponto (possível local de troca) e um intervalo de tempo, que juntos possibilitam que ocorra a troca da tripulação em operação por uma nova tripulação. Com base nas oportunidades de troca, as viagens são agrupadas em *tarefas*. Uma *tarefa* é definida como um conjunto de viagens compreendido entre duas oportunidades de troca consecutivas, e corresponde à menor porção de trabalho que pode ser designada a uma tripulação. Com base nessas definições, o Problema de Programação de Tripulações (PPT) pode ser definido da seguinte forma: dado um conjunto de tarefas, encontrar um conjunto de jornadas diárias com custo mínimo tal que 1) cada tarefa seja atribuída a uma única jornada, e

2) cada jornada seja uma seqüência de tarefas que pode ser executada por uma única tripulação respeitando as restrições operacionais e trabalhistas. Dentre essas restrições, destacamos: a) a duração da jornada diária de trabalho não pode superar um valor máximo; b) durante a jornada de trabalho deve haver um intervalo de tempo para descanso e alimentação; (c) as trocas de tripulações só podem ocorrer em determinados locais (pontos). Usualmente, a função de custo do PPT considera a minimização do número de jornadas, da quantidade de horas-extras existentes nas jornadas e da quantidade de trocas de tripulações, linhas e veículos. Para detalhes sobre a modelagem e a resolução de problemas desta natureza no âmbito nacional ver Silva *et al*. (2002, 2004), Souza *et al*. (2003a, 2003b, 2004a, 2004b).

Uma vez geradas as jornadas diárias de dias úteis, sábados, domingos e feriados, essas devem ser, agora, combinadas, de forma a fazer a escala mensal das tripulações ou Rodízio das Tripulações, problema conhecido na literatura inglesa como *Crew rostering*. Na elaboração da escala mensal, observa-se que uma parte da legislação trabalhista, dos acordos coletivos e das regras operacionais foi contemplada na confecção das jornadas diárias. Entretanto, outras tantas devem ser observadas, como por exemplo: (a) A cada 6 (seis) dias de trabalho o tripulante tem direito a uma folga; (b) Tripulantes que fazem dupla pegada têm folga obrigatoriamente aos sábados ou domingos. Em uma escala mensal procura-se, dentre outros objetivos: (a) minimizar, ou mesmo eliminar, as mudanças de horários de trabalho de uma tripulação; (b) equilibrar as horas mensais trabalhadas pelas tripulações; (c) reduzir as mudanças de linhas de uma tripulação ao longo do mês. Experiências com a modelagem e resolução desse problema considerando o cenário nacional são relatadas em Toffolo *et al*. (2005).

Pela Figura 1 observa-se que a solução de um problema de uma etapa do processo de planejamento do Sistema de Transporte Público depende da solução do problema da etapa anterior. Assim, na abordagem seqüencial de resolução dos problemas, se fosse possível garantir a solução ótima para o problema de cada etapa, não haveria garantia de que essa solução fosse ótima para o problema como um todo. O planejamento poderia ser mais eficiente se fosse adotada uma abordagem integrada de resolução dos problemas. Entretanto, devido à complexidade dos modelos matemáticos que consideram o problema como um todo, a grande maioria dos estudos teóricos trata os problemas de forma seqüencial.

Mais recentemente têm surgido na literatura abordagens que lidam com a integração de duas etapas consecutivas do processo de planejamento, a programação dos veículos e a programação de tripulações.

Embora haja um entendimento de que a abordagem integrada da programação de veículos e tripulações seja vantajosa, a maioria dos trabalhos encontrados na literatura segue a abordagem seqüencial, na qual a programação de veículos é definida a priori e independentemente da programação de tripulações.

O Problema de Programação Integrada de Veículos e Tripulações (PPVT), objeto de estudo neste trabalho, pode ser definido da seguinte maneira: dada uma tabela de horários de viagens a serem realizadas, encontrar a programação de custo mínimo para os veículos e as tripulações tal que ambas sejam factíveis e mutuamente compatíveis.

Os problemas dos veículos e das tripulações interagem entre si, uma vez que a especificação da programação dos veículos impõe certas restrições à programação das tripulações e vice-versa. Assim, a abordagem integrada do Problema de Programação de Veículos e de Tripulações sempre permitirá obter resultados superiores ou iguais aos resultados da abordagem seqüencial (Freling *et al*. 2003).

1.2 Estrutura do trabalho

Este trabalho está estruturado em sete capítulos, incluindo esta introdução.

Os objetivos do projeto estão apresentados no capítulo 2.

No capítulo 3 é apresentada uma revisão bibliográfica sobre alguns trabalhos desenvolvidos e suas técnicas utilizadas para resolver o PPV e o PPT. São descritos também os princípios básicos das heurísticas e metaheurísticas empregadas neste trabalho.

O capítulo 4 descreve o Problema de Programação de Veículos e de Tripulações segundo as políticas operacionais que regem o Sistema de Transporte Público da cidade de Belo Horizonte, objeto de estudo neste projeto.

A metodologia adotada neste trabalho é apresentada no capítulo 5. Aborda-se, em tal capítulo, a forma de representação de uma solução, os tipos de movimentos que constituem as estruturas de vizinhança, a forma de avaliação de uma solução e o algoritmo de exploração do espaço de busca.

No capítulo 6 são apresentados os resultados obtidos.

No capítulo 7 são apresentadas as conclusões.

2 Objetivos do Trabalho

Este trabalho tem como objetivo o desenvolvimento de um algoritmo heurístico, baseado nas metaheurísticas *Iterated Local Search*, Descida em Vizinhança Variável e Busca Tabu, para resolver de forma eficiente o Problema de Programação Integrada de Veículos e Tripulações de Ônibus Urbano.

Objetiva-se, também, comparar os resultados obtidos com o algoritmo proposto com aqueles produzidos por uma metodologia clássica de solução dessas duas fases do planejamento de transporte. Isto é, com um algoritmo seqüencial tradicional, o qual inicialmente resolve a programação de veículos e depois, com a solução desta, resolve o problema de programação de tripulações.

3 Revisão Bibliográfica

3.1 Introdução

Esta seção apresenta uma revisão de literatura do Problema de Programação de Veículos e, em seguida, do Problema de Programação de Tripulações; posteriormente, faz uma revisão da classificação de Freling *et al*. 1999 para o PPVT e da integração entre esses dois problemas.

O Problema de Programação de Veículos foi um dos primeiros problemas de transporte público a serem estudados e resolvidos com o auxílio de computadores. Os primeiros trabalhos tratavam o problema tanto utilizando métodos exatos (Kirkman, 1968) quanto métodos heurísticos (Saha, 1970; Wren, 1972).

Martin-löf (1970) resolve o problema de programação de veículos considerando viagens com flexibilidade nos horários de partida. É apresentado um modelo de programação linear inteira, o qual é resolvido por um método simples de busca do tipo *branch-and-bound*. O método, entretanto, tem aplicabilidade limitada a problemas de pequeno porte.

Gavish *et al*. (1978) formulam o PPV como um problema de designação e pseudo-designação, que tem como objetivos primários minimizar o número de veículos utilizados durante o horário de pico e minimizar o tempo de deslocamento fora de operação no horário entre picos. O objetivo secundário é minimizar as mudanças em relação à programação já existente.

Hoffstadt (1981) utiliza uma heurística de encadeamento de viagens para determinar o número mínimo de veículos. Posteriormente, para minimizar o custo operacional é aplicado o algoritmo húngaro (Kuhn, 1956) a um problema de designação.

Carraresi e Gallo (1984) tratam o problema de programação de veículos de natureza periódica, isto é, após um determinado período, por exemplo, 24 horas, todas as viagens devem ser executadas novamente. Dessa forma, são praticadas apenas três programações diferentes: uma para os dias úteis, uma para os sábados e uma para os domingos e feriados.

Wren e Chamberlain (1988) incorporam a programação da tripulação ao sistema para programação de veículos, denominado VAMPIRES, de Wren (1972), que passa então a ser denominado BUSMAN, sendo a componente responsável pela programação de veículos denominada BUSPLAN.

Kwan e Rahin (1995) apresentam os módulos incorporados à componente BUSPLAN do sistema, os quais permitem estabelecer certa interatividade com o usuário. São apresentadas ferramentas semi-automáticas para coordenar a geração de novos horários das viagens com ligação infactível, além de melhoramentos nas heurísticas que tratam da programação com diferentes tipos de veículos.

Kwan e Rahin (1999) apresentam as características do BOOST, que é a versão orientada a objetos do sistema anterior, o BUSPLAN. Nesse sistema, uma solução inicial é gerada por meio de um método construtivo guloso que admite infactibilidade. A seguir, a programação é refinada por um procedimento iterativo 2-*optimal*.

Psarras *et al*. (1997) utilizam as vantagens da *Constraint Logic Programming*, para gerar uma solução inicial para o Problema de Programação de Veículos e posteriormente

avaliar as várias soluções intermediárias obtidas por uma busca local com o objetivo de verificar a conformidade com as restrições impostas ao problema. Segundo os autores, a grande vantagem desta abordagem é a obtenção de soluções em um tempo de execução satisfatório.

Em Atkinson (1998) uma heurística GRASP é aplicada a um PPV em que os intervalos de desembarque e embarque de um veículo são reduzidos, além de outras restrições adicionais. Duas formas adaptativas de busca são propostas, uma global e outra local. Em ambas, o cálculo da função gulosa é modificado por uma quantidade que mensura heuristicamente a qualidade da solução parcial que é obtida quando uma decisão é tomada. A busca global é um procedimento adaptativo tratado como uma regra de aprendizado, uma vez que ele tenta aprender com os erros anteriores fazendo uma atualização da função gulosa a cada iteração da heurística.

Baita *et al*. (2000) abordam um caso prático do PPV comparando o desempenho do modelo tradicional de designação com duas novas heurísticas propostas para o problema. O modelo de designação é resolvido com o algoritmo húngaro descrito por Carpaneto *et al*. (1988). Neste modelo, a estrutura da função de custo considera os seguintes elementos: viagens mortas entre os terminais, mudanças de linhas e tempo de espera nos terminais. A primeira heurística proposta é baseada na técnica de Inteligência Artificial denominada Programação em Lógica. Uma vez que o PPV pode ser visto como um problema de otimização multiobjetivo (minimizar frota, tempo de viagens mortas e tempo de terminal) a heurística tenta satisfazer os diferentes critérios na ordem que segue. O algoritmo tenta conectar as viagens mais próximas que pertencem à mesma linha e ao mesmo terminal, satisfazendo os tempos mínimos e máximos de terminal. Se a primeira estratégia não cobrir todas as viagens, então a restrição de pertinência à mesma linha é retirada, e são conectadas viagens que partem do mesmo terminal. Se for necessário, são conectadas viagens que partem de diferentes terminais, iniciando a busca pelos terminais mais próximos. No algoritmo genético proposto pelos autores, uma programação é representada por um gene, sendo que cada alelo (valor dentro do vetor) representa um veículo, e sua posição representa a viagem atribuída ao veículo. Para manter a factibilidade da solução após uma operação de cruzamento, aplica-se um mecanismo específico de reparação. A escolha dos indivíduos a serem reproduzidos na próxima geração é feita utilizando dois procedimentos não convencionais, além do método tradicional, segundo o qual a probabilidade de um indivíduo ser selecionado é proporcional ao valor da função objetivo.

Silva (2001) explora modelos de fluxo em redes para resolver o PPV. Duas técnicas de redução de rede são apresentadas. A primeira combina o modelo de fluxo com custo mínimo à técnica de redução de arcos, denominada Eliminação dos Arcos Longos. Apesar da redução significativa da rede, o método se mostra limitado na sua aplicabilidade. A segunda abordagem combina a técnica de Geração de Arcos Longos com o problema de pseudo-designação. A adaptação feita, denominada Geração de Arcos, considera, além da minimização do número de veículos, a redução do tempo de viagem morta e de terminal. O modelo não considera o tempo máximo de operação do veículo e se mostra pouco flexível para a inclusão de outras restrições operacionais, limitando sua aplicação.

O Problema de Programação de Tripulações de ônibus urbano (PPT) trata da geração de escalas de trabalho de motoristas e cobradores. 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. O PPT pertence a uma classe de problemas combinatoriais chamada programação de pessoal (*staff scheduling ou rostering*). Tais problemas são de difícil resolução, uma vez que apresentam um grande número de restrições operacionais e trabalhistas. Além disso, pertencem à classe de problemas NP-difíceis. Sendo assim, o uso

exclusivo de métodos exatos limita-se a resolver problemas de pequenas dimensões. Para problemas de dimensões maiores, a abordagem mais comum é através de heurísticas.

Marinho (2005) resolve o PPT por um método Busca Tabu com Relaxação Adaptativa que explora o espaço de busca por meio de duas estruturas de vizinhança: realocação e troca.

A representação de uma solução para o PPV pode ser facilmente adaptada para representar uma solução do PPT. Embora as restrições sejam diferentes, os dois problemas podem ser reduzidos a problemas de designação. No PPV, busca-se uma boa alocação de viagens em veículos, enquanto isso, no PPT a meta é encontrar uma boa alocação de tarefas às jornadas de trabalho dos tripulantes. Dessa forma, as mesmas técnicas utilizadas na resolução do PPV podem ser adaptadas para resolver o PPT.

Grande parte dos trabalhos que abordam o PPVT o faz resolvendo o PPV de maneira exata enquanto utiliza uma heurística para resolver o PPT. Esses trabalhos têm como referência o procedimento proposto por Ball *et al.* (1983), que envolve a definição de uma rede de programação, consistindo de vértices caracterizados por conjuntos de viagens denominados *d-trips* que devem ser executadas por um veículo e uma tripulação. Diversos tipos de arcos podem ser agrupados em duas categorias, aqueles que indicam que uma tripulação e veículo seguem de uma *d-trip* para outra a aqueles que indicam que apenas a tripulação segue de uma *d-trip* para outra (arco_tripulações). O procedimento de resolução é decomposto em três componentes, enfatizando a programação da tripulação: construção dos pedaços de jornadas, melhoramento dos pedaços de jornadas e geração das jornadas de trabalho das tripulações. A programação dos veículos é gerada simultaneamente excluindo os arcos_tripulações e fixando os arcos usados pelos pedaços de jornadas na solução. Heurísticas similares são propostas por Tosini & Vercellis (1988) e Patrikalakis & Xerocostas (1992).

Trabalhos mais recentes modelam o PPVT como Problema de Programação Linear Inteira Zero-Um, acrescentando ao modelo de Pseudo-designação, para a programação dos veículos, variáveis de decisão e restrições que garantem que cada tarefa seja atribuída a uma jornada. A função de avaliação é composta por componentes de custo dos veículos e das jornadas de trabalho. Uma restrição de *link*, incluída no modelo, garante que cada tarefa seja coberta por uma jornada se o arco correspondente está na solução do PPV (Haase & Friberg, 1999). Freling *et al.* (2001) realizam uma aplicação prática, considerando diferentes regras operacionais e seus impactos sobre a redução de custos.

Em Silva *et al.* (2006a) e Silva *et al*. (2006b) relatam-se experiências com a resolução integrada do problema, no caso, introduzindo na programação de veículos algumas restrições da programação das tripulações. Os resultados obtidos mostraram que a abordagem integrada produziu soluções com qualidade superior àquelas geradas pela abordagem tradicional. Em uma instância considerada, a abordagem integrada utilizou um veículo a mais do que a tradicional, mas em contrapartida houve uma redução de 17 tripulações e ainda uma economia no número de horas extras pagas. Os autores lembram que o custo operacional das tripulações é superior ao custo operacional dos veículos, motivando o estudo de uma metodologia integrada para abordar os dois problemas.

Freling *et al.* (1999) classificam as abordagens de resolução do Problema de Programação de Veículos e Tripulações em: independente, seqüencial e integrada. Segundo os autores, a abordagem sequencial pode ser dividida em tradicional e inversa, enquanto que a abordagem seqüencial pode ser dividida nível um e nível dois.

A abordagem seqüencial tradicional consiste em resolver o PPV e em seguida o PPT. A abordagem seqüencial inversa é uma variação dessa abordagem, consistindo em inverter a

ordem de resolução dos problemas, isto é, resolve-se primeiramente o PPT e em seguida o PPV (Souza *et al*., 2004; Marinho *et al*., 2004; Silva *et al*., 2004; Atzingen, 2006).

As principais vantagens das abordagens seqüenciais são a facilidade de implementação e a diminuição da complexidade do problema. A abordagem seqüencial tradicional é recomendada para casos em que a programação dos veículos possui custo operacional superior ao custo operacional da programação das tripulações, e a inversa é indicada quando o custo operacional dos veículos é inferior ao das tripulações.

Na programação independente, a programação dos veículos é obtida ignorando-se a programação dos tripulantes e a programação dos tripulantes é obtida ignorando-se a programação dos veículos. Experiências desse tipo são retratadas em Bassi *et al*. 2007, que resolve o PPT sem considerar a programação dos veículos. O problema dessa abordagem é que, na maioria dos casos, os resultados obtidos são inviáveis na prática, pois as soluções obtidas para o PPV e para o PPT dificilmente são compatíveis entre si (Freling *et al.* 1999). Entretanto, essa abordagem permite determinar os limitantes inferiores para a solução ótima do PPVT.

A abordagem integrada começou a ser estudada mais recentemente, tendo em vista a evolução tecnológica dos computadores, uma vez que devido à grande complexidade do problema, o custo computacional desta abordagem era proibitivo.

A abordagem integrada nível um consiste em resolver simultaneamente o PPV e o PPT. Este é um problema complexo devido às suas grandes dimensões. Existem modelos computacionais para a sua resolução (Freling *et al*., 1999; Friberg e Haase, 1999; Haase *et al*., 2001), porém esses modelos não resolvem problemas de grandes dimensões em tempo aceitável devido às limitações computacionais existentes.

A abordagem integrada nível dois é baseada na resolução do PPV considerando características dos tripulantes de tal forma que a programação dos tripulantes é facilitada pela programação dos veículos.

Segundo Freling *et al*. 2003, a abordagem integrada do Problema de Programação de Veículos e de Tripulações sempre permitirá obter resultados superiores ou iguais aos resultados da abordagem seqüencial, e resultados inferiores ou iguais aos resultados da abordagem independente.

3.2 Heurísticas

As heurísticas são procedimentos de busca que têm suas pesquisas guiadas por meio da quantificação de proximidade a um determinado objetivo (solução). Para tanto, emprega-se a função de avaliação que se baseia nas instâncias, características e estrutura do problema para encontrar boas soluções, possivelmente a ótima, a um custo computacional aceitável sem, contudo garantir a otimalidade ou quão próxima dela a solução se encontra.

As heurísticas são classificadas em heurísticas construtivas ou de refinamento dependendo da estratégia adotada para se convergir para a solução desejada. Uma heurística construtiva tem por objetivo construir uma solução, elemento por elemento, sendo a forma de escolha de cada elemento a ser inserido a cada passo variável de acordo com a função de avaliação adotada. As heurísticas de refinamento consistem em técnicas que realizam a exploração do espaço de soluções, através da estrutura de vizinhança que pode ser definida da seguinte forma: seja S o espaço de pesquisa de um problema de otimização, $s \in S$ uma das possíveis soluções que pode ser obtida de uma heurística construtiva ou gerada aleatoriamente

e *f* a função objetivo a ser otimizada. O conjunto $N(s) \subseteq S$, o qual depende da estrutura do problema tratado, reúne um número determinado de soluções *s'*, denominado *vizinhança* de *s*. Cada solução $s' \in N(s)$ é chamada de *vizinho* de *s* e é obtido de *s* a partir de uma operação chamada de *movimento*.

O obstáculo a ser vencido pelas heurísticas é encontrar, em um tempo reduzido, uma solução tão próxima da ótima quanto possível.

A seguir serão apresentadas as heurísticas referenciadas ao longo deste trabalho.

3.2.1 Construção Gulosa

A Construção Gulosa é uma heurística construtiva que, a cada iteração, integra à solução sendo gerada aquele elemento que trará mais benefícios dentre todos os candidatos, conforme uma função de avaliação (função gulosa).

A estimativa do benefício da inserção de cada elemento candidato à solução em construção deve ser calculada a cada iteração, uma vez que, esta solução parcial sofre alterações sempre que um novo elemento passa a integrá-la e conseqüentemente a relevância dos possíveis candidatos também é modificada.

O pseudocódigo da heurística construtiva gulosa aplicada a um problema de minimização é apresentado a seguir na Figura 3.1.

Deve ser salientado que os passos para se construir uma solução dependem do problema abordado e a diferentes funções de avaliação estão associadas soluções gulosas distintas.

1) Seja C o conjunto de elementos candidatos a integrar a solução.
2) Seja *s* a solução sendo construída.
3) $s \leftarrow \emptyset$
4) enquanto $C \neq \emptyset$, faça
   a) $f(t_{min}) = min\{ f(t) \mid t \in C\}$
   b) $s \leftarrow s \cup \{t_{min}\}$
   c) Atualize o conjunto C de elementos candidatos.
5) Retorne a solução *s*.

Figura 3.1 – Pseudocódigo da Construção Gulosa

3.2.2 Método de Descida

O Método da Descida é uma técnica iterativa que parte de uma solução inicial qualquer, explora a vizinhança movendo-se sempre para o melhor vizinho segundo a função de avaliação, até atingir um ótimo local, do qual o método não é capaz de escapar.

Inicialmente, o Método da Descida explora toda a vizinhança de sua solução corrente. Se o melhor vizinho encontrado proporcionar uma melhora no valor da função de avaliação até então conhecida, ele é aceito como a nova solução a ser explorada. Caso contrário, um ótimo local foi alcançado e a solução corrente é retornada.

A Figura 3.2 mostra o pseudocódigo do Método de Descida, cujo objetivo é minimizar uma função de avaliação *f* .

1) Seja s_0 uma solução inicial conhecida.
2) $s \leftarrow s_0$, onde *s* é a solução corrente.
3) Faça *s'*← *melhorVizinho(s)*, onde *s'* é o melhor dentre todos os vizinhos de uma dada solução *s*.
4) enquanto (*f*(*s'*) < *f*(*s*)), faça
   a) $s \leftarrow s'$
   b) *s'*←*melhorVizinho(s)*
5) Retorne a melhor solução encontrada, *s*.

Figura 3.2 – Pseudocódigo do Método da Descida

3.2.3 Método Descida Randômica

O procedimento Descida Randômica (DR) é uma variante do procedimento Descida apresentado na seção 3.2.2, e tem por finalidade evitar a busca exaustiva em toda a vizinhança de uma dada solução.

A idéia básica dessa técnica é obter a cada iteração um vizinho qualquer da solução corrente, admitindo-o como nova solução corrente caso ele proporcione melhoras no valor da função de avaliação. Caso contrário, um novo vizinho é gerado. Esse procedimento se repete até que se exceda um número determinado de iterações sem melhoras na solução.

Portanto, trata-se de um procedimento computacionalmente mais barato que o método de Descida clássico, mas que também fica preso ao primeiro ótimo local encontrado. Devido à aleatoriedade na escolha dos vizinhos, pode ocorrer de a solução retornada pelo método Descida Randômica com Primeiro de Melhora não corresponder a um ótimo local.

Na Figura 3.3 é apresentado o pseudocódigo do método Descida Randômica aplicado à minimização de uma função *f*.

6) Seja s_0 uma solução inicial conhecida.
7) $s \leftarrow s_0$, onde *s* é a solução corrente.
8) enquanto (*critério de parada não satisfeito*), faça
   a) Faça *s'*← *primeiraMelhora(s)*, onde *s'* é o primeiro vizinho de melhora de uma dada solução *s*.
   b) $s \leftarrow s'$, se *s'* for melhor que *s* de acordo com *f*
   c) Atualize o critério de parada
9) Retorne a melhor solução encontrada, *s*.

Figura 3.3 – Pseudocódigo do Método Descida Randômica

3.3 Metaheurísticas

Metaheurísticas podem ser vistas como sendo heurísticas genéricas mais sofisticadas, que visam explorar inteligentemente as instâncias do problema e o seu espaço de soluções. As metaherísticas se caracterizam pela habilidade de escapar de ótimos locais e são destinadas a encontrar uma boa solução, eventualmente a ótima, para um problema de otimização.

As metaheurísticas se dividem em duas categorias, de acordo com o princípio usado para explorar o espaço de soluções: busca local e busca populacional. Nas metaheurísticas baseadas em busca local, a exploração do espaço de soluções é feita por meio de movimentos, os quais são aplicados a cada passo sobre a solução corrente, gerando outra solução promissora em sua vizinhança. Busca Tabu, *Simulated Annealing*, Método de Pesquisa em Vizinhança Variável, Método de Descida em Vizinhança Variável e *Iterated Local Search* são exemplos de métodos que se enquadram nesta categoria. Os métodos baseados em busca populacional, por sua vez, consistem em manter um conjunto de boas soluções e combiná-las de forma a tentar produzir soluções ainda melhores. Exemplos clássicos de procedimentos dessa categoria são os Algoritmos Genéticos e os Algoritmos Meméticos.

A seguir são apresentadas as metaheurísticas referenciadas neste trabalho.

3.3.1 *Iterated Local Search*

ILS (*Iterated Local Search* - Busca Local Iterada) é um método aplicado em conjunto com um procedimento de busca local, com a finalidade de melhorar as soluções obtidas por este através da realização de perturbações. Uma perturbação consiste em modificações realizadas sobre uma solução por meio da aplicação de um ou mais tipos de movimentos.

Para iniciar a sua exploração o ILS precisa primeiramente de uma boa solução para o problema, para tanto, é gerada uma solução inicial e esta é melhorada por meio de um procedimento de busca local.

Uma vez obtida uma solução é iniciada a fase de pesquisa sobre a sua vizinhança, que consiste em: se realizar uma perturbação sobre a solução corrente, seguida de um procedimento de busca que retornará o ótimo local da solução perturbada. Este ótimo é então avaliado e caso seja satisfeito o critério de aceitação (por exemplo, se houve melhora na função objetivo), ele será considerado a nova solução corrente e recomeça-se o processo de pesquisa pela vizinhança dessa nova solução. Caso contrário é aumentado o nível de perturbação, ou seja, torna-se a perturbação mais intensa e o processo de pesquisa é retomado. Este procedimento é repetido até que um critério de parada seja satisfeito, como por exemplo, quando o tempo de execução do algoritmo exceder um dado limite.

É importante salientar que o sucesso do ILS depende fortemente das perturbações realizadas pelo mesmo. Portanto, a perturbação aplicada a uma dada solução, deve ser dosada de maneira tal que as alterações decorrentes de sua aplicação sejam suficientes para se escapar de ótimos locais e explorar diferentes regiões sem, contudo causar um caos absoluto que implicaria na perda de características do ótimo local corrente.

Em princípio, qualquer método de busca local pode ser utilizado pelo ILS, no entanto, seu desempenho com respeito à qualidade da solução final e a velocidade de convergência dependem fortemente do método escolhido. Normalmente um método de descida é usado, mas também é possível aplicar algoritmos mais sofisticados, tais como Busca Tabu ou outras metaheurísticas.

Na Figura 3.6 é apresentado o pseudocódigo do algoritmo ILS básico em que se identificam quatro componentes:

a) Procedimento *GeraSolucaoInicial*, que gera uma solução inicial s_0 para o problema;

b) Procedimento *BuscaLocal*, que retorna uma solução melhorada *s* após geração da solução inicial e outra *s''* após perturbação por meio de um método de busca local;

c) Procedimento *Perturbacao*, que modifica a solução corrente *s* guiando a uma solução intermediária *s'*;

d) Procedimento *CriterioAceitacao*, decisão que se refere tanto à aceitação da solução perturbada como solução corrente, quanto quão intenso será a próxima perturbação a ser aplicada (isto é, determinação do nível de perturbação).

1) $s_0 \leftarrow$ *GeraSolucaoInicial()*;
2) $s \leftarrow$ *BuscaLocal*(s_0);
3) enquanto (os critérios de parada não estiverem satisfeitos) faça
   a) $s' \leftarrow$ *Perturbacao*(historico, *s*);
   b) $s'' \leftarrow$ *BuscaLocal*(s');
   c) $s \leftarrow$ *CriterioAceitacao*$(s, s'',$ historico);
4) Retorne a solução *s*

Figura 3.4 – Pseudocódigo do algoritmo *Iterated Local Search*

3.3.2 *Variable Neighborhood Descent*

O Método de Descida em Vizinhança Variável (*Variable Neighborhood Descent*, VND), proposto por Nenad Mladenovic e Pierre Hansen (Mladenovic e Hansen, 1997), é um método de refinamento que consiste em explorar o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança, aceitando somente soluções de melhora da solução corrente e retornando à primeira estrutura quando uma solução melhor é encontrada.

O pseudocódigo desse algoritmo, em que se considera o refinamento de uma solução *s* utilizando uma função de avaliação *f*, a ser minimizada, e um conjunto de *r* diferentes vizinhanças $N = N^{(1)}; N^{(2)}; ... ; N^{(r)}$, é apresentado pela Figura 3.5.

Dependendo do problema abordado, a busca pelo melhor vizinho (linha 5 da Figura 3.5) pode ser cara computacionalmente. Nessa situação é comum fazer a busca pela primeira solução de melhora. Outra alternativa é considerar a exploração apenas em certo percentual da vizinhança.

```
Algoritmo VND
1:  Entrada: f(.), N(.), r, s
2:  Seja r o número de estruturas diferentes de vizinhança
3:  tipo ← 0
4:  enquanto tipo < r faça
5:      Encontre o melhor vizinho s' ∈ N^(tipo) (s)
6:      se ( f(s') < f(s) )
7:          s ← s'
8:          tipo ← 0
9:      senão
10:         tipo ← tipo + 1
11:     fim se
12: fim enquanto
13: retorne s
```

Figura 3.5 – Algoritmo VND

Segundo os autores, o método VND baseia-se em três princípios básicos:

- Um ótimo local com relação a uma dada estrutura de vizinhança não corresponde necessariamente a um ótimo local com relação a uma outra estrutura de vizinhança;
- Um ótimo global corresponde a um ótimo local para todas as estruturas de vizinhança;
- Para muitos problemas, ótimos locais com relação a uma ou mais estruturas de vizinhança são relativamente próximas.

Segundo os autores, o último princípio, de natureza empírica, indica que um ótimo local frequentemente fornece algum tipo de informação sobre o ótimo global. Esse é o caso em que os ótimos, tanto local quanto global, compartilham muitas variáveis com o mesmo valor, o que sugere uma investigação sistemática da vizinhança de um ótimo local até a obtenção de uma nova solução de melhor valor.

3.3.3 *Busca Tabu*

O método de Busca Tabu, proposto independentemente por Glover (1986) e Hansen (1986), é um procedimento iterativo para a solução de problemas de otimização combinatória que aceita movimentos de piora para tentar escapar de ótimos locais.

Partindo de uma solução inicial s_0, a cada interação, o método Busca Tabu explora um subconjunto V de uma vizinhança $N(s)$ da solução corrente s. A busca nessa vizinhança é interrompida quando um movimento que melhore a solução corrente for encontrado. A idéia de reduzir o espaço de busca se deve ao fato da iteração do método de Busca Tabu ser custosa computacionalmente para o PPVT. Essa interrupção da busca, conhecida como primeira melhora, é uma idéia aplicada em métodos heurísticos e visa reduzir o esforço computacional para explorar a vizinhança.

A escolha do melhor vizinho a cada iteração é utilizada para tentar escapar de mínimos locais. Porém, este critério de escolha pode fazer com que o algoritmo cicle, ou seja, que retorne a uma solução já visitada anteriormente. Para evitar a ciclagem, existe um mecanismo chamado memória de curto prazo ou lista tabu. O objetivo dessa lista é tentar evitar

movimentos que levem à regiões já visitadas do espaço de soluções, o que usualmente é alcançado pela proibição dos últimos movimentos realizados. Esses movimentos são mantidos nesta lista por um certo número de iterações chamado *tabu tenure*. Durante estas iterações, os movimentos armazenados ficam proibidos de serem realizados. Mas essa estratégia pode ser muito restritiva e desconsiderar soluções de alta qualidade. Para que isto não ocorra, movimentos com status tabu podem ser aceitos se a nova solução produzida satisfizer ao critério de aspiração por objetivo, isto é, se a solução, ainda que tabu, for melhor que a melhor solução gerada até então.

De forma a reduzir ainda mais a probabilidade de ciclagem, o método de Busca Tabu implementado utiliza uma lista tabu de tamanho dinâmico. No caso, a cada iteração do método, o movimento tabu permanece na lista por um certo número de iterações, sorteado no intervalo [minTabuTenure, maxTabuTenure]. Na Figura 3.6 é apresentado o pseudocódigo do algoritmo de Busca Tabu, aplicado a um problema de minimização.

Algoritmo BT

```
1: Entrada: f(), N(), |V|, MinTabuTenure, MaxTabuTenure, s
2: s* ← s;                                   { Melhor solução até então }
3: ListaTabu ← ∅;                            { Lista Tabu }
4: enquanto ( critério de parada não satisfeito ) faça
5:     Seja s' ← s ⊕ m o melhor vizinho de V ⊂ N(s) tal que o movimento m não seja tabu
       (m ∉ ListaTabu) ou, se tabu, satisfaça f(s') < f(s*);
6:     Atualize ListaTabu;
7:     s ← s';
8:     se ( f(s) < f(s*) ) então
9:         s* ← s;
10:    fim-se
11: fim-enquanto;
12: s ← s*;
13: retorne s;
Fim BT;
```

Figura 3.6 - Algoritmo Busca Tabu

FONTE: Souza (2008)

3.4 Método de Intensificação

3.4.1 *Relaxação Adaptativa*

A Relaxação Adaptativa ou oscilação estratégica consiste em orientar movimentos em relação a um nível crítico, que pode ser identificado por um estágio de construção ou um intervalo escolhido de valores para uma função. Tal nível crítico (ou fronteira de oscilação) freqüentemente representa um ponto onde o método normalmente iria ser interrompido. Em vez de parar quando esta fronteira é alcançada, as regras para selecionar movimentos são modificadas para permitir que a região definida pelo nível crítico seja atravessada. Esta abordagem então continua até uma dada profundidade além da fronteira de oscilação e volta novamente. O limite da oscilação é novamente alcançado e atravessado, desta vez na direção oposta, e assim por diante.

O processo de repetidamente abordar e atravessar o nível crítico a partir de diferentes direções cria um comportamento oscilatório. O controle sobre esse comportamento é estabelecido por meio da geração de avaliações e regras modificadas para os movimentos, dependendo da região navegada e da direção da busca. A possibilidade de percorrer uma trajetória já visitada pode ser evitada por mecanismos tabu padrões, tais como aqueles estabelecidos pelas funções da memória de curto prazo.

Schaerf (1996) apresenta um mecanismo de relaxação adaptativa onde os pesos para cada fonte de inviabilidade da função de avaliação são ajustados dinamicamente, como proposto originalmente por Gendreau *et al*. (1994). Para cada inviabilidade *i* o peso β_i é multiplicado por um fator σ_i que varia de acordo com o seguinte esquema:

- No início da busca $\sigma_i \leftarrow 1$.
- A cada *k* movimentos:
  - se todas as *k* soluções visitadas são factíveis em relação à restrição *i* então $\sigma_i \leftarrow \sigma_i / \gamma$;
  - se todas as *k* soluções visitadas são infactíveis em relação à restrição *i* então $\sigma_i \leftarrow \sigma_i \times \gamma$;
  - se algumas soluções são factíveis e outras são infactíveis, considerando a restrição *i*, então σ_i permanece inalterado.

O parâmetro γ é randomicamente selecionado, a cada *k* movimentos, no intervalo [1,8; 2,2] conforme proposto por Schaerf (1996). Cada valor de σ_i é limitado por duas constantes σ_{min} e σ_{max}, de forma a evitar que a relaxação adaptativa incremente/decremente indefinidamente os pesos para restrições que são sempre insatisfeitas/satisfeitas.

4 O Problema de Programação de Veículos e Tripulações Abordado

O presente capítulo descreve, em detalhes, o problema abordado. Na seção 4.1, apresenta-se as restrições e objetivos da programação de veículos e na seção 4.2, da programação de tripulações.

O Problema de Programação Integrada de Veículos e Tripulações (PPVT) abordado neste trabalho pode ser definido da seguinte maneira: dada uma tabela de viagens a serem realizadas, encontre a programação de custo mínimo para os veículos e as tripulações, fazendo o melhor aproveitamento da frota de veículos e dos tripulantes de uma empresa de modo que todas as viagens a ela responsabilizadas sejam realizadas.

Para que a programação de atividades da frota de veículos e do conjunto de tripulantes seja factível é necessário atender a uma série de restrições, as quais podem variar conforme as políticas operacionais que regem o Sistema de Transporte Público no qual a empresa atua. Neste trabalho é tratado o problema particular envolvendo empresas da cidade de Belo Horizonte/Brasil.

A resolução do problema de programação de veículos e de tripulantes é feita a partir de uma tabela de horários, na qual se encontram todas as viagens a serem realizadas em um determinado dia por uma empresa. A Tabela 4.1 é um exemplo de tabela de horários.

Tabela 4.1 – Exemplo de uma tabela de viagens

idViagem	Horário de Inicio	Ponto Inicial	Horário de Término	Ponto Final	Linha
0	100	1	290	1	LN305
1	140	1	210	1	LN305
2	310	1	380	1	LN305
3	310	1	460	1	LN150
4	470	1	540	1	LN150
5	480	1	550	1	LN305
6	575	1	790	1	LN305
7	640	1	710	1	LN150
8	730	1	880	1	LN150
9	860	1	1090	1	LN150

Na tabela de horários, cada viagem é identificada univocamente por meio de um conjunto de características que consistem em informações relevantes no processo de resolução do problema. Organizados em colunas e aparecendo seqüencialmente na tabela, temos: o número da viagem, horário de início da viagem (em minutos), o ponto inicial que corresponde ao local de início da viagem, horário de término da viagem (em minutos), o ponto final que se refere ao local em que a viagem termina e a linha a qual a viagem pertence.

O tempo de embarque e o tempo de desembarque de passageiros também é um parâmetro importante em uma viagem. O primeiro dependente do horário de início da viagem e o segundo do horário de fim da mesma.

Ao longo do dia existem determinados períodos em que geralmente há um número maior de passageiros em trânsito pelo meio de transporte em questão. Esses períodos são os chamados horários de pico. As empresas estimam que na prática nesses horários o tempo necessário para o de embarque ou desembarque de passageiros é de dois minutos e fora desses

horários esse tempo é de um minuto. Existem dois horários de pico ao longo do dia: das 5:00 às 7:59 da manhã e das 4:00 às 18:59.

Além do conjunto de viagens, outra informação importante é a matriz de tempos de deslocamento entre os vários pontos de parada dos veículos e tripulantes, incluindo a garagem. De posse dessa informação é possível, por exemplo, determinar o tempo que um veículo percorre fora de operação, ou seja, sem passageiros para sair de um ponto e chegar a outro. A este tipo de deslocamento é dado o nome de viagem morta.

Quanto aos horários de início e término a tabela de viagens de exemplo (Tabela 4.1) é representada esquematicamente na Figura 4.1:

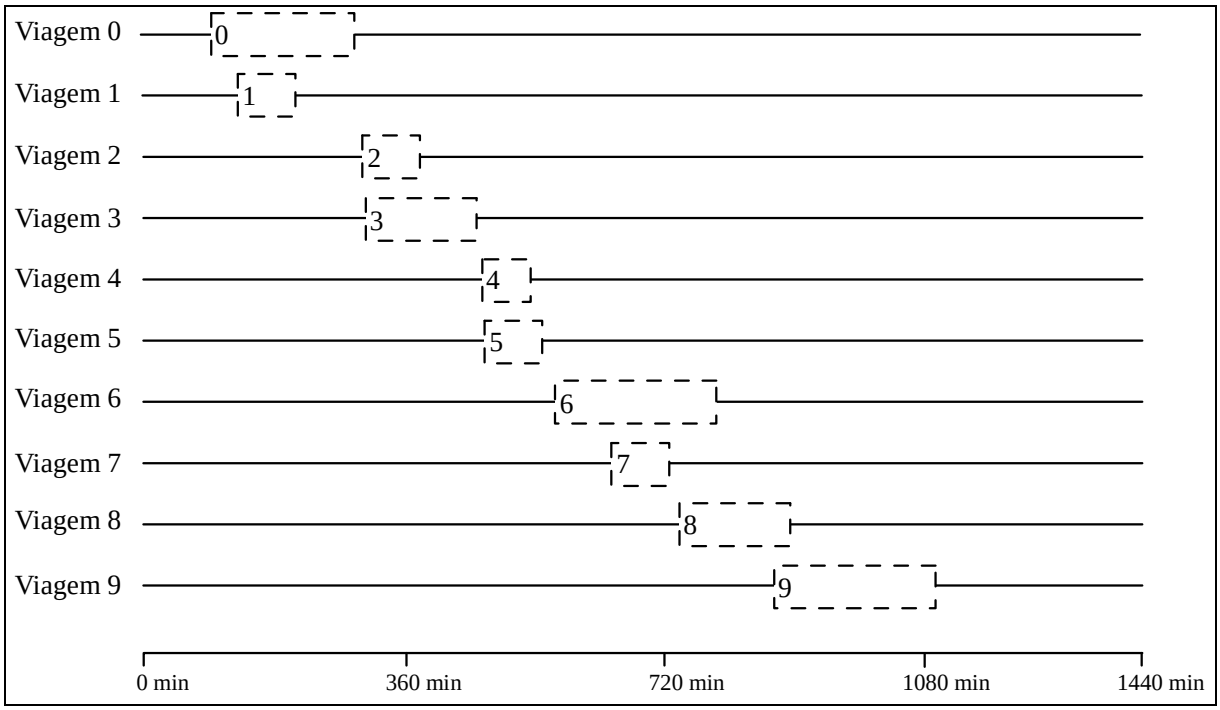

Figura 4.1 - Representação esquemática das viagens da Tabela 4.1

O PPT em um sistema de transporte público consiste na definição da jornada de trabalho de cada tripulação pertencente ao quadro de uma dada empresa. Uma tripulação é constituída por um motorista e um cobrador (quando houver), responsáveis pela operação de um veículo. As tarefas são formadas a partir de jornadas de trabalho de veículos, agrupando conjuntos de viagens. Este agrupamento é feito de maneira que cada tarefa fica compreendida entre duas *Oportunidades de Trocas (OT).* Uma OT é um intervalo de tempo suficiente, em um ponto apropriado, que possibilita a troca de tripulações de um veículo em operação. Viagens de uma mesma tarefa são executadas pelo mesmo veículo. Durante a realização de uma tarefa, não é possível que haja a troca da tripulação. Sendo assim, uma tarefa é a mínima porção de trabalho que pode ser adicionada ou retirada da jornada de trabalho de uma tripulação.

A programação das tripulações é feita a partir de um conjunto de tarefas que, unidas, contêm todas as viagens a serem realizadas em um determinado dia por uma empresa. A Figura 4.2 representa um conjunto de tarefas, gerado a partir de uma possível organização das viagens da Tabela 4.1. Na Figura 4.2 a Tarefa 2 contém a viagem 3 e a viagem 4.

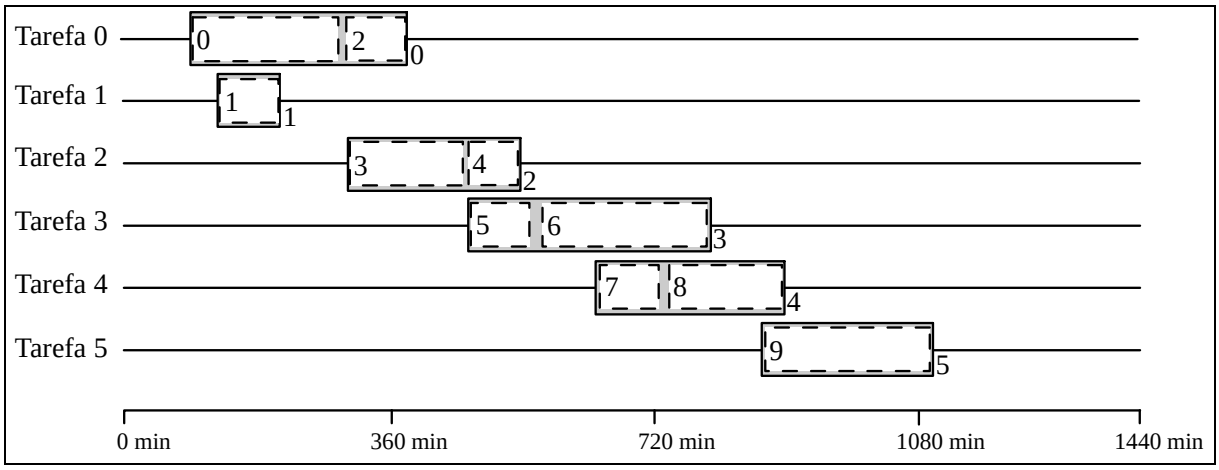

Figura 4.2 - Representação esquemática de uma lista de tarefas

4.1 Restrições e Objetivos da Programação de Veículos

As restrições que foram levadas em consideração para geração da programação diária dos veículos neste trabalho, foram as seguintes:

a) Um veículo deve permanecer no mínimo trinta minutos consecutivos na garagem, por dia. Este tempo pode ser cumprido quando o veículo pára na garagem entre duas viagens consecutivas pertencentes à sua jornada diária ou mesmo depois de completada toda a sua jornada de trabalho;

b) Um mesmo veículo não pode realizar duas viagens simultaneamente, o que caracterizaria uma sobreposição;

c) Antes de se iniciar e após finalizar uma viagem, um veículo deve obrigatoriamente cumprir um tempo mínimo no terminal, que corresponde ao tempo necessário para o embarque e desembarque dos passageiros;

d) Um veículo não deve permanecer mais de duas horas em um terminal esperando para executar a sua próxima viagem. Se o tempo de espera exceder esse limite, o veículo deverá se deslocar para a garagem ou outro local onde possa esperar o horário de início da viagem subseqüente. Diz-se, nesse último caso, que o veículo está fazendo uma dupla pegada;

e) Todas as viagens de responsabilidade de uma empresa, sem exceções, devem ser realizadas por algum veículo da frota.

Os objetivos do PPV considerados neste trabalho são os seguintes: (1) diminuir o tempo ocioso dos veículos, isto é, que os veículos não fiquem parados entre duas viagens consecutivas descontando-se os tempos de embarque e desembarque que são obrigatórios; (2) diminuir o número de deslocamentos fora de operação (deslocamentos que não equivalem a nenhuma viagem); (3) reduzir o tamanho da frota utilizada e (4) diminuir o número de trocas de linha entre as viagens dos veículos da frota.

A solução para o Problema de Programação de Veículos consiste, portanto, em atribuir um conjunto de viagens para um número de veículos menor ou igual à frota disponível pela empresa, considerando as restrições apresentadas acima, de tal forma que todas as viagens sejam realizadas com o menor custo operacional possível. Cada conjunto de viagens a serem executadas por um veículo é chamado de “bloco do veículo”.

4.2 Restrições e Objetivos da Programação de Tripulações

Para que a programação das jornadas de trabalho das tripulações seja factível é necessário atender a uma série de restrições. Sendo assim, o Problema de Programação de Tripulações (PPT) pode ser definido da seguinte forma: dado um conjunto de tarefas, encontrar um conjunto de jornadas diárias com custo mínimo tal que cada jornada seja uma seqüência de tarefas que pode ser executada por uma única tripulação respeitando as restrições operacionais e trabalhistas. As restrições consideradas são as seguintes:

a) Cada tarefa deve ser atribuída a uma única jornada de trabalho;

b) Um mesmo tripulante não pode realizar duas tarefas simultaneamente, o que caracterizaria uma sobreposição;

c) Um tripulante não deve permanecer mais de duas horas em um terminal esperando para executar a sua próxima viagem. Se o tempo de espera exceder esse limite, o tripulante deverá se deslocar para a garagem ou outro local onde possa esperar o horário de início da tarefa subseqüente. Diz-se, nesse último caso, que o tripulante está fazendo uma dupla pegada;

d) A duração da jornada diária de trabalho não pode superar 9 horas se a jornada for classificada como simples, ou 8:40 se for classificada como de dupla pegada;

e) As trocas de tripulações só podem ocorrer em determinados locais, nos pontos;

f) Nenhum dos tripulantes da frota deve efetuar trocas proibidas de veículos, linhas ou pontos;

g) Todas as viagens de responsabilidade de uma empresa, sem exceções, devem ser realizadas por algum tripulante.

Uma troca de veículos é permitida quando há tempo para se efetuar a troca, caso contrário ela é proibida. Uma troca de linha é permitida se as linhas pertencerem a um mesmo grupo, caso contrário ela é proibida. A definição dos grupos das linhas é feita pela empresa de transporte. Uma troca de ponto de ônibus é permitida (1) se os pontos envolvidos na troca pertencerem ao mesmo grupo de pontos; (2) caso contrário, se as tarefas pertencerem ao mesmo veículo; (3) caso contrário, apenas se houver dupla pegada entre as duas tarefas. Uma troca de ponto é proibida caso não satisfaça nenhum dos três quesitos anteriores.

Os objetivos do PPT considerados neste trabalho são os seguintes: gerar soluções finais factíveis, ou seja, sem sobreposição de tarefas e sem excesso de trabalho, além disso, deve haver tempo suficiente entre as jornadas de trabalho de um tripulante, segundo as leis trabalhistas; o número de trocas de veículos, o número de trocas de linhas e o número de trocas de pontos devem ser mínimos, assim como o tempo ocioso e o tempo de hora extra; não deve haver excesso de duplas pegadas; e o número de tripulantes também é um item a ser minimizado.

5 Metodologia

Neste capítulo apresenta-se a metodologia proposta para resolver, de forma integrada, o problema de programação de veículos e tripulações (PPVT). Na seção 5.1 mostra-se como uma solução para o problema é representada. Na seção 5.2 são apresentados os tipos de movimentos que constituem as estruturas de vizinhança desenvolvidas para explorar o espaço de busca. Nas seções 5.3 e 5.4 são mostradas duas formas utilizadas para avaliar uma solução, sendo a primeira com pesos empíricos e a segunda, baseada em custos. A descrição de como uma solução inicial é gerada é apresentada na seção 5.5. O algoritmo proposto para resolver o PPVT é apresentado na seção 5.6.

5.1 Representação do PPVT

Uma solução s para o PPVT consiste em um par (s_V, s_T), onde s_V é uma solução para o problema de programação de veículos e s_T uma solução para o problema de programação de tripulações.

A solução s_V consiste em um conjunto de veículos, a cada qual está associado uma lista de viagens a serem por ele executadas durante um dia de trabalho. As viagens executadas por cada veículo são mantidas ordenadas por seus horários de início. Essa abordagem é importante, uma vez que através dela será possível avaliar a jornada de trabalho de um veículo sob vários aspectos, tais como: tempo de operação em viagem, tempo de locomoção fora de operação, ou seja, deslocamentos que não correspondem a nenhuma viagem, tempo de espera em terminal e total de retornos à garagem.

A solução s_T, por sua vez, consiste em um conjunto de tripulações (motoristas e cobradores), a cada qual está associada uma lista de tarefas a serem executadas pelos tripulantes diariamente.

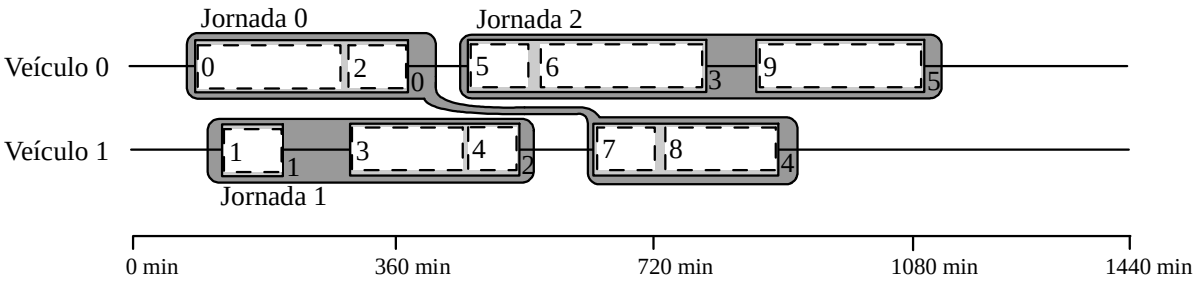

Figura 5.1 – Exemplo de uma solução s para o PPVT

A Figura 5.1 ilustra uma solução $s = (s_V, s_V)$ para o PPVT, na qual uma frota de dois veículos executa dez viagens a ela responsabilizada. Cada viagem é representada por um retângulo pontilhado de cor branca, cada tarefa é um retângulo de cor cinza claro, e cada jornada é a figura que une tarefas, cinza escuro. Nessa solução, o Veículo 1 executa, ao longo do dia, as viagens 1, 3, 4, 7 e 8. Onde, a viagem 1 constitui a tarefa 1, as viagens 3 e 4 constituem a tarefa 2 e as viagens 7 e 8 a tarefa 4. Note que o tripulante responsável pela jornada 0 executa a tarefa 0 no veículo 0 e a tarefa 4 no veículo 1.

5.2 Estruturas de vizinhança

Os métodos heurísticos usados neste trabalho são baseados na efetuação de modificações em uma dada solução com a finalidade de se melhorar sua qualidade. Essas modificações são chamadas *movimentos* e diferentes tipos de movimento podem ser aplicados em uma solução. Uma estrutura de vizinhança é definida por um tipo de movimento.

Um vizinho de uma solução *s* é definido pela aplicação de um movimento na solução *s*. Aplicando diferentes tipos de movimentos os métodos heurísticos podem explorar diferentes regiões do espaço de soluções.

Com vistas a explorar o espaço de soluções definimos 6 estruturas de vizinhança.

Duas delas N_v^R e N_v^T fazem modificações somente nos veículos, ignorando os reflexos destes movimentos na programação de tripulações, o que gera, possivelmente, soluções inviáveis.

Outras duas estruturas N_t^R e N_t^T modificam somente a programação das tripulações, preservando a compatibilidade entre veículos e tripulações.

O terceiro par de estruturas de vizinhança N_v^{RP} e N_v^{TP} é integrado e modifica ambas as partes, veículos e tripulações. Essas duas últimas estruturas consistem na realocação ou troca as viagens de veículos, seguida de uma reconstrução das jornadas de forma a tornar as programações de veículos e tripulações compatíveis entre si. Para cada veículo envolvido na modificação, são eliminadas da programação de tripulações todas as tarefas pertencentes a esses veículos. A seguir, a partir da nova configuração dos blocos dos veículos envolvidos, são geradas as novas tarefas. Essas tarefas, por sua vez, são adicionadas a qualquer tripulante, de forma gulosa de acordo com o procedimento descrito na seção anterior. Para essas duas estruturas, N_v^{RP} e N_v^{TP}, diz-se que a modificação realizada é "propagada" para as tripulações.

5.2.1 Vizinhança N_v^R

O movimento de realocação definido pela estrutura de vizinhança N_v^R consiste em transferir uma viagem pertencente à lista de viagens de um dado veículo para a lista de outro veículo. Portanto, este movimento pode ser visto como uma seqüência de decisões a serem tomadas e uma respectiva ação da seguinte forma:

1. Escolher um primeiro veículo, dentre todos os envolvidos no problema;
2. Escolher um segundo veículo que não seja o mesmo selecionado anteriormente;
3. Escolher uma viagem do primeiro veículo que será transferida para a lista de viagens do segundo;
4. Efetivar o movimento.

5.2.2 Vizinhança N_v^T

O movimento de troca definido pela estrutura de vizinhança N_v^T consiste em permutar uma tarefa da jornada de trabalho de um dado tripulante com uma tarefa pertencente a um outro tripulante. Igualmente aos processos anteriores, no movimento de troca, antes que se possa efetivamente realizar a permutação das tarefas algumas decisões devem ser tomadas:

1. Escolher um primeiro veículo, dentre todos os envolvidos no problema;
2. Escolher um segundo veículo que não seja o mesmo selecionado anteriormente;

3. Escolher uma viagem do primeiro veículo que será transferida para o bloco do segundo;
4. Escolher uma viagem do segundo veículo que será transferida para o bloco do primeiro;
5. Efetivar o movimento.

5.2.3 Vizinhança N_t^R

O movimento de realocação definido pela estrutura de vizinhança N_t^R consiste em transferir uma tarefa da jornada de trabalho de um dado tripulante para a jornada de trabalho de outro tripulante. Portanto, este movimento pode, também, ser visto como uma seqüência de decisões a serem tomas e uma respectiva ação da seguinte forma:

1. Escolher um primeiro tripulante, dentre todos os envolvidos no problema;
2. Escolher um segundo tripulante que não seja o mesmo selecionado anteriormente;
3. Escolher uma tarefa do primeiro tripulante que será transferida para a jornada de trabalho do segundo;
4. Efetivar o movimento.

5.2.4 Vizinhança N_t^T

O movimento de troca definido pela estrutura de vizinhança N_t^T consiste em permutar uma tarefa da jornada de trabalho de um dado tripulante com uma tarefa pertencente a um outro tripulante. Igualmente aos processos anteriores, no movimento de troca, antes que se possa efetivamente realizar a permutação das tarefas algumas decisões devem ser tomadas:

6. Escolher um primeiro tripulante, dentre todos os envolvidos no problema;
7. Escolher um segundo tripulante que não seja o mesmo selecionado anteriormente;
8. Escolher uma tarefa do primeiro tripulante que será transferida para a jornada de trabalho do segundo;
9. Escolher uma tarefa do segundo tripulante que será transferida para a jornada de trabalho do primeiro.
10. Efetivar o movimento.

5.2.5 Vizinhança N_v^{RP}

Na estrutura N_v^{RP}, um vizinho *s*’ de uma solução $s = (s_V, s_T)$ do PPVT é gerado fazendo-se a seguinte seqüência de operações: (i) realocação de uma viagem de um veículo para outro veículo na solução s_V; (ii) eliminação, na solução s_T, das tarefas que pertencem aos veículos envolvidos na troca; (iii) agrupamento, em tarefas, das viagens dos dois veículos envolvidos na realocação; (iv) geração de uma nova solução s_T, consistindo na distribuição das tarefas aos tripulantes por um procedimento guloso, conforme descrito na seção seguinte.

As estruturas de vizinhança N_v^{RP} e N_v^{TP} fazem uso de um método que, de forma gulosa, aloca as tarefas geradas por uma modificação, realocação ou troca, na programação dos veículos. Nessas estruturas, a primeira modificação se dá primeiramente na programação de veículos, sendo essa modificação propagada para a programação de tripulações. Essa propagação é feita substituindo as tarefas, anteriormente geradas com base na lista de viagens

dos veículos envolvidos, pelas novas tarefas, geradas a partir da lista de viagens dos mesmos veículos depois da modificação.

5.2.6 Vizinhança N_t^{TP}

Na estrutura N_t^{TP}, um vizinho *s*' de uma solução $s = (s_V, s_T)$ do PPVT é gerado fazendo-se a seguinte seqüência de operações: (i) troca de viagens entre dois veículos da solução s_V; (ii) eliminação, na solução s_T, das tarefas que pertencem aos veículos envolvidos na troca; (iii) agrupamento, em tarefas, das viagens pertencentes aos dois veículos envolvidos na troca; (iv) geração de uma nova solução s_T, consistindo na distribuição das tarefas, geradas pelo passo anterior, aos tripulantes por um procedimento guloso, conforme descrito na seção 5.2.5.

5.3 Avaliação de uma solução - 1ª Abordagem

Nesta abordagem, baseada em pesos empíricos, uma solução $s = (s_V, s_T)$ do PPVT é avaliada por uma função *f* baseada em penalidades, a qual deve ser minimizada, na forma:

$$f(s) = \alpha \times f_V(s) + \beta \times f_T(s) \tag{1}$$

em que $f_V(s)$ representa a componente que avalia *s* com relação à programação dos veículos, $f_T(s)$ representa a componente que avalia *s* com relação à programação das tripulações e α e β representam pesos atribuídos à importância relativa de cada uma dessas soluções.

5.3.1 *Avaliação da programação de veículos*

A função de avaliação f_V considerada para a componente s_V relativa a programação dos veículos é dada pela expressão (2), baseada em Simões *et al*. (2006).

A avaliação de uma solução para os veículos considera: tempo de sobreposição de viagens, tempo de viagem morta, tempo de espera em terminal, número de trocas de linhas, excesso de duplas pegadas e número de veículos.

$$f_V(s) = \sum_{k \in F} f_V^k(s) + numTrLinha \times \omega_{TL} + \text{excesso}DuplaPegV \times \omega_{\text{EDP}} + \\ + numVeículos \times \omega_{NV} \tag{2}$$

onde:

a) $f_V^k(s)$ é a função de custo do *k*-ésimo veículo da frota *F* de uma solução *s*.

b) *numTrLinha* é a quantidade de trocas de linha realizadas pelos veículos da frota e ω_{TL} é a penalização associada.

c) *excessoDuplaPegV* é a quantidade de duplas pegadas da programação de veículos que supera um certo número admissível previamente definido pela empresa e ω_{EDP} é a penalização correspondente.

d) *numVeículos* é a quantidade de veículos escalados para executar o conjunto de viagens e ω_{NV} é a penalização associada.

O custo $f_V^k(s)$ atribuído ao veículo *k* de uma solução *s* é calculado pela expressão (3). Sendo *T* o conjunto das suas *n* viagens diárias e *i* e *j* viagens consecutivas de sua jornada de trabalho.

$$f_V^k(s) = CD_{g1} + CD_{ng} + \sum_{(i,j)\in T} C_{ij} \tag{3}$$

Na equação (3), CD_{g1} e CD_{ng} são, respectivamente, os custos dos deslocamentos (isto é, das viagens mortas) do veículo *v* da garagem para o ponto de início de sua primeira viagem e do ponto final da sua última viagem para a garagem. C_{ij} é o custo associado à realização das viagens consecutivas *i* e *j* e é calculado conforme a expressão (4):

$$C_{ij} = \begin{cases} CD_{ij} + CS_{ij} & \text{se} \quad t_{ij} < 0 \\ CT_{ij} + CD_{ij} & \text{se} \quad 0 \le t_{ij} \le TempoMaxTerm \\ CD_{ig} + CD_{gj} & \text{se} \quad t_{ij} > TempoMaxTerm \end{cases} \tag{4}$$

onde CT_{ij}, CD_{ij} e CS_{ij} correspondem, respectivamente, aos custos de permanência no terminal, deslocamento realizado fora de operação (viagem morta) e sobreposição entre as viagens *i* e *j*. Na expressão (4), *TempoMaxTerm* é o tempo máximo que um veículo pode permanecer no terminal, estipulado segundo políticas operacionais que regem o Sistema de Transporte Público, e t_{ij} corresponde ao período de tempo compreendido entre o fim da viagem *i* e o início da viagem *j* após descontado o tempo de viagem morta, de embarque e desembarque entre elas.

Note que t_{ij} corresponde ao tempo real de ociosidade do veículo *v* entre as viagens i e j. Portanto, o valor t_{ij} poderá ser:

a) Negativo, indicando que há sobreposição de horário entre as viagens *i* e *j*. Assim, além da penalização associada à viagem morta realizada pelo veículo *v* entre as viagens *i* e *j*, há um custo atribuído ao tempo de sobreposição entre elas;

b) Não-negativo e menor ou igual ao tempo máximo de terminal permitido para um veículo (*TempoMaxTerm*), indicando que o veículo *v*, após a realização da viagem *i*, ficará esperando no terminal pelo horário de início de sua próxima viagem (viagem *j*). Portanto, o custo de se realizar as viagens *i* e *j* consecutivamente é obtido penalizando-se o tempo de deslocamento do veículo fora de operação e o tempo que o mesmo permanece no terminal entre as viagens;

c) Maior que o tempo máximo de terminal permitido para um veículo (*TempoMaxTerm*), indicando que o veículo *v* terá que retornar à garagem entre as viagens i e j, ou seja, realizará uma dupla pegada; não havendo, dessa forma, a necessidade de espera no terminal. O custo de se realizar as viagens, nesse caso, é calculado penalizando-se os tempos de deslocamento do veículo fora de operação.

Na prática, é desejável não apenas obter uma solução factível para o PPV com tempos de terminal e de deslocamento baixos na soma global da programação, mas também que exista uma homogeneidade na distribuição desses tempos, isto é, deve ser evitado que haja paradas no terminal excessivamente demoradas ou que os tempos de deslocamentos de um veículo fora de operação sejam muito elevados. Para cumprir esse objetivo, atribui-se um custo individual para cada espera no terminal ou para cada viagem morta de um veículo na expressão (4), de acordo com uma função exponencial. Os custos CD_{ij} e CT_{ij} são obtidos conforme as equações (5) e (6), apresentadas a seguir:

$$CD_{ij} = \begin{cases} (\alpha_1 \times d_{ij})^{\beta_1} & \text{se } d_{ij} \leq TempDesloc\_Lim \\ d_{ij} \times \omega_{\text{TD}} & \text{se } d_{ij} > TempDesloc\_Lim \end{cases} \quad (5)$$

$$CT_{ij} = \begin{cases} (\alpha_2 \times t_{ij})^{\beta_2} & \text{se } t_{ij} \leq TempTerm\_Lim \\ t_{ij} \times \omega_{\text{TT}} & \text{se } t_{ij} > TempTerm\_Lim \end{cases} \quad (6)$$

em que:

a) t_{ij} é o tempo real de ociosidade do veículo *v* entre as viagens i e j;

b) d_{ij} é a duração da viagem morta do veículo *v* entre as viagens i e j;

c) *TempDesloc_Lim* é o tempo limite para a duração de uma viagem morta;

d) ω_{TD} é a penalização, por unidade de tempo (em minutos), atribuída a uma viagem morta caso a sua duração supere o limite dado por *TempDesloc_Lim*;

e) α_1 e $\beta_1 \in (0,1)$ são parâmetros utilizados para o cálculo da penalização atribuída a uma viagem morta caso a sua duração seja menor ou igual ao limite dado por *TempDesloc_Lim*;

f) *TempTerm_Lim* é o tempo limite para a duração de uma parada no terminal;

g) ω_{TT} é a penalização, por unidade de tempo (em minutos), atribuída a uma espera no terminal que exceda o limite dado por *TempTerm_Lim*;

h) α_2 e $\beta_2 \in (0,1)$ são parâmetros utilizados para o cálculo da penalização atribuída a uma espera no terminal caso a sua duração seja menor ou igual ao limite dado por *TempTerm_Lim*.

Observe que os custos CT_{ij} e CD_{ij} são dependentes da duração da espera no terminal ou da viagem morta efetuada pelo veículo *v* entre as viagens *i* e *j*. A um tempo de terminal ou de deslocamento aceitável é associado um custo mais baixo, calculado com base em uma função exponencial. Caso esse tempo seja inaceitável, o custo é mais elevado e é calculado com base em uma função linear para diferenciar o nível de inviabilidade.

O motivo de se utilizar uma função exponencial está no fato de que ela estabelece uma variação suave do custo da espera no terminal (ou viagem morta) quando a duração é pequena e mais acentuada quando essa duração é maior. Assim, a tempos de terminal (ou viagem

morta) toleráveis estão associados custos mais próximos e para tempos quase intoleráveis, os custos são elevados.

Uma vez que a existência de sobreposição em uma solução para o PPV é uma inviabilidade, não se aplica para o cálculo de seu custo a mesma metodologia que a adotada anteriormente para penalizar o tempo de terminal e a viagem morta. Assim, o custo CS_{ij} é obtido de forma simples penalizando cada minuto de sobreposição pelo valor ω_{TS}, conforme equação (7):

$$CS_{ij} = - \left(t_{ij}\right) \times \omega_{\mathrm{TS}} \tag{7}$$

Note que, uma vez que o valor t_{ij} é sempre negativo quando há sobreposição entre as viagens *i* e *j*, é necessário multiplicá-lo por (-1) para que um custo positivo seja associado a esta sobreposição.

5.3.2 *Avaliação da programação de tripulações*

A função de avaliação considerada para a componente s_T relativa a programação das tripulações é baseada em penalizações de requisitos essenciais e não essenciais. Como as restrições devem ser satisfeitas sob pena de a solução do PPT não ter aplicabilidade, os pesos dados aos requisitos essenciais são bem mais elevados que aqueles atribuídos aos requisitos não-essenciais.

A avaliação de uma solução para os tripulantes considera: tempo de sobreposição de tarefas, tempo de excesso de trabalho, tempo insuficiente entre jornadas, número de trocas de veículos, número de trocas de linhas, número de trocas de pontos, tempo ocioso, tempo de hora extra, excesso de duplas pegadas e número de tripulantes.

Uma solução s_T do PPT é avaliada com base na seguinte função $f_T(s)$, baseada em Marinho *et al*. (2004):

$$\begin{aligned} f_T(s) = \sum_{i \in Trip} (& penalidadeTempoOcioso_i + penalidadeHoraExtra_i + \\ & + \mu_3 \times numTrVPerm_i + \mu_4 \times numTrLPerm_i + \mu_5 \times numTrPPerm_i + \\ & + \lambda_1 \times tempoSobrep_i + \lambda_2 \times tempoExcTrab_i + \lambda_3 \times tempoInsufEJorn_i + \\ & + \lambda_4 \times numTrocaV\,\mathrm{Proib} + \lambda_5 \times numTrL\,\mathrm{Proib}_i + \lambda_6 \times numTrP\,\mathrm{Proib}_i) + \\ & + \lambda_7 \times excessoDuplaPeg + \delta \times numTripulantes \end{aligned} \tag{8}$$

onde:

(i) *Trip* é o conjunto de tripulantes da componente s_T.

(ii) μ_i e λ_i são as penalidades aplicadas aos requisitos não essenciais e essências respectivamente.

(iii) $penalidadeTempoOcioso_i$ é a penalidade referente à quantidade total de ociosidade, em minutos, na jornada *i* e μ_1 é a penalização desse quesito. O valor μ_1 deve ser escolhido de tal forma que ele seja maior que o valor $penalidadeTempoOcioso_i$ caso o $tempoOcioso_i$ seja igual a 120 minutos (duas horas). O valor dessa penalidade é dado pela expressão 9, onde *tempoOcioso* representa a quantidade de tempo, em minutos, em que uma jornada *i* é ociosa.

$$penalidadeTempoOcioso_i = \begin{cases} \mu_1, se \quad tempoOcioso_i \succ 120 \\ \lfloor 0.347 \times tempoOcioso_i^{\,0.8} \rfloor, cc \end{cases} \quad (9)$$

(iv) $penalidadeHoraExtra_i$ é a penalidade referente à quantidade total de horas extras, em minutos, na jornada i e μ_2 é a penalização desse quesito. O valor μ_2 deve ser escolhido de tal forma que ele seja maior que o valor $penalidadeHoraExtra_i$ caso o $tempoExtra_i$ seja igual a 120 minutos (duas horas). O valor dessa penalidade é dado pela expressão 10.

$$penalidadeHoraExtra_i = \begin{cases} \mu_2, se \quad tempoExtra_i \succ 120 \\ \lfloor 1.357 \times tempoExtra_i^{\,0.6} \rfloor, cc \end{cases} \quad (10)$$

(v) $numTrVeiculosPerm_i$ é o número de vezes em que ocorreu troca de veículos permitida durante a jornada i e μ_3 a penalização para esse quesito. Uma troca de veículos é permitida quando há tempo para se efetuar a troca.

(vi) $numTrLPerm_i$ é o número de vezes em que ocorreu troca de linhas permitida durante a jornada i e μ_4 a penalização correspondente. Uma troca de linha é permitida se as linhas pertencerem a um mesmo grupo. A definição dos grupos das linhas é feita pela empresa de transporte.

(vii) $numTrPPerm_i$ é o número de vezes em que ocorreu troca de pontos permitida durante a jornada i e μ_5 a penalização desse quesito. Uma troca de ponto de ônibus é permitida (1) se os pontos envolvidos na troca pertencerem ao mesmo grupo de pontos; (2) caso contrário, se as tarefas pertencerem ao mesmo veículo; (3) caso contrário, apenas se houver dupla pegada entre as duas tarefas. Uma troca de ponto é proibida caso não satisfaça nenhum dos três quesitos anteriores.

(viii) $tempoSobrep_i$ é a quantidade de tempo, em minutos, em que duas tarefas são realizadas simultaneamente na jornada i e λ_1 a penalização para esse quesito.

(ix) $tempoExcTrab_i$ é a quantidade de tempo excedente, em minutos, na jornada i e λ_2 a penalização correspondente. O tempo excedente em uma jornada é dado por seu tempo de trabalho menos o tempo de máxima duração de uma jornada, que é dado pelo tempo limite de trabalho segundo a classificação da jornada, simples ou pegada dupla, mais o tempo máximo de hora extra, que, neste caso é de duas horas (120 minutos). A expressão 11 apresenta o cálculo desse valor.

$$tempoExcTrab_i = \begin{cases} \max(0, tempoTrab_i - (tempoLimJornSimples + 120)), \\ \quad se \quad i \quad for \quad Jornada \quad Simples \\ \max(0, tempoTrab_i - (tempoLimJornDupla + 120)), \\ \quad se \quad i \quad for \quad Jornada \quad Dupla \end{cases} \quad (11)$$

(x) $tempoTrab_i$ é a quantidade de tempo de trabalho, em minutos, da jornada i. Esse valor é dado pelo horário final da última tarefa, da jornada em questão, menos o horário de início da primeira, menos a duração do intervalo de dupla pegada se a jornada for classificada como de pegada dupla.

(xi) $tempoInsufEJorn_i$ é a quantidade de tempo, em minutos, que falta para que a diferença entre o horário final da jornada i em um dia e o horário inicial no dia subseqüente atinja o tempo mínimo entre jornadas de 11 (onze) horas. λ_3 é a penalização correspondente.

(xii) *numTrVProib*$_i$ é o número de vezes em que ocorreu troca de veículos proibida durante a jornada *i* e λ_4 a penalização para esse quesito. Uma troca de veículos é proibida quando não há tempo para se efetuar a troca.

(xiii) *numTrLProib*$_i$ é o número de vezes em que ocorreu troca de linhas proibida durante a jornada *i* e λ_5 a penalização desse quesito. Uma troca de linha é proibida se as linhas envolvidas pertencerem a grupos diferentes.

(xiv) *numTrPProib*$_i$ é o número de vezes em que ocorreu troca de pontos proibida durante a jornada *i* e λ_6 a penalização desse quesito.

(xv) *excessoDuplaPeg* é a quantidade de jornadas classificadas como de dupla pegada que supera um certo número previamente definido pela empresa. λ_7 é a penalização correspondente.

(xvi) *numTripulantes* é o número de tripulantes escalados para executar o conjunto de tarefas. δ é a penalização correspondente.

5.4 Avaliação de uma solução - 2ª Abordagem

Na programação dos veículos, e isso não é diferente na programação das tripulações, existem requisitos operacionais que precisão ser satisfeitos e requisitos de qualidade os quais devem ser minimizados. Esses requisitos podem ser diferentes de acordo com o caso estudado. As seções 5.4.1 e 5.4.2 apresentam quais os requisitos considerados neste trabalho e como esses requisitos são computados. Nós acreditamos que os requisitos considerados nessa abordagem podem ser aplicados em todos os casos práticos.

O custo de uma solução para o PPVT é calculado de acordo com a fórmula (1), a qual deve ser minimizada. Nesta formula, $f_v(s)$ representa a componente *f* que avalia a solução *s* com respeito à programação dos veículos e $f_t(s)$ avalia a solução *s* com respeito a programação das tripulações. Observa-se que tal função é, ao contrário da anterior, baseada em custos, tal como em Reis (2007).

$$f(s) = f_v(s) + f_t(s) \quad (1)$$

5.4.1 Avaliação dos veículos

Quanto menor é o numero de veículos necessários para que a empresa faça todas as viagens de sua responsabilidade, melhor é a solução. Outro ponto é, quanto menor é o tempo em que um veículo está ocioso (tempo ocioso) e quanto menor é o tempo em que um veículo efetua viagens as quais não pertencem ao conjunto de viagens de sua responsabilidade (tempo de viagem morta), melhor é a solução. Assim, nesta abordagem nós tentamos minimizar os seguintes itens referentes à programação de veículos:

- Número de veículos
- Tempo ocioso
- Tempo de viagem morta

Existe uma restrição obvia sobre a programação de veículos: é impossível que um veículo efetue mais que uma viagem ao mesmo tempo. Como a representação da solução permite que isso ocorra, é necessário especificar esta restrição. Nesta abordagem nós consideramos também uma restrição específica para as empresas de transporte público consideradas, uma solução não é válida se mais que 60% do número de veículos voltam à garagem durante, digo nem no início nem no final, sua jornada de trabalho (pegada dupla). Assim, uma programação de veículos é factível se:

- Não há conflitos de viagens na programação dos veículos
- O número de pegadas duplas é menor que 60% do número de veículos

O custo da programação de veículos é dado pela seguinte fórmula:

$$f_v(s) = f(s_v) = \sum_{i \in Q_v} \alpha_i q_i^v(s_v) + \sum_{i \in O_v} \gamma_i o_i^v(s_v) \quad (2)$$

Nesta formula: Q_v e O_v são requisitos de qualidade e operacionais da programação de veículos, respectivamente; α e γ são penalidades aplicadas aos requisitos de qualidade e operacionais da solução s_v, respectivamente; e q_i^v e o_i^v são valores do *i*-ésimo requisito de qualidade e operacional, respectivamente.

5.4.2 Avaliação das tripulações

As minimal is the number of crews that are necessary for the enterprises to make the set of trips, better is the solution. Another point is, as minimal is the time that a crew is idle and as minimal is the time that a crew is in overtime payment, better is the solution too. The overtime of a duty is given by the time that supers seven hours of work, i.e., the duration of a crew duty that has one hour of overtime payment is eight hours. Thus, in this work, we try to minimize the follow items about the crew schedule:

Quanto menor é o numero de tripulações necessárias para que a empresa faça todas as viagens de sua responsabilidade, melhor é a solução. Outro ponto é, quanto menor o tempo em que uma tripulação está em hora extra (tempo de hora extra), melhor é a solução também. O tempo de hora extra de uma tripulação é dado pelo tempo que supera 7 (sete) horas de trabalho, isto é, a duração da jornada de uma tripulação que possui uma hora extra é oito horas. Assim, nesta abordagem nós tentamos minimizar os seguintes itens referentes à programação das tripulações:

- Número de tripulações
- Tempo de hora extra
- Tempo ocioso

Assim como na programação dos veículos, na programação das tripulações, há uma restrição óbvia: é impossível que uma tripulação efetue mais que uma tarefa ao mesmo tempo. Além disso, por aspectos legais, no Brasil, uma empresa que possui alguma tripulação trabalhando mais que nove por dia ou não possui no mínimo onze horas entre o final de uma jornada de trabalho e o início da próxima jornada é ilegal e está sujeito a multa. Nesta abordagem também foi considerada uma restrição específica para as empresas de transporte público em questão, uma solução não pode ser aplicada se mais que 25% das tripulações voltam para a garagem durante suas jornadas. Assim, uma programação de tripulações é factível se:

- Não há conflitos entre as tarefas da programação das tripulações
- Não há tripulações trabalhando mais que 9 horas por dia
- Nenhuma tripulação possui menos que 11 horas de descanso entre o fim de uma jornada de trabalho e o início da próxima
- O número de pegadas duplas for menor que 25% do número de tripulações

O custo da programação das tripulações é dado pela fórmula:

$$f_t(s) = f(s_t) = \sum_{i \in Q_t} \beta_i q_i^t(s_t) + \sum_{i \in O_t} \theta_i o_i^t(s_t) \quad (3)$$

Nesta formula: Q_t e O_t são requisitos de qualidade e operacionais da programação de tripulações, respectivamente; β e θ são penalidades aplicadas aos requisitos de qualidade e operacionais da solução s_t, respectivamente; e q_i^t e o_i^t são valores do *i*-ésimo requisito de qualidade e operacional, respectivamente.

5.4.3 Definindo os pesos da avaliação

O objetivo da função de avaliação é avaliar a qualidade de uma solução. Como visto em seções anteriores, esta função considera requisitos que devem ser minimizados. É fato que alguns desses requisitos são relativamente mais importantes que outros. Para contemplar fielmente a importância relativa desses requisitos, os pesos apresentados nessa abordagem são baseados em custos reais descritos em Reis (2007). A Tabela 5.1 apresenta os dados utilizados no cálculo dos pesos da função de avaliação.

Tabela 5.1 – Dados usados no calculo dos pesos – 2ª Abordagem

Descrição	Valor
Pagamento do motorista (mês)	R$ 1.000,99
Pagamento do auxiliar	R$ 600,59
Pagamento da tripulação	R$ 1.601,58
Pagamento + taxas (38%)	R$ 2.210,18
Dias trabalhos por mês	24
Tempo de trabalho durante um dia	430
Pagamento do minuto de trabalho	R$ 0,21
Pagamento de um minuto de hora extra	R$ 0,32
Pagamento de um dia de trabalho	R$ 92,09
Custo fixo do veículo	R$ 13.415,58
Dias trabalhados dos veículos	30
Custo diário de um veículo	R$ 447,19
Velocidade média (km/h)	20
Custo por quilometro	R$ 2,64
Custo de um minuto de viagem morta	R$ 0,88
Custo variável por quilometro	R$ 0,78
Custo pelo minuto ocioso do veículo	R$ 0,26

A Tabela 5.2 apresenta os pesos utilizados, nessa abordagem, para a avaliação de uma solução do PPVT. Note que, os pesos de qualidade são baseados nos custos reais apresentados na Tabela 5.1, já os custos operacionais são empíricos e seus valores são maiores que os de qualidade. A última coluna desta tabela nos fornece um exemplo das características de uma solução exemplo. O custo dessa solução exemplo, de acordo com a esta abordagem, é 4450,84.

Tabela 5.2 – Pesos da função de avaliação – 2ª Abordagem

Tipo	Referente a	Requisito	Peso	Exemplo de s
Qualidade	Veículos	Número de veículos	447,19	5
		Tempo ocioso	0,26	1700
		Tempo de viagem morta	0,88	161
	Tripulações	Número de tripulações	92,09	11
		Tempo ocioso	0,21	1702
		Tempo de hora extra	0,32	815
Operacional	Veículos	Tempo de conflito	4000	0
		Excesso de pegadas dúplas	1000	0
	Tripulações	Tempo de conflito	4000	0
		Excesso de pegadas dúplas	1000	0
		Tempo insuficiente entre jornadas	4000	0
		Excesso de tempo de trabalho	4000	0

5.5 Geração de uma solução inicial para o PPV

A geração de uma solução inicial para o PPVT é feita de forma seqüencial. Constrói-se uma solução para os veículos e, posteriormente, constrói-se uma solução para as tripulações.

5.5.1 Método Guloso aplicado à geração de uma solução inicial para o PPV

Uma solução inicial para o PPV é gerada por um procedimento construtivo guloso. Esse procedimento consiste na alocação a cada passo de uma nova viagem de responsabilidade da empresa na lista de viagens de algum veículo de sua frota, de tal forma que esta combinação seja a que implique no menor custo de operação possível determinado pela aplicação da função de avaliação descrita na seção 5.3.

O pseudocódigo do método guloso considerado é descrito a seguir na Figura 5.4.

Algoritmo SIG_V

Entrada: $f(.)$, X, F
Seja X o conjunto de viagens a serem realizadas
Seja F uma frota de veículos, ainda sem viagens a executar
Seja s_V uma solução para os veículos
$s_V \leftarrow F$
Para cada viagem $x \in X$ faça
    $melhorDelta \leftarrow \infty$
    Para cada veículo $k \in s_V$ faça
        $custoAnterior \leftarrow$ custo do veículo k antes de receber a viagem x
        Aloque a viagem x no veículo k
        $custoPosterior \leftarrow$ custo do veículo k após inserir a viagem x
        Remova a viagem x do veículo k
        $\Delta = custoPosterior - custoAnterior$
        Se ($\Delta < melhorDelta$) então
            $melhorVeículo \leftarrow k$
            $melhorDelta \leftarrow \Delta$
        fim se
    fim para
    Aloque a viagem x ao veículo indicado por $melhorVeículo$
    $fo \leftarrow fo + melhorDelta$, ou seja, atualize o valor da função de avaliação
fim para
Retorne s_V

Figura 5.2 – Método Gerador de Solução Inicial para os Veículos

Para exemplificar como construir uma solução para o PPV pelo procedimento proposto, suponha que a empresa que deve executar as viagens da Tabela 4.1 possua dois veículos disponíveis. A seguinte seqüência de passos mostra o funcionamento do procedimento.

Passo 0: Constrói-se uma lista com o número de veículos da empresa. A seguir, aloca-se a primeira viagem da lista de viagens, ordenada por horário de início, ao veículo que gera o menor custo. Nesse caso, a Viagem 0 é alocada ao Veículo 0.

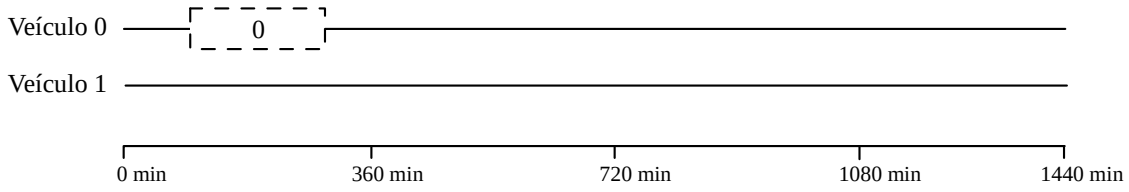

Figura 5.3 –Exemplo da geração de uma solução inicial para os veículos: Passo 0

Passo 1: Aloca-se a segunda viagem ao veículo que gera o menor custo de alocação. Nesse caso, a Viagem 1 é alocada ao Veiculo 1, pois a alocação dessa viagem ao Veículo 0 geraria uma sobreposição da Viagem 0 com a Viagem 1, gerando um custo de alocação muito elevado de acordo com a função de avaliação f_T descrita na seção 5.3.

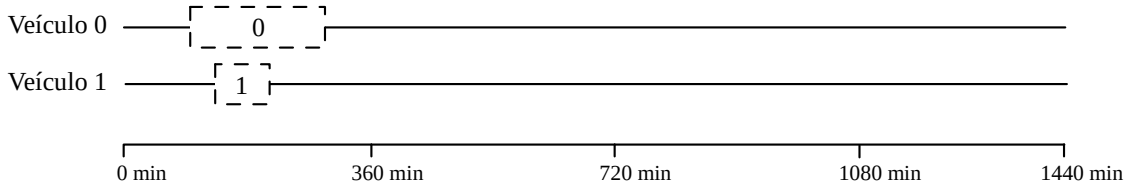

Figura 5.4 – Exemplo da geração de uma solução inicial para os veículos: Passo 1

Passo 2: Aloca-se a terceira viagem da lista ao veículo que gera o menor custo de alocação. No caso, a Viagem 2 é alocada ao Veículo 0, pois se ela fosse alocada ao Veículo 1 o tempo ocioso seria maior.

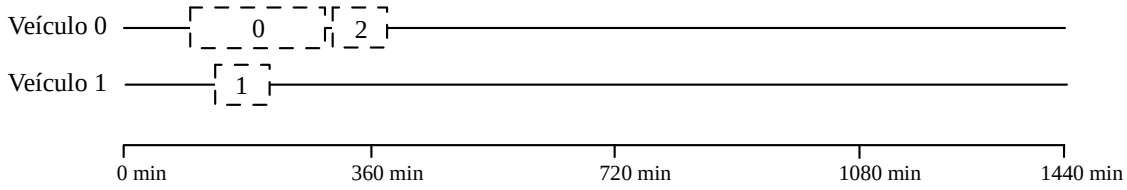

Figura 5.5 – Exemplo da geração de uma solução inicial para os veículos: Passo 2

Passo 3: Aloca-se a próxima viagem da lista no veículo que gera o menor custo de alocação. Nesse caso, a Viagem 3 é alocada ao Veículo 1. Essa viagem não poderia ser alocada ao Veículo 0 pois, neste caso, haveria sobreposição com a Viagem 2.

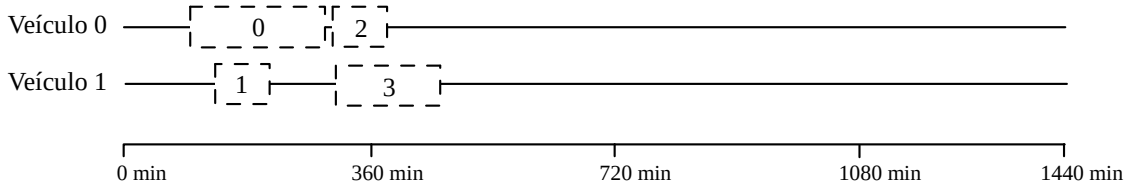

Figura 5.6 – Exemplo da geração de uma solução inicial para os veículos: Passo 3

Passo 4: Aloca-se a próxima viagem da lista no veículo que gera o menor custo de alocação. Neste caso, a Viagem 4 é alocada ao Veículo 1, uma vez que tal alocação resulta em menor tempo de ociosidade.

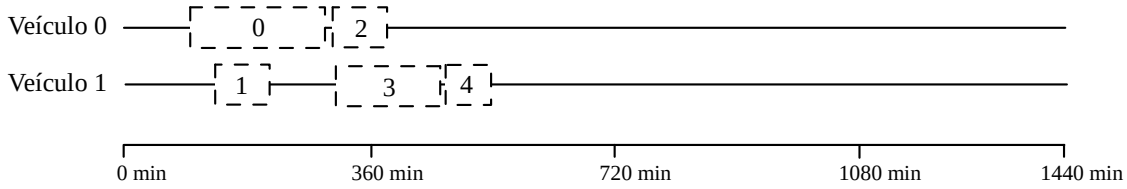

Figura 5.7 – Exemplo da geração de uma solução inicial para os veículos: Passo 4

Passo 5: Aloca-se a próxima viagem da lista ao veículo que gera o menor custo de alocação. Nesse caso, considerando que haveria sobreposição de viagens com a alocação da Viagem 5 ao Veículo 1, a Viagem 5 é alocada ao Veículo 0.

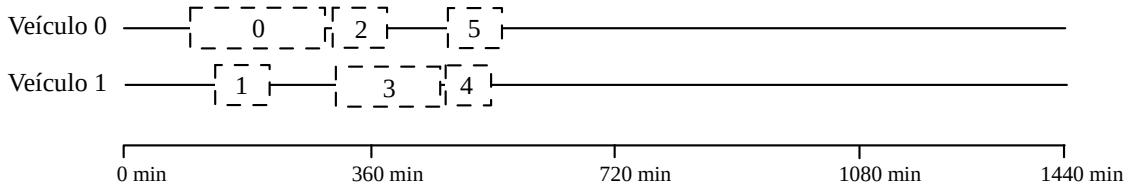

Figura 5.8 – Exemplo da geração de uma solução inicial para os veículos: Passo 5

Passo 6: Aloca-se a próxima viagem da lista no veículo que gera o menor custo de alocação. Nesse caso, a Viagem 6 é alocada ao Veículo 0.

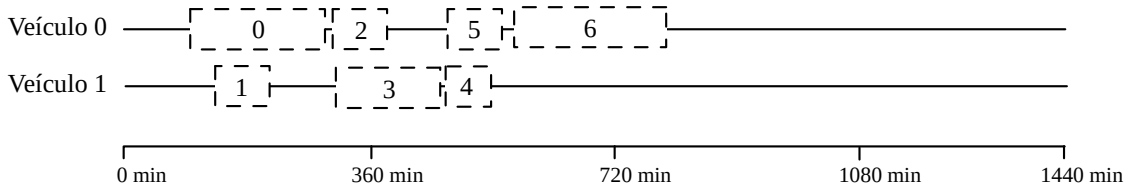

Figura 5.9 – Exemplo da geração de uma solução inicial para os veículos: Passo 6

Passo 7: Aloca-se a próxima viagem da lista no veículo que gera o menor custo de alocação. Nesse caso, a Viagem 7 é alocada ao Veículo 1.

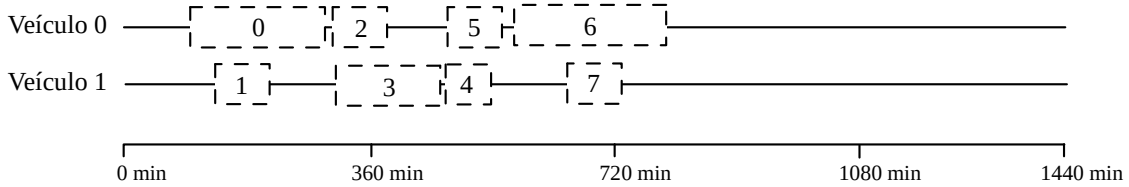

Figura 5.10 – Exemplo da geração de uma solução inicial para os veículos: Passo 7

Passo 8: Aloca-se a próxima viagem da lista no veículo que gera o menor custo de alocação. Nesse caso, a Viagem 8 é alocada ao Veículo 1.

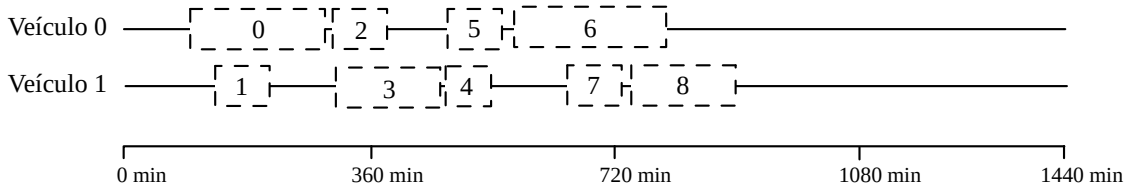

Figura 5.11 – Exemplo da geração de uma solução inicial para os veículos: Passo 8

Passo 9: Aloca-se a próxima viagem da lista no veículo que gera o menor custo de alocação. Nesse caso, a Viagem 9 é alocada ao Veículo 0. Como não há mais viagens a serem alocadas, encerra-se o procedimento de construção da solução inicial para os veículos.

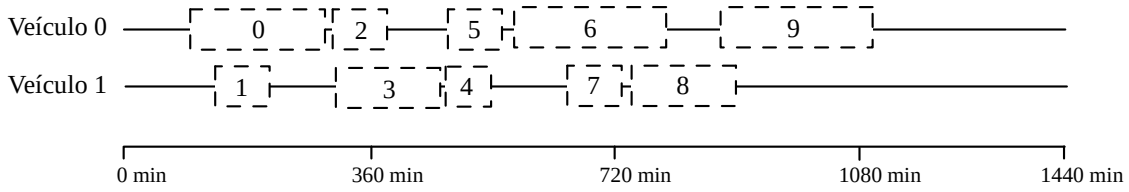

Figura 5.12 – Exemplo da geração de uma solução inicial para os veículos: Passo 9

5.5.2 Método Guloso aplicado à geração de uma solução inicial para o PPT

Nesta forma de construção, considera-se uma lista de tarefas T_i (i = 0, 1, 2, 3, ...), ordenadas pela hora de início, e uma lista de jornadas, onde em cada passo, há sempre uma jornada sem nenhuma tarefa. O método começa com a alocação da tarefa T_0 à primeira jornada. A seguir, para cada tarefa T_i (i = 1, 2, 3, ...) da lista de tarefas, calcula-se o custo da alocação de T_i em cada uma das jornadas já com alguma tarefa alocada e mais uma jornada sem nenhuma tarefa. O custo da alocação é calculado tendo em vista a expressão (8) da seção 5.3.2. A seguir, aloca-se a tarefa T_i à jornada que gerou o menor custo.

O pseudocódigo do método guloso considerado é descrito a seguir na Figura 5.5.

Algoritmo SIG_T

Entrada: $f(.)$, T
Seja T o conjunto de tarefas a serem executadas
Seja s_T uma solução para as tripulações
$s_T \leftarrow \varnothing$
Adicione uma jornada a s_T
para cada $t \in T$ faça
    $melhorDelta \leftarrow \infty$
    para cada jornada $j \in s_T$ faça
        $custoAnterior \leftarrow$ custo da jornada j antes de receber a tarefa t
        Aloque a tarefa t na jornada j
        $custoPosterior \leftarrow$ custo da jornada j após inserir a tarefa t
        Remova a tarefa t da jornada j
        $\Delta = custoPosterior - custoAnterior$
        Se ($\Delta < melhorDelta$) então
            $melhorJornada \leftarrow j$
            $melhorDelta \leftarrow \Delta$
        fim se
    fim para
    se $melhorJornada$ não possuir nenhuma tarefa
        Adicione uma nova jornada à s_T
    Aloque a tarefa t na jornada indicada por $melhorJornada$
    $fo \leftarrow fo + melhorDelta$, ou seja, atualize o valor da função de avaliação
fim para
Remova de s_V a jornada que não possui nenhuma tarefa
Retorne s_V

Figura 5.13 – Método Gerador de Solução Inicial para as Tripulações

Estabelecida a programação dos veículos, é gerado um conjunto de tarefas a serem executadas pelos tripulantes da empresa. Cada tarefa é um conjunto de viagens reunidas de maneira que haja apenas duas oportunidades de troca: uma no início e outra no final da tarefa. Assim, durante sua realização não é possível a troca de tripulação.

A Figura 5.16 ilustra a lista de tarefas geradas a partir da programação dos Veículos da Figura 5.14.

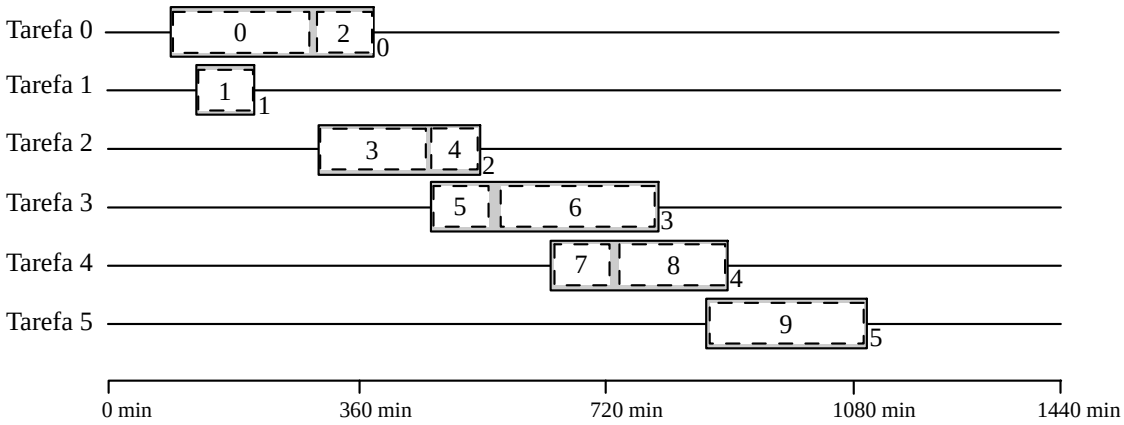

Figura 5.14 – Tarefas geradas pela programação dos veículos da Figura 5.14

Passo 0: Cria-se uma lista com apenas uma jornada, inicialmente sem nenhuma tarefa.

Passo 1: Aloca-se a primeira tarefa da lista de tarefas na única jornada da lista. Posteriormente, adiciona-se uma nova jornada na lista de jornadas. Nesse caso, a Tarefa 0 é adicionada à Jornada 0. Como deve-se sempre avaliar o caso de uma tarefa ser atribuída a uma jornada sem tarefas, a Jornada 1 é adicionada à lista de Jornadas.

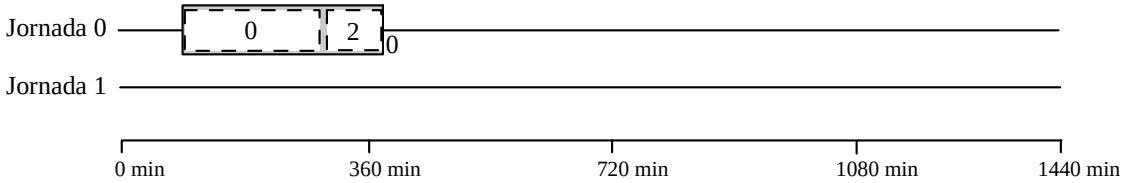

Figura 5.15 – Exemplo da geração de uma solução inicial para as tripulações: Passo 1

Passo 2: Aloca-se a próxima tarefa da lista de tarefas na jornada que gera menor custo. Nesse caso, a Tarefa 1 é alocada à Jornada 1. Como não há jornadas vazias, o procedimento cria uma nova jornada, a Jornada 2, que é adicionada à lista de Jornadas.

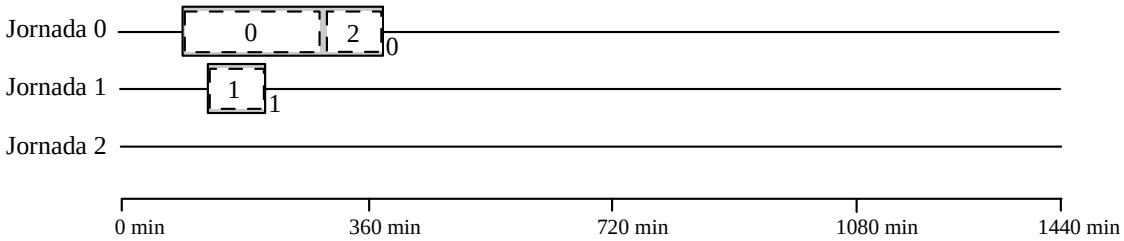

Figura 5.16 – Exemplo da geração de uma solução inicial para as tripulações: Passo 2

Passo 3: Aloca-se a próxima tarefa da lista de tarefas na jornada que gera menor custo. Nesse caso, a Tarefa 2 é alocada à Jornada 1, porque ela geraria uma sobreposição se fosse alocada à Jornada 0 e se fosse alocada à Jornada 2 haveria um custo mais elevado por utilizar uma nova jornada.

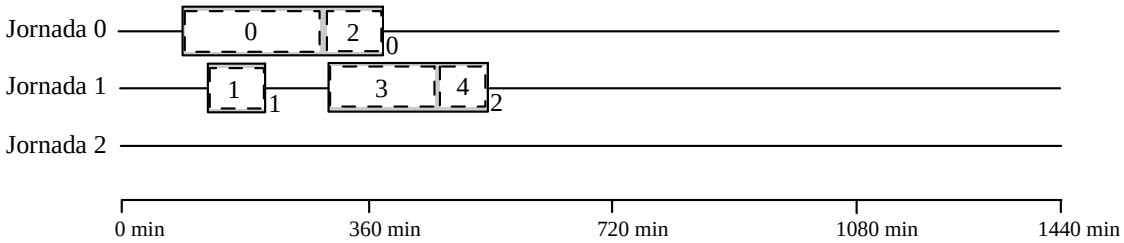

Figura 5.17 – Exemplo da geração de uma solução inicial para as tripulações: Passo 3

Passo 4: Aloca-se a próxima tarefa da lista de tarefas na jornada que gera menor custo. Nesse caso, a Tarefa 3 é alocada à Jornada 2.

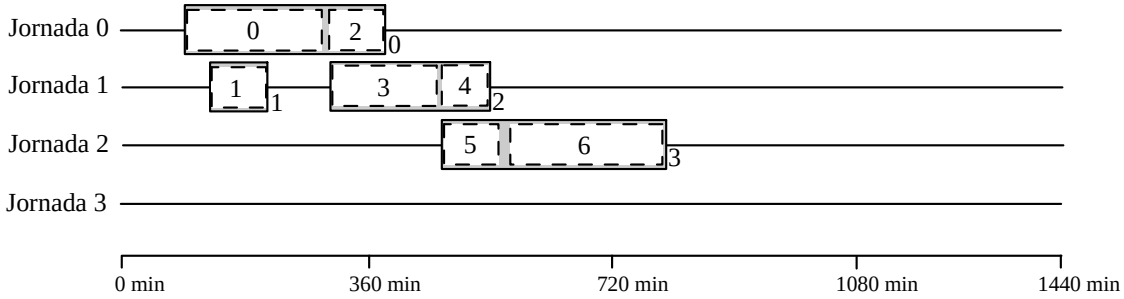

Figura 5.18 – Exemplo da geração de uma solução inicial para as tripulações: Passo 4

Passo 5: Aloca-se a próxima tarefa da lista de tarefas na jornada que gera menor custo. Nesse caso, a Tarefa 4 é alocada à Jornada 0.

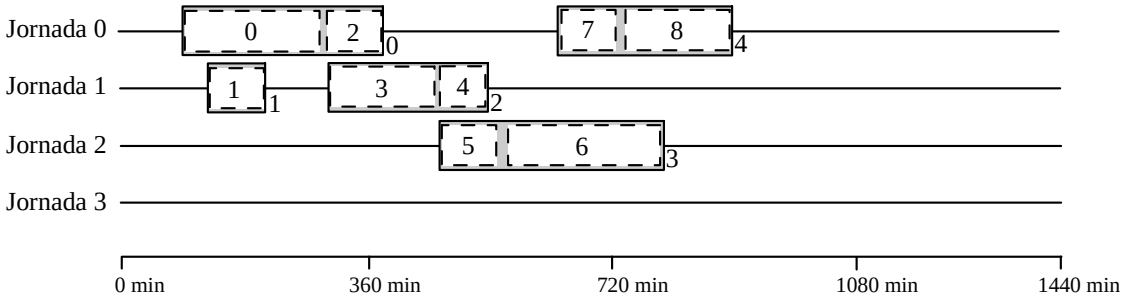

Figura 5.19 – Exemplo da geração de uma solução inicial para as tripulações: Passo 5

Passo 6: Aloca-se a última tarefa da lista na jornada que gera menor custo. Nesse caso, a Tarefa 5 é alocada à Jornada 2 e o procedimento construtivo se encerra.

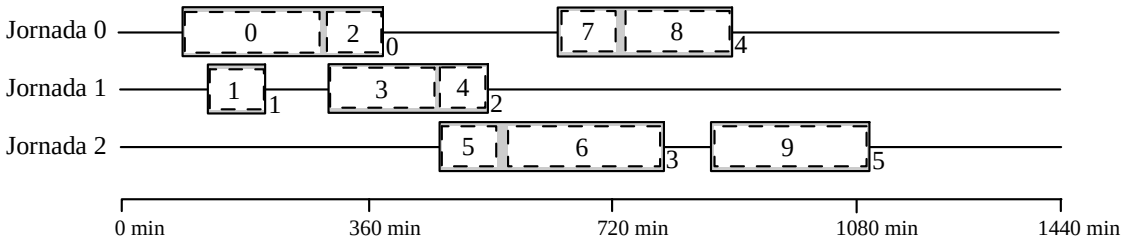

Figura 5.20 – Exemplo da geração de uma solução inicial para as tripulações: Passo 6

5.6 Metaheurísticas aplicadas ao PPVT

Neste trabalho, o refinamento de uma solução é feito pela metaheurística *Iterated Local Search* (ILS) com uma fase de refinamento que combina uma técnica baseada no procedimento *Variable Neighborhood Descent* (VND) em dois níveis com uma heurística clássica e a poderosa metaheurística Busca Tabu com Relaxação Adaptativa.

5.6.1 *Iterated Local Search*

O algoritmo *Iterated Local Search* (ILS) é uma meta-heurística que para escapar das armadilhas dos ótimos locais, efetua perturbações em uma solução *s*. Esse algoritmo se mostra bem sucedido em várias aplicações, motivando, assim, sua utilização na resolução do PPVT. Este algoritmo é baseado na idéia de que um procedimento de busca local pode ser melhorado gerando-se novas soluções de partida, as quais são obtidas por meio de perturbações na solução ótima local.

Neste trabalho, uma perturbação de um nível *i* consiste de efetuar *i*+2 movimentos na vizinhança N_V^{RP} escolhidos aleatoriamente. Desta forma, uma perturbação de nível 0 consiste em efetuar 2 movimentos nesta estrutura de vizinhança. Para evitar buscas exaustivas na fase de refinamento do ILS, os movimentos feitos em uma perturbação não geram inviabilidade segundo o quesito sobreposição de viagens.

O pseudocódigo do ILS aplicado ao PPVT é apresentado na Figura 5.22. Em que, *Nívelmax* é o maior nível de perturbação a ser explorado, *ILSmax* é o número de iterações em que um mesmo nível de perturbação é explorado, *Tempomax* é o tempo limite de processamento do ILS e *f*(.) é uma função que avalia uma solução.

Basicamente, este método possui duas fases a cada iteração: a primeira é a fase de perturbação, e a segunda, a de refinamento. A fase de refinamento do ILS é descrita em detalhes na seção 5.6.2.

```
Algoritmo ILS_PPVT
1:  Entrada: s, f(.), Nívelmax , ILSmax , Tempomax
2:  s ← VND_2N(s)
3:  s* ← s
4:  nível ← 0
5:  enquanto nível < Nivelmax faça
6:     iter ← 0
7:     enquanto iter < ILSmax e tempo < Tempomax faça
8:        s ← Perturbação(s, nível)
9:        s ← VND_2N(s)
10:       se f(s) < f(s*) então
11:          s* ← s
12:          nível ← 0
13:          iter ← 0
14:       senão
15:          s ← s*
16:          iter ← iter + 1
17:       fim se
18:    fim enquanto
19:    nível ← nível +1
20: fim enquanto
21: retorne s*
```

Figura 5.21 – Algoritmo ILS com VND_2N aplicado ao PPVT

5.6.2 *Variable Neighborhood Descent*

Um ótimo local com relação a uma dada estrutura de vizinhança não corresponde necessariamente a um ótimo local com relação a uma outra estrutura de vizinhança. Este é um dos princípios básicos do método de descida em vizinhança variável, *Variable Neighborhood Descent* (VND). Proposto por Nenad Mladenovic e Pierre Hans, este é um método de refinamento que consiste em explorar o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança, aceitando somente soluções de melhora da solução corrente e retornando à primeira estrutura quando uma solução melhor é encontrada.

Neste trabalho, o algoritmo proposto para o refinamento de uma solução após a perturbação causada no ILS é baseado na técnica VND. Aqui, o algoritmo, além de alterar a estrutura de vizinhança, também altera o método de refinamento utilizado. Esse algoritmo consiste em uma aplicação em dois níveis, um externo e um interno, da idéia VND. A Figura 5.9 é uma representação descritiva do algoritmo proposto para o refinamento do ILS.

As duas linhas dos passos 1, 2 e 3 do algoritmo em questão constituem o VND interno. No passo 1 aplica-se a uma solução s_0 o método DR2VP utilizando a vizinhança N_V^{RP} resultando uma solução s_1. Como DR2VP é um método de descida, a solução s_1 ou é melhor que a solução s_0 ou é a própria solução s_0. Visto que este é um problema de minimização, uma solução é melhor que outra se o valor de sua função de avaliação (seção 5.3) é inferior ao valor da função de avaliação da outra solução. E então, a solução s_1 é atribuída à solução s_0. O algoritmo continua com a chamada do método DR2VP usando a estrutura de vizinhança N_V^{TP}, gerando a solução s_2. Se o custo da solução s_2 for inferior ao custo de s_0, então se inicia novamente a execução do passo 1. O primeiro VND interno termina quando os métodos DR2VP não produzirem melhoras nas soluções. A execução dos passos 2 e 3 é semelhante a deste passo: no passo 2 o refinamento de uma solução é feito por meio do método BTRA

(Busca Tabu com Relaxação Adaptativa), primeiro utilizando a vizinhança N_T^R e depois, a vizinhança N_T^T; e no passo 3, o refinamento é feito pelo método DR2VI usando a vizinhança N_V^{RP} e depois a N_V^{TP}.

Alg.	**VND_2N**
Início	Seja s_0 uma solução
Passo 1	$s_1 \leftarrow$ VDR(s_0, *VDRmax*, N_v^R, N_v^{RP}, f_v)
	$s_2 \leftarrow$ VDR(s_1, *VDRmax*, N_v^T, N_v^{TP}, f_v)
	Se s_2 melhor que s_0, faça $s_0 = s_2$ e volte ao início desse passo
Passo 2	$s_3 \leftarrow$ BTRA (s_2, *BTtempo*, N_T^R, f_t)
	$s_4 \leftarrow$ BTRA (s_3, *BTtempo*, N_T^T, f_t)
	Se s_4 melhor que s_2, faça $s_2 = s_4$ e volte ao início desse passo
Passo 3	$s_5 \leftarrow$ IRD(s_4, *IDRmax*, N_v^R, N_v^{RP}, f, λ)
	$s_6 \leftarrow$ IRD(s_5, *IDRmax*, N_v^R, N_v^{RP}, f, λ)
	Se s_6 melhor que s_4, faça $s_4 = s_6$ e volte ao início desse passo
	Se houve alguma melhora no Passo 3, volte ao Passo 2
Passo 4	Retorne s_6

Figura 5.22 – Algoritmo VND em dois níveis como refinamento do ILS

O VND externo consiste em tomar passos como tipos de vizinhança no VND original (seção 3.3.2). A terceira linha do passo 3 verifica se, nesse passo, houve alguma melhora na solução. Caso haver, o método inicia novamente sua busca a partir da primeira estrutura, aqui a partir do passo 2. O VND externo é feito somente utilizando os passos 2 e 3.

5.6.3 Busca Tabu com Relaxação Adaptativa

Para o refinamento de uma solução PPVT utilizando Busca Tabu com Relaxação Adaptativa adaptou-se o método BTAR-FI-IC proposto por Marinho (2005) para a abordagem do problema integrado. Este método, cujo pseudocódigo é apresentado na Figura 5.24, consiste no método Busca Tabu acrescido do mecanismo de intensificação Relaxação Adaptativa.

O mecanismo de Relaxação Adaptativa utilizado pelo método BTAR-FI-IC é baseado naquele descrito por Schaerf (1996), onde os pesos para cada fonte de inviabilidade são ajustados dinamicamente, como proposto originalmente por Gendreau *et al*. (1994). Para cada inviabilidade *i* o peso β_i é multiplicado por um fator σ_i que varia de acordo com o seguinte esquema:

- No início da busca $\sigma_i \leftarrow 1$.
- A cada *k* movimentos:
  - se todas as *k* soluções visitadas são factíveis em relação à restrição *i* então $\sigma_i \leftarrow \sigma_i / \gamma$;
  - se todas as *k* soluções visitadas são infactíveis em relação à restrição *i* então $\sigma_i \leftarrow \sigma_i \times \gamma$;
  - se algumas soluções são factíveis e outras são infactíveis, considerando a restrição *i*, então σ_i permanece inalterado.

O parâmetro γ é randomicamente selecionado, a cada k movimentos, no intervalo [1,8; 2,2] conforme proposto por Schaerf (1996). Cada valor de σ_i é limitado por duas constantes σ_{min} e σ_{max}, de forma a evitar que a relaxação adaptativa incremente/decremente indefinidamente os pesos para restrições que são sempre insatisfeitas/satisfeitas.

Algoritmo Busca Tabu com Relaxação Adaptativa

```
1:  Entrada: s, BTtempo, N_t (N_t^R ou N_t^T), f_t(.)
2:  s* ← s;                                          { Melhor solução até então }
3:  ListaTabu ← ∅;                                   { Lista Tabu}
4:  Iter ← 0;                                        { Contador do número de iterações }
5:  repita
6:      Defina um subconjunto V ⊂ N_t(s);
7:      melhorMovimento ← movimentoRandomico( V );   { Melhor Movimento em V ⊂ N(s) }
8:      melhorCustoDinamico ← ∞;                     { Melhor custo dinâmico }
9:      para todo ( movimento m ∈ V ) faça
10:         se ( f( s ⊕ m ) < f( s* ) ) então
11:             melhorMovimento ← m;
12:             interromper;
13:         senão
14:             se ( m ∉ ListaTabu ) então
15:                 se ( f( s ⊕ m ) < f( s ) ) então
16:                     melhorMovimeto ← m;
17:                     interromper;
18:                 senão
19:                     se ( custoPorPesosDinamicos( s ⊕ m ) < melhorCustoDinamico ) então
20:                         melhorMovimento ← m;
21:                         melhorCustoDinamico ← custoPorPesosDinamicos ( s ⊕ m );
22:                     fim-se
23:                 fim-se
24:             fim-se
25:         fim-se
26:     fim-se
27:     fim-para
28:     s ← s ⊕ melhorMovimento;
29:     se ( f( s ) < f( s* ) ) então
30:         s* ← s;
31:     fim-se
32:     AtualizarListaTabu();
33:     se IteracaoAtualizacao() então
34:         AtualizarPesosDinamicos();
35:     fim-se
36: até que ( Critério de parada seja satisfeito )
37: retorne s*;
```

Figura 5.23 – Algoritmo Busca Tabu com Relaxação Adaptativa aplicado ao PPVT
FONTE: Marinho (2005)

5.6.4 O procedimento *Vehicle Random Descent*

Para problemas grandes, encontrar o melhor vizinho em cada iteração de um método de descida é muito custoso. Nesses casos, normalmente são utilizadas abordagens randômicas. O clássico procedimento Descida Randômica (DR) funciona dessa forma: é gerado um ponto aleatório *s*' em uma dada vizinhança. Se *s*' for melhor que a solução atual, então *s* é atualizada (*s* ← *s*') e a busca continua de *s*'. Se não houver melhora em *RDmax* iterações, então a busca é finalizada. Note que o procedimento pode não retornar um ótimo local.

Quando se trata de problemas integrados, encontrar o melhor vizinho a cada iteração é uma operação ainda mais custosa. Isto se deve ao fato de que, nas vizinhanças integradas consideradas (N_v^{RP} e N_v^{TP}), cada modificação nos veículos tem reflexo na programação das tripulações. Para reduzir o número de propagações dessas modificações na programação das tripulações, propõe-se o procedimento *Vehicle Random Descent* (VRD), descrito na Figura 5.24.

O princípio de funcionamento desse algoritmo é bem simples e lembra bastante o clássico Descida Randômica. A principal diferença é que nesse procedimento, apenas as modificações feitas nos veículos (realocação ou troca) que melhoram a função de avaliação f_v são propagadas na programação das tripulações.

Na Figura 5.25, *s* é uma solução formada pelo par de soluções s_v e s_t, *VRDmax* é o número máximo de iterações sem melhora, N_v é uma estrutura de vizinhança (realocação ou troca) especifica para os veículos, *m* é o movimento realizado na estrutura considerada, N_v^P é a estrutura de vizinhança (realocação ou troca) com a propagação segundo a modificação *m* e f_v é a função de avaliação que avalia a parte s_v da solução *s*.

Observa-se que a solução final desse algoritmo pode, não necessariamente, ser melhor que a solução inicial em relação à função integrada *f* uma vez que só é usada a função de avaliação de veículos f_v.

Algoritmo VRD

1: Entrada: *s*, *VRDmax*, $N_v(.)$, $N_v^P(.)$, $f_v(.)$
2: Saída: Solução *s** melhor ou igual à solução *s* de acordo com a função f_v
3: *iter* ← 0
4: **enquanto** *iter* < *VRDmax* **faça**
5: $m \leftarrow movimentoAleatorio(s, N_v)$
6: $s' \leftarrow vizinho(s, m)$
7: **se** *s' for melhor que s de acordo com* f_v **faça**
8: $s \leftarrow vizinho(s, N_v^P, m)$
9: *iter* ← 0
10: **fim se**
11: *iter* ← *iter* +1
12: **fim enquanto**
13: *s** ← *s*
14: **retorne** *s**

Figura 5.24 – Procedimento *Vehicle Random Descent*

5.6.5 O procedimento *Integrated Vehicle and Crew Random Descent Procedure*

Esta estratégia é semelhante à apresentada na seção anterior. Há duas diferenças básicas. A primeira é que são candidatos apenas os movimentos que produzirem soluções, com respeito aos veículos, melhores que f_v ou priores que f_v em até $(\lambda - 1)\times 100\%$. A segunda, é que se considera o custo tanto de s_v quanto de s_t, o que não acontece no *Vehicle Random Descent* que, quando aplicado à resolução do PPVT, considera apenas o custo de s_v. O objetivo desse algoritmo é refinar uma solução *s* usando movimentos integrados sem onerar demasiadamente o processamento, como aconteceria se o refinamento fosse feito pelo algoritmo clássico descida randômica usando unicamente movimentos integrados.

A Figura 5.26 apresenta o *Integrated Vehicle and Crew Random Descent Procedure* (IRD). Nesta figura, *s* é uma solução formada pelo par de soluções s_v e s_t compatíveis entre si, *IRDmax* é o número de iterações sem melhora em *f*, *m* é um movimento realizado nos veículos de acordo com a vizinhança N_v (realocação ou troca), N_v^P é a estrutura de vizinhança

(realocação ou troca) com propagação segundo a modificação m, f_v é a função que avalia s_v e λ ($\lambda \geq 1$) é um valor usado para determinar o quanto uma solução dos veículos pode piorar para que valha a pena efetuar a parte mais custosa do refinamento, a construção e a propagação do movimento integrado. Assim, esse valor determina se é um momento propício para testar o movimento com propagação. λ = 1,01 implica que se um movimento gerar uma solução s_v' de custo maior que 101% do custo de s_v, o método não testa o movimento de propagação; caso contrário, o teste é realizado e, se a solução s' resultante do movimento de propagação for melhor que a solução s, então a solução s é atualizada, $s \leftarrow s$'. Testes realizados mostraram que, para as instancias analisadas, 1,01 é um bom valor para λ e este é o valor utilizado neste trabalho.

Algoritmo IRD

```
1:  Entrada: s, IRDmax, N_v(.), N_v^P(.), f(.), Real λ
2:  Saída: Solução s* melhor ou igual à solução s de acordo com a função f
3:  iter ← 0
4:  enquanto iter < IRDmax faça
5:     m ← movimentoAleatorio(s, N_v)
6:     s' ← vizinho(s, m)
7:     se f_v(s') ≤ f_v(s) × λ faça
8:        s'' ← vizinho(s, N_v^P, m)
9:        se s'' for melhor que s de acordo com f faça
10:          s ← s''
11:          iter ← 0
12:       fim se
13:    fim se
14:    iter ← iter +1
15: fim enquanto
16: s* ← s
17: retorne s*
```

Figura 5.25 – Procedimento *Integrated Vehicle and Crew Random Descent*

6 Resultados

Apresentam-se, nesse capítulo, os resultados obtidos aplicando-se a heurística ILS com VND em dois níveis utilizando Busca Tabu com Relaxação Adaptativa, denotada por ILS-VND-BTRA, conforme descrito na seção 5.6.

O algoritmo descrito foi implementado na linguagem C++ usando o compilador Borland C++ Builder 6.0 e testado em um microcomputador com processador Intel® Pentium®, 3.00 GHz, com 1 GB de memória RAM, sob sistema operacional Windows XP Professional Service Pack 2.

Para validar as abordagens foram disponibilizadas pela empresa responsável pelo gerenciamento do Sistema de Transporte Público da cidade de Belo Horizonte, instâncias reais pertencentes a uma Bacia de Linhas de tal cidade.

A seção 6.1 apresenta os resultados obtidos utilizando a primeira abordagem da função de avaliação, enquanto na seção 6.2 apresentam-se os resultados obtidos considerando a segunda abordagem da função de avaliação.

6.1 Resultados – 1ª Abordagem

Nas tabelas 6.1 e 6.2 são apresentados os valores dos parâmetros utilizados para o cálculo da função de avaliação apresentada na seção 5.3, a qual é baseada em pesos empíricos. Na Tabela 6.2 os valores *TempDesloc_Lim* e *TempTerm_Lim* são vinculados a cada instância e representam os maiores tempos de viagem morta e terminal, nessa ordem, em minutos, encontrados na solução gerada pela empresa.

Tabela 6.1 – Parâmetros utilizados na função de avaliação do PPVT

Parâmetro	Valor
MaxTimeTerm	119 minutos
α_1	2
β_1	ln (16) / ln (α1 x *TempDesloc_Lim*)
α_2	1
β_2	ln (16) / ln (α2 x *TempTerm_Lim*)
ω_{TD}	50
ω_{TT}	40
ω_{TS}	80
ω_{EDP}	80
ω_{TL}	5
ω_{NV}	0
μ_1	18
μ_2	40
μ_3	19
μ_4	5
μ_5	5
λ_1	40
λ_2	40
λ_3	40
λ_4	45
λ_5	35
λ_6	35
λ_7	45
δ	100

Tabela 6.2 – Instâncias e seus respectivos tempos limites relativos ao PPV

Instâncias	*TempDesloc_Lim*	*TempTerm_Lim*
G02_DOM	18	73
G02_SEG	18	118
G02_SEX	18	113
G02_SAB	18	62
G27_DOM	23	110
G27_SEG	23	47
G27_SEX	23	42
G27_SAB	23	57

No cálculo de β_1 e β_2 da Tabela 6.1, o valor 16 corresponde à penalidade máxima atribuída a uma viagem morta ou tempo de espera no terminal que obedece ao limite estabelecido por *TempDesloc_Lim* ou *TempTerm_Lim* de acordo com o que se esteja penalizando. Ou seja, trata-se do valor máximo da função exponencial utilizada no calculo da função de avaliação.

Para as empresas em questão, considerou-se o número de duplas pegadas na programação dos veículos inferior a 60% do tamanho da frota; enquanto que na programação dos tripulantes esse número foi considerado inferior a 1,25 × *tamanho da frota de veículos*.

Esta última consideração é uma estimativa para o número de tripulantes, que corresponde a 2,5 tripulantes para cada veículo.

Os parâmetros utilizados no método ILS-VND-BTRA, utilizado para a abordagem integrada, foram, na chamada do método ILS, *Tempomax* = 30 minutos, *Nivelmax* = 5 e *ILSmax* = 50; *Itermax = total de viagens* × *tamanho máximo da frota de veículos* nas chamadas dos métodos DR2VP e DR2VI. O valor da constante ρ, parâmetro do método DR2VI, é fixado em 1,2. Os pesos α e β relativos à programação dos veículos e das tripulações como descrito na seção 5.3, são fixados em 1,0 e 2,5, respectivamente.

Para os métodos destinados a resolver os problemas de programação de veículos e tripulações de forma seqüencial o tempo de processamento foi limitado a 15 minutos para cada fase, isto é, 15 minutos para a resolução do PPV pelo método ILS proposto em Simões *et al*. (2006) e 15 minutos para a resolução do PPT pelo método baseado em Busca Tabu, proposto em Marinho *et al*. 2004.

A Tabela 6.3 mostra uma comparação entre os resultados produzidos pelas metodologias, seqüencial e integrada, em termos da função de avaliação apresentada na seção 5.3. Foram analisadas duas das empresas que operam na Bacia contemplada, considerando os quatro dias da semana que apresentam quadros de horários distintos: segunda-feira, sexta-feira, sábado e domingo. Nesta tabela, considera-se que "FO" é o custo da programação realizada pela empresa e "Melhor FO", "Média FO", "Desvio", "PM" e "PMM" são, respectivamente, o menor custo encontrado pelo método considerado, a média dos custos em 10 execuções, o percentual de desvio calculado conforme fórmula a seguir.

$$desvio = \frac{MediaFO - MelhorFO}{MelhorFO} \tag{12}$$

Tabela 6.3 – Comparação de resultados (Seqüencial x Integrada)

	Algoritmo Sequencial			Algoritmo Integrado		
Instancia	Melhor FO	Média FO	Desvio	Melhor FO	Média FO	Desvio
G02_DOM	7036	7160,35	6,47%	6725,5	6841,5	**1,72%**
G02_SEG	22507,5	22828,55	8,26%	21087,5	21297,7	**1,00%**
G02_SEX	23045,5	23354,1	7,50%	21724,5	21910,1	**0,85%**
G02_SAB	12938,5	13260,5	8,51%	12220	12468,05	**2,03%**
G27_DOM	3497,5	3550,75	4,51%	3397,5	3442,3	**1,32%**
G27_SEG	8299	8463,25	7,30%	7887,5	7992,55	**1,33%**
G27_SEX	8429,5	8486,25	7,59%	7887,5	8015,55	**1,62%**
G27_SAB	5193,5	5346,65	5,73%	5057	5176,35	**2,36%**

Pelos resultados exibidos na Tabela 6.3, verifica-se que, quanto aos desvios, a metodologia integrada proposta apresenta-se superior à metodologia seqüencial em todos os problemas teste. Assim, em todos os dias analisados, o método proposto foi capaz de gerar, em média, soluções melhores que a da metodologia seqüencial, sob o ponto de vista da função de avaliação.

Nos domingos da G02, a metodologia integrada mostrou-se melhor por apresentar um desvio de 1,72%, muito inferior ao desvio de 6,47% da metodologia seqüencial. Nas segundas-feiras da mesma empresa, a metodologia integrada mostrou-se melhor por apresentar um desvio de 1,00%, inferior ao desvio de 8,26% da metodologia seqüencial. Nas

sextas-feiras da G02, a metodologia integrada mostrou-se melhor por apresentar um desvio de 0,85%, inferior ao desvio de 7,50% da metodologia seqüencial. E, nos sábados da G02, a metodologia integrada mostrou-se melhor por apresentar um desvio de 2,03%, inferior ao desvio de 8,51% da metodologia seqüencial. Para a empresa G27 o método integrado também obteve melhoras significativas quanto ao seqüencial.

O método integrado proposto foi capaz de gerar soluções melhores em todos os oito problemas teste (domingos, segundas, sextas e sábados das empresas G02 e G27). Além disso, ao se analisar os valores de desvio apresentados na Tabela 6.3 para a integração observa-se que eles são baixos. Este é um ponto importante, uma vez que não há grandes discrepâncias entre os custos encontrados para uma mesma instância.

As tabelas 6.4 e 6.5 mostram, detalhadamente, as características das melhores soluções geradas pelas metodologias utilizadas nos quatro dias da semana da empresa G02 e da empresa G27, respectivamente, em relação ao total de viagens a serem realizadas "NVIAG", número de veículos utilizados "NVEIC", tempo de espera no terminal "TE", tempo de viagem morta "TVM", número de duplas pegadas dos veículos "NDP(v)" e dos tripulantes "NDP(t)", número de tripulantes "NTRIP", tempo ocioso "TO", tempo de hora extra "THE", número de trocas de linha "NTL", número de trocas de ponto "NTP", número de trocas de veículos "NTV" função objetivo de veículos "f(v)", função objetivo de tripulações "f(t)", função objetivo de veículos e tripulações "f(v,t)".

Os números em negrito apontados nessas tabelas mostram melhoras da metodologia integrada em relação à metodologia seqüencial. Assim, por exemplo, nos domingos da empresa G02, há uma redução de 2h29min no tempo de espera em terminal (de 16h13min para 13h44min), uma redução de 2 tripulantes (de 21 para 19), 4h6min no tempo de ociosidade das tripulações (de 15h13min para 11h07min) e uma redução de 4 trocas de veículos (de 6 para 2). É importante observar que na função de avaliação dos veículos não é penalizado o número de veículos.

Tabela 6.4 – Características das melhores soluções das metodologias seqüencial e integrada para a empresa G02

Empresa G02	Algoritmo Sequencial				Algoritmo Integrado			
	DOM	SEG	SEX	SAB	DOM	SEG	SEX	SAB
NVIAG	90	260	260	172	90	260	260	172
NVEIC	11	40	40	23	11	40	**39**	23
TE	16:13	22:26	5:56	21:02	**13:44**	**8:12**	**22:14**	21:02
TVM	8:24	12:50	12:14	16:48	9:50	15:06	14:58	18:36
NDP(v)	3	21	20	5	5	24	24	8
NDP(t)	12	50	50	16	**7**	**49**	**43**	20
NTRIP	21	62	63	37	**19**	62	63	**34**
TO	15:13	15:04	9:23	15:44	**11:07**	**7:52**	**6:28**	**15:12**
THE	5:04	6:44	7:34	20:08	14:58	8:28	12:31	**19:44**
NTL	1	5	7	1	4	**4**	**6**	2
NTP	2	18	19	9	3	**11**	**8**	10
NTV	6	62	66	11	**2**	**34**	**34**	15
f(v)	841	2510	2553	1491	893	2740	2887	1585
f(t)	2478	7999	8197	4579	**2333**	**7339**	**7535**	**4254**
f(v,t)	7036	22507.5	23045.5	12938.5	**6725.5**	**21087.5**	**21724.5**	**12220**

Tabela 6.5 – Características das melhores soluções das metodologias seqüencial e integrada para a empresa G27

Empresa G27	Algoritmo Sequencial				Algoritmo Integrado			
	DOM	SEG	SEX	SAB	DOM	SEG	SEX	SAB
NVIAG	52	98	98	69	52	98	98	69
NVEIC	5	14	13	7	5	14	13	7
TE	12:57	10:25	8:45	11:12	**10:47**	**10:22**	11:08	**10:54**
TVM	3:50	16:06	15:20	8:26	4:36	16:52	15:20	9:12
NDP(v)	0	7	7	4	1	8	7	5
NDP(t)	6	17	16	8	7	**13**	**13**	**6**
NTRIP	10	24	25	15	10	**23**	**23**	**14**
TO	6:30	10:52	13:44	10:14	**5:06**	**10:06**	**8:57**	**8:54**
THE	6:53	5:58	6:29	7:33	**6:35**	11:38	14:42	13:19
NTL	0	0	0	0	0	0	0	0
NTP	0	3	2	1	0	3	3	**0**
NTV	4	14	12	4	**2**	**6**	**5**	**3**
f(v)	370	1019	952	661	385	1050	1005	682
f(t)	1251	2912	2991	1813	**1205**	**2735**	**2753**	**1750**
f(v,t)	3497.5	8299	8429.5	5193.5	**3397.5**	**7887.5**	**7887.5**	**5057**

Pelas tabelas 6.4 e 6.5 verifica-se que, em todos os dias e nas duas empresas, o método proposto foi capaz de gerar soluções melhores, sob o ponto de vista da função de avaliação, que a da metodologia seqüencial. A aplicação da metodologia integrada desenvolvida neste trabalho é vantajosa para situações em que, para a empresa, os custos envolvidos são maiores para os tripulantes, e não para os veículos. Pelos resultados obtidos observa-se que, quanto à integração, há um aumento sutil no custo dos veículos e uma redução significativa no custo dos tripulantes, produzindo assim soluções melhores para o conjunto.

6.2 Resultados – 2ª Abordagem

Nesta abordagem, assim como na anterior, testes preliminares foram realizados com a finalidade de calibrar os parâmetros do ILS-VND-BTRA. Em especial, na segunda abordagem, o número de iterações sem melhora do ILS (*ILSmax*) foi fixado em 50. O tempo de cada chamada do BTRA (BTtempo) foi limitado a 20 segundos. O número máximo de iterações sem melhora do VRD (VRDmax) foi fixado em 100. O número de iterações sem melhora do IRD (*IRDmax*) foi fixado em 50. E, o tempo total de processamento foi limitado a 20 minutos.

A primeira fase da abordagem seqüencial usa o algoritmo ILS descrito em Sousa *et al*. (2007) e a segunda, o algoritmo BTRA descrito em Marinho (2004). Os dois algoritmos são interrompidos após 10 minutos de processamento. Nesta abordagem, *ILSmax* é fixado em 50, uma perturbação de nível *i* é feita aplicando-se *i* + 2 movimentos da vizinhança N_v^R, a busca local do ILS é feita pela aplicação de dois métodos de descida randômica, o primeiro usando a vizinhança N_v^R, e o segundo usando a vizinhança N_v^S.

Para testar os algoritmos, foram examinados problemas de doze problemas-teste disponíveis em http://www.decom.ufop.br/prof/marcone/pt/instances/VC.zip, referentes a dados de nove empresas de transporte público.

A Tabela 6.6 apresenta uma comparação dos resultados da metodologia seqüencial e da integrada em relação ao melhor valor, ao valor médio e ao desvio. Além disso, esta tabela mostra a melhora obtida em cada caso.

Tabela 6.6 – Comparação de resultados dos algoritmos seqüencial e integrado com função de avaliação baseada em custos

	Tradicional			Integrado			Melhora	
Problema	Melhor	Média	Desvio	Melho	Média	Desvio	Melhor	Média
VC52	3936,96	3988,46	2,97%	3873,31	3895,43	0,57%	1,64%	2,39%
VC90	7519,20	7561,96	1,14%	7476,57	7518,14	0,56%	0,57%	0,59%
VC98a	9638,79	9675,39	1,16%	9564,95	9594,67	0,31%	0,77%	0,84%
VC98b	9500,16	9514,50	1,47%	9376,67	9398,28	0,23%	1,32%	1,24%
VC260a	28479,80	28714,00	1,43%	28310,50	28329,11	0,07%	0,60%	1,36%
VC260b	28064,80	28215,82	0,54%	27921,40	27950,78	0,11%	0,51%	0,95%
VC498	32673,50	32756,81	0,26%	32500,10	32612,90	0,35%	0,53%	0,44%
VC538	18917,00	18964,83	0,25%	18887,20	18939,01	0,27%	0,16%	0,14%
VC639a	39478,10	39586,00	0,27%	39301,30	39398,46	0,25%	0,45%	0,48%
VC639b	42557,60	42608,22	0,12%	42336,20	42436,66	0,24%	0,52%	0,40%
VC1036	48764,60	48840,86	0,16%	48669,10	48778,18	0,22%	0,20%	0,13%
VC1038	49223,90	49335,35	0,23%	49085,40	49140,27	0,11%	0,28%	0,40%

Pela Tabela 6.6 pode-se observar que os dois algoritmos têm bom desempenho, com os melhores valores e valores médios bem próximos, e com desvios baixos. Observa-se, no entanto, um desempenho melhor do algoritmo integrado para todos os casos analisados. Por exemplo, há uma melhora de até 1,64% em relação ao valor da melhor solução encontrada em relação ao seqüencial e de 2,30% em relação ao valor médio. Adicionalmente, o desvio é menor em todos os casos, o que mostra a sua robustez do algoritmo integrado proposto.

A Tabela 6.7 apresenta as características das melhores soluções encontradas em cada problema-teste, com respeito à metodologia seqüencial e integrada.

Por esta tabela observa-se que o número de veículos permanece o mesmo nos dois algoritmos. Com relação ao número de tripulantes houve redução de um tripulante nos problemas teste VC52, VC260a, VC498. Nas instâncias VC90 e VC639, o número de tripulantes foi maior, mas houve uma redução significativa no tempo de hora-extra e de ociosidade. Para os outros problemas, o número de tripulantes permaneceu o mesmo pelos dois algoritmos. Uma observação interessante é que a abordagem integrada produziu soluções piores para os veículos nos problemas VC498 e VC538. Por outro lado, a programação dos tripulantes passou a ser melhor que a encontrada na metodologia seqüencial. Além disso, a solução final foi sempre melhor na metodologia integrada.

Tabela 6.7 – Características das melhores soluções dos algoritmos seqüencial e integrado com função de avaliação baseada em custos

Problema-teste	**VC52**	**VC52**	**VC90**	**VC90**	**VC98a**	**VC98a**	**VC98b**	**VC98b**
Algoritmo	**Seq.**	**Int.**	**Seq.**	**Int.**	**Seq.**	**Int.**	**Seq.**	**Int.**
Número de veículos	5	5	10	10	13	13	13	13
Tempo ocioso dos veículos (hh:mm)	12:57	12:57	16:13	16:13	11:58	11:58	10:21	10:21
Tempo de viagem morta (hh:mm)	03:50	03:50	8:24	08:24	15:20	15:20	14:34	14:34
Número de pegadas duplas (veículos)	0	0	4	4	7	7	6	6
Número de tripulações	12	11	20	21	27	27	25	25
Tempo ocioso das tripulações (hh:mm)	11:31	08:31	18:44	14:28	17:05	13:29	12:50	09:29
Tempo de hora extra (hh:mm)	02:25	05:52	14:13	10:00	06:38	05:09	15:13	10:59
Número de pegadas dúplas (tripulações)	2	2	1	3	7	7	5	5
f_v	2640,4	2640,4	5168,4	5168,4	6809,8	6809,75	6744,1	6744,1
f_t	1296,6	1232,9	2350,8	2308,2	2829,0	2755,2	2756,1	2632,6
f_v+f_t	3937,0	3873,3	7519,2	7476,6	9638,8	9565,0	9500,2	9376,7
Problema-teste	**VC260a**	**VC260a**	**VC260b**	**VC260b**	**VC498**	**VC498**	**VC538**	**VC538**
Algoritmo	**Seq.**	**Int.**	**Seq.**	**Int.**	**Seq.**	**Int.**	**Seq.**	**Int.**
Número de veículos	40	40	39	39	47	47	24	24
Tempo ocioso dos veículos (hh:mm)	19:15	19:15	13:22	22:02	37:38	39:27	65:26	65:28
Tempo de viagem morta (hh:mm)	37:54	37:54	37:54	37:54	23:36	24:48	18:59	20:23
Número de pegadas duplas (veículos)	22	22	23	23	25	26	3	3
Número de tripulações	76	75	76	76	94	93	53	53
Tempo ocioso das tripulações (hh:mm)	91:31	83:03	84:10	79:07	55:03	44:51	50:55	47:57
Tempo de hora extra (hh:mm)	07:14	08:46	11:28	07:19	24:36	22:17	33:17	29:48
Número de pegadas dúplas (tripulações)	17	17	17	17	19	19	2	2
f_v	20189,0	20189,0	19785,3	19785,3	22851,1	22942,8	12755,6	12830,1
f_t	8290,8	8121,5	8279,5	8136,19	9822,4	9557,3	6161,36	6057,1
f_v+f_t	28479,8	28310,5	28064,8	27921,4	32673,5	32500,1	18917	18887,2
Problema-teste	**VC639a**	**VC639a**	**VC639b**	**VC639b**	**VC1036**	**VC1036**	**VC1038**	**VC1038**
Algoritmo	**Seq.**	**Int.**	**Seq.**	**Int.**	**Seq.**	**Int.**	**Seq.**	**Int.**
Número de veículos	55	55	60	60	69	69	70	70
Tempo ocioso dos veículos (hh:mm)	50:36	51:45	50:42	52:16	66:07	69:13	67:41	69:10
Tempo de viagem morta (hh:mm)	52:03	52:37	32:59	56:06	50:28	51:41	50:32	51:46
Número de pegadas duplas (veículos)	25	25	25	27	35	35	32	33
Número de tripulações	109	110	116	116	136	136	137	137
Tempo ocioso das tripulações (hh:mm)	57:47	51:29	59:44	49:23	67:57	60:41	78:26	67:10
Tempo de hora extra (hh:mm)	30:10	17:48	31:06	22:01	43:20	37:16	30:50	26:25
Número de pegadas dúplas (tripulações)	22	22	24	24	28	27	28	28
f_v	28133,0	28180,9	30525,4	30608,8	34552,2	34664,8	35027,3	35115,6
f_t	11345,1	11120,3	12032,2	11727,4	14212,4	14004,4	14196,6	13969,8
f_v+f_t	39478,1	39301,3	42557,6	42336,2	48764,6	48669,1	49223,9	49085,4

7 Conclusões

Este projeto de pesquisa apresentou um algoritmo híbrido, baseado nas metaheurísticas *Iterated Local Search* (ILS), *Variable Neighborhood Descent* (VND) e Busca Tabu (BT), aplicado ao Problema de Programação Integrada de Veículos e Tripulações.

Para testar o método proposto foram utilizados dados reais das empresas do Sistema de Transporte Público da cidade de Belo Horizonte (MG), que atuam na bacia do Barreiro.

Os resultados obtidos pelo método proposto são comparados com aqueles obtidos pela metodologia clássica, de resolver os problemas PPV e PPT de forma independente e seqüencial. Foram consideradas duas funções de avaliação: uma baseada em pesos empíricos e outra em custos. Os experimentos computacionais realizados mostraram que em ambos os casos, a metodologia integrada proposta é adequada, pois foi capaz de produzir resultados superiores àqueles alcançados pela clássica metodologia seqüencial.

Os resultados também mostram que a metodologia integrada proposta, em comparação com a metodologia seqüencial, pode produzir soluções piores para os veículos, mas melhores para as tripulações, gerando assim soluções melhores para o conjunto. Esse fato mostra claramente que é uma boa idéia explorar a integração desses problemas em vez de gerar a melhor solução para os veículos e depois gerar a melhor solução para as tripulações como faz a metodologia seqüencial.

Referências Bibliográficas

Atkinson, J B (1998) A greedy randomised search heuristic for time-constrained vehicle scheduling and the incorporation of a learning strategy. *The Journal of the Operational Research Society*, v. 49, p. 700-708.

Baita, F.; Pesenti, R.; Ukovich, W.; Favaretto, D. (2000) A comparison of different solution approaches to the vehicle scheduling problem in a practical case. *Computers and Operations Research*, v.27, p.1249-1269.

Ball, M. O; Bodin, L. D. & Dial, R. (1983) A matching based heuristic for scheduling mass transit crew and vehicles. *Transportation Science*, v. 17, p. 4-31.

Bassi, Hugo Vinícius; Marinho, Euler Horta; Souza, M. J. F.; Silva, Gustavo Peixoto. (2004). Heurísticas GRASP e Reconexão por Caminhos aplicadas ao problema de escalonamento de tripulações In: Simpósio Brasileiro de Pesquisa Operacional, 2007, Fortaleza. Anais do XXXIX SBPO. Rio de Janeiro: SOBRAPO, 2007. v.1. p.1081 – 1092.

Bicalho, M. S. S., Silva, G. P.; Souza, M. J. F. (2005) Otimização da operação dos veículos de empresas do transporte público de Belo Horizonte. *Revista da Pesquisa e Pós-Graduação*. Universidade Federal de Ouro Preto, p. 1-8.

Carpaneto, G.; Martello, S.; Toth, P. (1988) Algorithms and codes for the assignment problem. *Annals of Operations Research*, v.13, p. 193-223.

Freling, R.; Huisman, D. & Wagelmans, A. P. M (2001) Applyng an Integrated approach to vehicle and crew scheduling in practice. In: *Computer-Aided Scheduling of Public Transport*, S. Voss e J. R. Daduna (eds.), Springer-Verlag, Berlin, p. 73-90.

Freling, R.; Wagelmans, A. P. M. & Paixão, J. M. P. (1999) An overview of models and techniques for integrating vehicle and crew scheduling. In: *Computer-Aided Transit Scheduling*, N. H. M. Wilson (ed.), Springer-Verlag, Berlin, p.441-459.

Gavish, B.; Schweitzer, P.; Shlifer, E. (1978) Assigning buses to schedules in a metropolitan area. *Computers and Operations Research*, v.5, p. 129-138.

Glover, F.; Kochenberger, G. (2003) *Handbook of Metaheuristics*. Kluwer Academic Publishers.

Haase, K. & Friberg, C. (1999) An Exact Branch and Cut Algorithm for the Vehicle and Crew Scheduling Problem. In: *Computer-Aided Transit Scheduling*, N. H. M. Wilson (ed.), Springer-Verlag, Berlin, p. 63-80.

Hoffstadt, J. (1981) Computerized vehicle and driver scheduling for the Hamburger Hochbahn Aktiengesellschaft. In: *Computer Scheduling of Public Transport*, Wren A. (ed.), North-Holland, Amsterdam, p. 35-52.

Kirkman, F. (1968) Problems of innovation in the transport industry: a bus scheduling program. In: *Proceedings of PTRC Public Transport Analysis Seminar, Planning and Transport Research and Computation Co. Ltd.*, London, v.1, p. 1-15.

Kwan, R. K.; Rahin, M. A. (1995) Bus scheduling with trip co-ordenation and complex constraints. In: *Computer-Aided Transit Scheduling*, J. R. Daduna; I. M. Branco; J. M. P. Paixão (eds.), Springer-Verlag, Berlin, p. 91-101.

Kwan, R. K.; Rahin, M. A. (1999) Object oriented bus vehicle scheduling - the BOOST system. In: *Computer-Aided Transit Scheduling*, N. H. M. Wilson (ed.), Springer-Verlag, Berlin, p. 177-191.

LOURENÇO, H. R.; MARTIN, O.; STÜTZLE, T. (2003) Iterated Local Search. In: F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, Kluwer Academic Publishers, Boston, MA, USA..

Marinho, Euler Horta, Ochi, Luiz Satoru, Drummond, Lúcia M. A., Souza, M. J. F., Silva, Gustavo Peixoto. (2004). Busca Tabu aplicada ao Problema de Programação de Tripulações de Ônibus Urbano In: Simpósio Brasileiro de Pesquisa Operacional, 2004, São João Del Rey. Anais do XXXVI SBPO. Rio de Janeiro: SOBRAPO, 2004. v.1. p.1471 – 1482.

Martin-Löf, A. (1970) A branch-and-bound algorithm for determining the minimal fleet size of a transportation system. *Transportation Science*, v.4, p.159-163.

Mlavenovic, N. & Hansen, P. (1997). A Variable Neighborhood Search. *Computers and Operations Research*, v. 24, p. 1097-1100.

Psarras, J.; Stefanitsis, E.; Christodoulou N. (1997) Combination of local search and CLP in the vehicle-fleet scheduling problem. *European Journal of Operational Research*, v. 98, p. 512-521.

REIS, J. von A. (2007) Heurísticas baseadas em Busca em Vizinhança Variável para o Problema de Programação Integrada de Veículos e Tripulações no Transporte Coletivo Urbano por Ônibus, Dissertação de mestrado, Escola Politécnica da USP, São Paulo.

Saha, J. L. (1970) An algorithm for bus scheduling problems. *Operational Research Quarterly*, v.21, p. 463-474.

Silva, G. P. (2001) Uma metodologia baseada na técnica de geração de arcos para o problema de programação de veículos. Tese de doutorado. Escola Politécnica da USP, São Paulo.

SILVA, Gustavo Peixoto, SOUZA, M. J. F., GOMES, Helton Cristiano. (2005) Otimização integrada da tabela de horários com a programação de veículos no sistema de transporte público In: Congresso de Pesquisa e Ensino em Transportes, 2005, Recife. Panorama Nacional da Pesquisa em Transportes. Recife: ANPET, 2005. v.1. p.680 – 691.

SILVA, Gustavo Peixoto, REIS, Jorge Von Atzingen dos, SOUZA, M. J. F. (2006a) Metaheurísticas aplicadas ao Sistema de Transporte Público In: XIII Congreso Latino Iberoamericano de Investigación Operativa, 2006, Montevideu. Anais do XIII CLAIO. Montevideu: Universidad de la República, 2006. v.1. p.1 – 6.

SILVA, Gustavo Peixoto, REIS, Jorge Von Atzingen dos, SOUZA, M. J. F. (2006b) Resolução Integrada do Problema de Programação de Veículos e Tripulações no Sistema de Transporte Público In: Congresso Luso Brasileiro para o Planeamento Urbano, Regional, Integrado e Sustentável, 2006, Braga. Anais do 2º Congresso Luso Brasileiro para o Planeamento Urbano, Regional, Integrado e Sustentável. , 2006. v.1. p.121-132.

SIMÕES, E. M. L.; SOUZA, M. J. F.; SILVA, G. P. (2006) Aplicação da Metaheurística Iterated Local Search à Programação de Veículos no Sistema de Transporte Público. In: XXXVIII Simpósio Brasileiro de Pesquisa Operacional, 2006, Goiânia. Anais do XXXVIII SBPO. Rio de Janeiro : SOBRAPO, 2006. v. 1. p. 1277-1288.

SOARES, Gleidson Fonseca, SILVA, Gustavo Peixoto, MARINHO, Euler Horta, SOUZA, M. J. F. (2006) Alocação da mão-de-obra no Sistema de Transporte Público: uma visão multiobjetivo In: Congresso de Pesquisa e Ensino em Transportes, 2006, Brasília. Panorama Nacional da Pesquisa em Transportes. Rio de Janeiro: Associação Nacional de Pesquisa e Ensino em Transportes, 2006. v.1. p.693 - 704

SOUZA, M. J. F. (2008) Inteligência Computacional para otimização, Notas de aula, Departamento de Computação, Universidade Federal de Ouro Preto, disponível em http://www.decom.ufop.br/prof/marcone/InteligenciaComputacional/InteligenciaComputacional.pdf.

SOUZA, M. J. F.; RODRIGUES, M. M. S.; MAPA, S. M. S; SILVA, G. P. (2003a) Um estudo das heurísticas Simulated Annealing e VNS aplicadas ao problema de programação de tripulações. In: Anais do XXIII Encontro Nacional de Engenharia de Produção, Ouro Preto, 8 p., outubro de 2003.

SOUZA, M. J. F.; SILVA, G. P.; TOFFOLO, T. A. M. (2005) Resolução do Problema de Rodízio de Tripulações de Ônibus Urbano via Simulated Annealing e Iterated Local Search. In XIX Anais do Congresso de Pesquisa e Ensino em Transportes, Recife, v. 1, p. 657-668.

Souza, M. J. F; Cardoso, L. X. T. & Silva, G. P. (2003b) Programação de Tripulações de Ônibus Urbano: uma abordagem heurística, XXXV Simpósio Brasileiro de Pesquisa Operacional, Natal: SOBRAPO, 2003, p. 1285-1294.

Souza, M. J. F; Xavier, L. X. T.; Silva, G. P.; Mapa, S. M. S.; Rodrigues, M. M. S. (2004a) Metaheurísticas Aplicadas ao Problema de Programação de Tripulações no Sistema de Transporte Público. *Tendências em Matemática Aplicada e Computacional* (TEMA), São José do Rio Preto, v. 5, n. 2, p.357-368.

Souza, M. J. F; Cardoso, Silva, G. P. & Mapa, S. M. S. (2004b) Métodos de Pesquisa em Vizinhança Variável Aplicados à Resolução do Problema de Programação Diária de Tripulações de Ônibus Urbano, XVIII Congresso de Pesquisa e Ensino em Transportes, Panorama Nacional de Pesquisa em Transportes, v. 2, p. 1492-1502.

SOUZA, M. J. F., SILVA, Gustavo Peixoto, SIMÕES, Emiliana Mara Lopes. (2006) Programação de Veículos de Ônibus Urbano: uma abordagem heurística In: Congresso de Pesquisa e Ensino em

Transportes, 2006, Brasília. Panorama Nacional da Pesquisa em Transportes. Rio de Janeiro: Associação Nacional de Pesquisa e Ensino em Transportes, 2006. v.1. p.705-716.

TOFFOLO, T. A.; SOUZA, M. J. F.; PEIXOTO, G. P. (2005) Algoritmos Simulated Annealing e Iterated Local Search aplicados à resolução do Problema de Rodízio de Tripulações de Ônibus Urbano. In VIII Simpósio de Pesquisa Operacional e Logística da Marinha, Rio de Janeiro, Anais do VIII SPOLM, 12 p.

Tosini, E. & Vercellis, C. (1988) An interactive system for extra-urban vehicle and crew scheduling problems. In: *Computer-Aided Transit Scheduling*, J. R. Daduna e A. Wren (eds.), Springer-Verlag, Berlin, p. 41-53.

Wren, A.; Chamberlain, M. P. (1988) The development of Micro-BUSMAN: scheduling on micro-computers. In: *Computer-Aided Transit Scheduling*, J. R. Daduna; A. Wren (eds.), Springer-Verlag, Berlin, p.160-174.

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