COPPE/UFRJ

META-HEURÍSTICAS PARA O PROBLEMA DE PROGRAMAÇÃO DE TRIPULAÇÕES

Tiago Luiz Gonçalves

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: Marcia Helena Costa Fampa
Luiz Satoru Ochi

Rio de Janeiro
Março de 2010

META-HEURÍSTICAS PARA O PROBLEMA DE PROGRAMAÇÃO DE TRIPULAÇÕES

Tiago Luiz Gonçalves

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Profa. Marcia Helena Costa Fampa, D.Sc.

Prof. Luiz Satoru Ochi, D.Sc.

Prof. André Gustavo dos Santos, D.Sc.

Prof. Nelson Maculan Filho, D.Sc

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2010

Gonçalves, Tiago Luiz

Meta-heurísticas para o Problema de Programação de Tripulações/Tiago Luiz Gonçalves. – Rio de Janeiro: UFRJ/COPPE, 2010.

XII, 86 p.: il.; 29, 7cm.

Orientadores: Marcia Helena Costa Fampa

Luiz Satoru Ochi

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2010.

Referências Bibliográficas: p. 75 – 83.

1. Programação de Tripulações. 2. Otimização. 3. Inteligencia Artificial. 4. Meta-heurística. I. Fampa, Marcia Helena Costa *et al.*. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Dedico este trabalho a Deus.

Agradecimentos

Agradeço...

A Deus, por tudo, saúde e proteção diante de todas adversidades enfrentadas.

Aos meus pais Sebastião Luiz Gonçalves e Luzia Rosa Gonçalves, pelo apoio incondicional.

À toda minha família, em especial ao Geraldo, pela amizade e companheirismo.

A todos amigos, que não pouparam esforços em me ajudar nesta longa caminhada, em especial às minhas amigas Uliana, Izabela, e suas famílias, pelas constantes orações.

Aos amigos da academia, em especial, Jurair, Francisco, Jesus Ossian e Michael.

À orientadora professora Marcia Fampa por sua extraordinária sabedoria e paciência mesmo diante da distância e outras adversidades.

Ao co-orientar professor Satoru, por acreditar no meu trabalho, e por todas contribuições.

Ao professor André Gustavo, pelo seu constante incentivo e apoio, por sua participação na banca e suas contribuições.

Ao professor Nelson Maculan, membro da banca, pela participação e contribuições.

Aos professores da UFRJ, pelo conhecimento compartilhado ao longo das disciplinas.

Ao professor Adilson, por acreditar em mim, e por incentivar-me.

Ao professor Marcone Jamilson e seus alunos, por terem me fornecido dados para o problema tratado neste trabalho.

A todos os demais, que de alguma forma contribuíram para o sucesso deste trabalho, o meu humildemente e sincero, muito obrigado.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

META-HEURÍSTICAS PARA O PROBLEMA DE PROGRAMAÇÃO DE TRIPULAÇÕES

Tiago Luiz Gonçalves

Março/2010

Orientadores: Marcia Helena Costa Fampa
Luiz Satoru Ochi

Programa: Engenharia de Sistemas e Computação

Aborda-se aqui o problema de programação de tripulações de transporte em massa (PPT) também conhecido como *Crew Scheduling*. O problema consiste em gerar um conjunto de escalas de trabalho, atribuindo tarefas a tripulações, com menor custo possível, e ao mesmo tempo satisfazendo restrições tais como leis trabalhistas, legislações federais, sindicais e normas internas da empresa. O presente trabalho aborda o problema utilizando em sua solução as meta-heurísticas Busca Tabu e *Iterated Local Search*. Realiza-se um estudo comparativo entre os métodos aqui aplicados e uma metodologia exata de programação matemática, onde o PPT é modelado como um problema de particionamento de conjuntos. Através deste é apresentado de forma categórica baseando-se em resultados concretos a grande eficiência das meta-heurísticas em fornecer rapidamente soluções de alta qualidade para problemas reais tais como o PPT. Dentre os métodos aqui aplicados, aquele que baseia-se na meta-heurística *Iterated Local Search* é o que mais se destaca, fornecendo os melhores resultados. A solução final deste problema é dada por um conjunto de jornadas diárias de trabalho. As instâncias de testes foram geradas baseando-se estritamente em dados reais de uma empresa do setor de transporte público coletivo que opera no município de Belo Horizonte/MG.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

META-HEURISTICS FOR THE CREW SCHEDULING PROBLEM

Tiago Luiz Gonçalves

March/2010

Advisors: Marcia Helena Costa Fampa
Luiz Satoru Ochi

Department: Systems Engineering and Computer Science

This work approaches Crew Scheduling Problem (CSP). Such problem consists in generating a set of work scales, assigning tasks for the crews, with smallest possible cost, and at the same time satisfying a set of constraints as, working laws, federal legislations, syndical and internal norms of the company. This work approaches CSP using Iterated Local Search (ILS) and Tabu Search (TS) meta-heuristics to solve it. A comparative study between these and an exact method was done, where CSP is modeled as a Set Partitioning Problem. The great efficiency of meta-heuristics quickly to provide high quality to real problems like this is presented in categorical form based on results. The results shown that quality of solutions produced by ILS were better. The final solution of this problem is a set of crew duties. All instances were generated with real data of a public transport company operating in Belo Horizonte city, Brazil.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Quando se trata do planejamento de companhias de transporte de massa, uma série de decisões devem ser tomadas a todo momento: programação das rotas; alocação, escalonamento, manutenção e renovação da frota de veículos; contratação, treinamento e programação da tripulação; replanejamento diante de situações adversas, atrasos, condições climáticas, dinâmica da demanda; além de outros tantos fatores, os quais deve-se levar em conta durante todo o processo.

Tais tomadas de decisão variam sob os mais diversos aspectos: desde um planejamento em longo prazo, como estabelecimento (planejamento) de rotas, ou em curto prazo, como escala semanal de trabalho de um membro da tripulação; desde decisões pontuais, como o treinamento de um tripulante para ocupação de uma nova função, como decisões mais amplas, como gerar escalas mensais de trabalho para todos os funcionários, incluindo dias de folga, férias, bem como preferências pessoais; desde decisões que possuem um prazo maior para serem cuidadosamente planejadas, como aquisição de novos veículos, há ainda aquelas que devem ser tomadas em caráter emergencial, como replanejamento e substituição de veículos e/ou tripulantes, devido a possíveis adversidades, como problemas no veículo ou até mesmo problemas de saúde de um dos membros da tripulação.

Com o objetivo de se sobressaírem em um mercado que encontra-se atualmente tão competitivo, grande parte das empresas já contam com algum tipo de sistema que as apoiem em suas tomadas de decisão, para a otimização do aproveitamento, aquisição, ou dispensa dos recursos envolvidos em todo processo [1].

Em algumas regiões, setores e localidades, ainda é possível encontrar certas em-

presas que detém o monopólio sob algumas rotas, devendo "apenas" planejar a utilização de seus recursos para cobrir tais rotas, o que na verdade já não se trata de uma tarefa trivial. Todavia, na maioria dos casos, mais de uma empresa atende a uma mesma região, competindo entre si na constante busca pela captação do maior número de clientes possíveis (passageiros).

Alguns usuários preferem empresas que prestem-se com maior regularidade e pontualidade em seus horários de viagem, outros por sua vez já optam por maior conforto e outros benefícios concedidos, há ainda aqueles que optam pela menor tarifa. Existem empresas que buscam atender a todas essas classes, logo oferecem seus serviços de transporte classificados sob diferentes categorias, a exemplo: econômica, executiva e primeira classe, no caso de aviões; regular, executivo, leito, dentre outros, no caso de ônibus. Para todos os casos, um minucioso planejamento deve ser realizado para se conquistar uma fatia no mercado, e ainda para manter ou até mesmo ampliar essa fatia.

Outro produto do mau planejamento e crescente demanda, que deve ser levado em conta é o surgimento de meios de transportes "alternativos" e até mesmo clandestinos, esses, em sua maioria não oferecem o mínimo de conforto, e além disso podem comprometer a segurança dos usuários. Esses encontram-se em sua maioria presentes dentro das cidades e algumas vezes nos transportes rodoviários. Demanda e busca por tarifas mais baixas são algumas das motivações que conduzem alguns usuários a este tipo de transporte. Em detrimento desse fator, tem-se perdido demanda por parte do sistema regular, seja para o transporte "alternativo" ou clandestino os quais captam principalmente passageiros de renda mais baixa, seja para o uso do automóvel próprio, que por sua vez capta passageiros de renda mais elevada.

Ao que consta a democratização do acesso ao sistema de transporte público, é outro fator que está diretamente vinculado à garantia de tarifas que condizem com a realidade econômica dos usuários, e esta depende tanto do aumento da eficiência, bem como da redução dos custos operacionais.

Tendo em vista a discussão supra realizada, observa-se facilmente o quanto o planejamento da rede de transporte gera impactos sob as mais diversas direções, congestionamentos crônicos, queda da mobilidade e da acessibilidade, degradação das condições ambientais, e ainda os altos índices de acidentes no trânsito.

Há ainda que se considerar o fato da existência de complicadores ligados as diferentes regras impostas por diferentes países e até mesmo diferentes empresas. Daí surge a necessidades de trabalhos como o presente, dedicados ao desenvolvimento de soluções que considerem o cenário real do Brasil.

A abordagem direta, ou seja, contemplar diretamente todos os problemas intrínsecos ao sistema de transporte pode ser uma tarefa computacionalmente intratável, por conseguinte este trabalho concentra-se então exclusivamente ao tratamento do Problema de Programação de Tripulações (PPT). Esta escolha se deve principalmente a que, para as companhias de transporte, a parte operacional com respeito às tripulações representa para si um dos fatores mais onerosos [2]. No setor de transporte aéreo por exemplo, o gasto com a tripulação representa aproximadamente 20% dos gastos operacionais para as empresas [3]. Portanto qualquer redução nesta componente, ainda que esta, a princípio, seja mínima, pode conduzir a um ganho expressivo no custo total do processo, e até mesmo refletir em possibilidades de redução no valor da tarifa a ser cobrada ao usuário final (passageiro), elevando assim o nível de competitividade da empresa.

O PPT consiste em gerar um conjunto de jornadas de trabalho que contenha todas as viagens a serem realizadas por uma empresa de transporte. O principal objetivo ao se resolver este problema é reduzir custos operacionais, como por exemplo, determinar o número mínimo de jornadas de trabalho de forma a atender todas as viagens. Entretanto essa não é uma tarefa trivial já que a solução deve respeitar todos os requisitos legais, o que gera um conjunto considerável de restrições [4]. Estudos recentes podem ser encontrados em [5].

O PPT é um problema que vem sendo estudado há vários anos, tendo seu início em meados da década de 60, sendo um dos primórdios [6]. Entretanto tais sistemas consistiam apenas na automação do trabalho, o qual outrora era manualmente realizado, assim não detectando possibilidades de otimização sob qualquer aspecto [7]. Estes tratavam-se de sistemas que empregavam heurísticas baseadas no trabalho manual. Já nos anos 70 [8] começam a surgir a introdução de métodos de otimização neste tipo de sistema [9].

O PPT é um problema de otimização, o qual pertence a classe $\mathcal{NP}$-*Difícil* [10], por conseguinte sabe-se que um algoritmo capaz de resolvê-lo em tempo polinomial

não pode ser encontrado, a menos que $\mathcal{P} = \mathcal{NP}$ [11]. Além disso em situações reais o PPT apresenta uma grande complexidade, de cunho prático, para sua resolução devido a presença de uma grande quantidade de restrições operacionais envolvidas no processo. Diante deste fato e dos vários objetivos encontrados no problema não é possível ater-se às formas tradicionais e exatas de resolução, pois as mesmas exigem um alto poder computacional para explorar o espaço de soluções durante a busca por uma solução ótima.

Portanto, com o objetivo de reduzir o tempo computacional na resolução de problemas como o PPT, tem se tornado muito comum o uso de meta-heurísticas. Técnicas como essas vêm expandindo os horizontes no que concerne à forma como problemas de alta complexidade têm sido abordados, com destaque para aqueles de natureza combinatória. Tais técnicas propõem a busca inteligente por uma boa solução para o problema sem explorar todo o espaço de busca, diminuindo expressivamente assim, o custo computacional. Vale ressaltar que estas técnicas não garantem a melhor das soluções para o problema, todavia apresentam resultados aceitáveis e de boa qualidade.

As técnicas meta-heurísticas se definem por uma heurística que possui uma ou mais heurísticas subordinadas com o objetivo de otimizar a solução, tentando escapar de mínimos locais. Essas técnicas exigem um projeto específico para cada tipo de problema. Contudo, vários problemas comungam modelagens compatíveis e além disso a técnica possui uma estrutura flexível podendo assim ser adaptada com relativa facilidade. Como exemplo de algumas meta-heurísticas é possível citar Algoritmos Genéticos (AG) [12], Busca Tabu (BT) [13], *Greedy Randomized Adaptive Search Procedure* (GRASP) [14], *Simullated Anealling* (SA) [15], entre tantas outras.

Tendo em vista fatos como os supra discutidos, o desenvolvimento de metodologias de solução eficientes é, portanto, de grande e "vital" importância. Neste trabalho apresenta-se duas abordagens distintas, bem como independentes. Ambas sob o ponto de vista de meta-heurísticas baseadas em busca local. A primeira baseia-se em Busca Tabu, e a segunda por sua vez em *Iterated Local Search* [16]. Embora ambas se foquem em busca local, estas possuem estratégias expressivamente distintas. Outra proposição é feita, que por sua vez combina ILS e uma estrutura

de memória. Ao final promove-se um estudo comparativo entre todas proposições.

1.1 Organização do texto

Os demais Capítulos encontram-se organizados da seguinte forma: No Capítulo 2 apresenta-se uma revisão bibliográfica de trabalhos encontrados na literatura com respeito ao problema abordado e as metodologias aqui aplicadas, estabelecendo assim pilares essenciais para o desenvolvimento deste. No Capítulo 3 descreve-se o Problema de Programação de Tripulações e apresenta-se uma aplicação sob um cenário real, de uma empresa que atua no ramo do transporte urbano. Por meio do Capítulo 4 apresentam-se as metodologias aplicadas utilizando-se uma abordagem direcionada ao estudo de caso que aqui se faz. Através do Capítulo 5 apresentam-se todos os resultados produzidos por meio dos experimentos realizados. E por fim tratam-se as conclusões e considerações finais por meio do Capítulo 6, onde também apresentam-se proposições, as quais vislumbram-se desenvolver em trabalhos futuros.

Capítulo 2

Estudos Precedentes

Conforme introduzido ao longo do Capítulo 1, os sistemas de redes de transporte de massa apresentam um problema de grande complexidade em seu planejamento [17]. Esse problema geralmente é abordado pelos mais diversos trabalhos presentes na literatura sub-dividido sob a forma de problemas menores: Programação de Horários (*Timetabling*), Programação de Veículos (*Vehicle Scheduling*), Programação de Tripulações (*Crew Scheduling*), e por fim, Rotação (*Rostering ou Crew Rostering*) [18, 19]. A Figura 2.1 a qual baseia-se em [18, 17] ilustra tal decomposição.

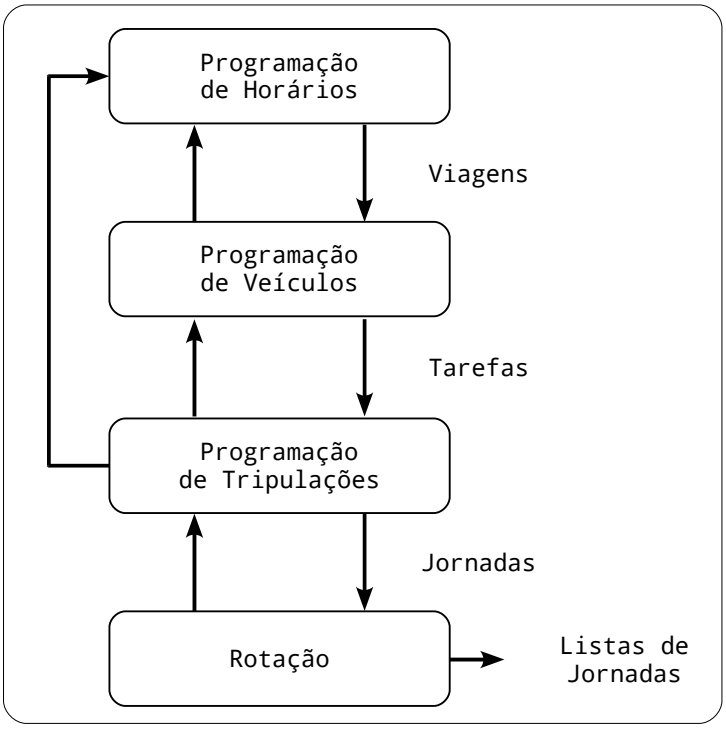


Figura 2.1: Planejamento de rede de transportes de massa

Vale ainda enfatizar que este é também o esquema de decomposição usualmente empregado pelas mais diversas investigações presentes na literatura ([20, 21]), bem como aquelas que consideram o cenário brasileiro ([22]).

Baseando-se no arquétipo ilustrado através da Figura 2.1, de uma forma geral este planejamento é inicialmente composto por um conjunto de linhas, estas por sua vez encontram-se devidamente identificadas, onde cada uma corresponde à viagem entre duas localidades sob um percurso próprio. Para cada uma destas linhas, a respectiva frequência é então determinada baseando-se essencialmente em aspectos tais como: demanda, disponibilidade de infra-estrutura, e serviços requeridos pelos usuários [23]. Passa-se a descrever sucintamente cada umas das etapas do planejamento do sistema de transporte.

Programação de horários A partir destas linhas, a programação de horários é então constituída resultando em viagens, as quais correspondem a uma localidade de início e uma outra de fim, bem como horário de início e horário de fim.

Programação de veículos A programação de veículos (PPV) por sua vez atribui veículos a tais viagens. Nesta fase todas as viagens são agrupadas em blocos (ver exemplo de um bloco ilustrado por meio da Figura 2.2), onde cada um dos quais representa uma sequência de viagens que um veículo deve realizar ao longo de um dia começando e finalizando em sua respectiva garagem (mais detalhes ver [23, 24]).

Particionamento dos blocos de viagens A partir das viagens contidas em cada um dos blocos (ver Figura 2.2) são constituídas as chamadas tarefas[1][27, 28]. Esta representa uma etapa intermediária entre o PPT e PPV. Por questões vinculadas à limitações físico-temporais deve-se particionar cada um dos blocos, pois nestes podem haver subsequencias de viagens onde não é possível trocar de tripulação, portanto é possível afirmar que tais viagens pertencerão a uma mesma tripulação. O tempo mínimo para haver troca de tripulação é regido por um elemento chamado *oportunidade de troca*[2]($\mathcal{OT}$), que é um valor fixo que representa o intervalo de duração mínima para que ocorra uma troca de tripulações. Tal fase é ilustrada

[1] *pieces of work* [25, 17, 26]

[2] *pontos de troca* [1], *relief opportunities* [9, 25, 17, 29, 26, 5], *relief points* [30, 28, 31].

por meio da Figura 2.2 [32], em seguida o procedimento é plenamente descrito na seção 2.1.

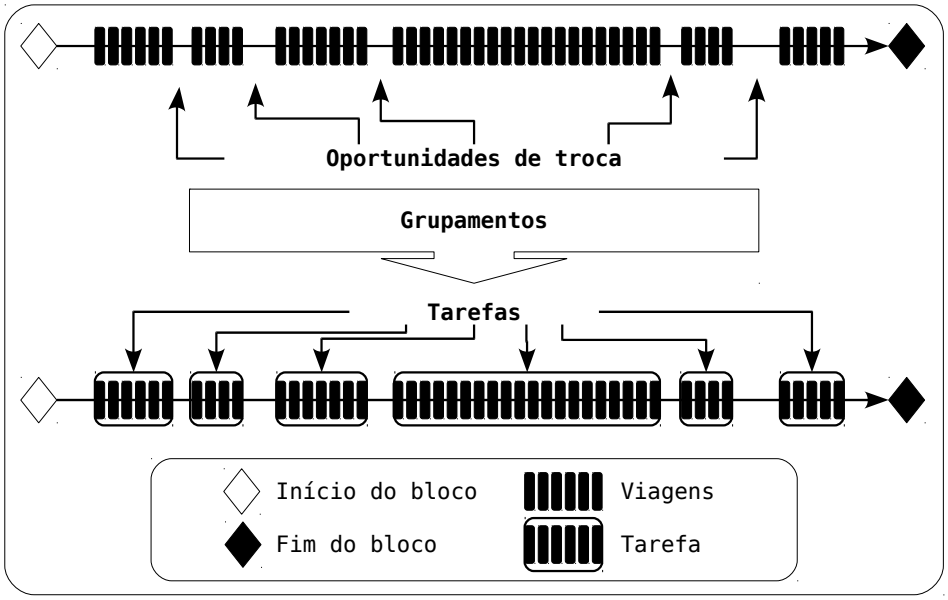


Figura 2.2: Bloco de viagens de um veículo

Programação de tripulações A partir das tarefas obtidas por meio da fase anterior são então construídas as jornadas[3] (*duties*)(ver Figura 2.5), que a posteriori são atribuídas às tripulações, as quais irão executá-las ao longo de um dia de trabalho.

Rotação De posse das jornadas construídas na fase anterior, finalmente dá-se a etapa, ou problema de rotação, o qual consiste em se distribuir as jornadas de trabalho aos tripulantes compondo assim as escalas de trabalho, sob um horizonte dado por um período de tempo, geralmente semanal, ou mensal. Nesse passo consideram-se os dias de folgas, eventuais feriados, recessos, férias das tripulações e algumas outras restrições legais, as quais também deve-se levar em conta [19, 7, 35]. Essa etapa dá-se essencialmente de tal forma que haja algum tripulante associado a cada uma das jornadas que devem ser cumpridas em cada um dos dias [1].

No transporte coletivo rodoviário, conforme ilustrado por meio da Figura 2.1 usualmente a programação de tripulações é realizada após a programação dos veículos [36, 37]. A próxima seção trata então a etapa de particionamento dos blocos de viagens, que é a fase intermediária entre estas duas etapas, e é elementar para se compor

[3]turnos [33, 7], ou *shift* [28, 25, 34, 26], ou *duty* [17, 29]

a entrada de dados para o PPT, e assim dar início ao processo de sua resolução.

2.1 Geração das tarefas

Conforme anteriormente mencionado, a saída, ou solução do PPV é dada por um conjunto de viagens as quais encontram-se agrupadas em blocos. Cada um destes blocos representa o itinerário de um veículo partindo e retornando à sua garagem.

Convenciona-se que: hi(.) e hf(.) representam respectivamente o horário de início e de fim de uma atividade seja ela: viagem, tarefa ou até mesmo jornada.

Denota-se por meio de $\mathcal{B} = \{b_1, \ldots, b_n\}$ o conjunto formado por todos estes blocos, onde cada elemento $b_k = \{v_{k1}, \ldots, v_{km}\}$ é formado por um conjunto de viagens. As propriedades de uma viagem encontram-se dispostas na Tabela 2.1.

viagem	
horário inicial	horário final
ponto inicial	ponto final
linha	veículo

Tabela 2.1: Atributos de uma viagem

Naturalmente as viagens de cada um dos blocos encontram-se dispostas sob ordenação temporal, com respeito ao horário em que devem ser realizadas.

$$\text{hi}(v_{k1}) < \text{hf}(v_{k1}) \leq \text{hi}(v_{k2}) < \ldots < \text{hf}(v_{km-1}) \leq \text{hi}(v_{km}) < \text{hf}(v_{km}) \tag{2.1}$$

A partir destas informações deseja-se compor então, a entrada de dados para o PPT. O processo de geração das tarefas encontra-se plenamente ilustrado por meio do Algoritmo 1.

Algoritmo 1 Geração das tarefas

requer solução $\mathcal{B} = \{b_1, \ldots, b_n\}$ do PPV.

assegura entrada de dados para o PPT.

1: $i \leftarrow 0$ {contador de tarefas}
2: **para** $b_k \in \mathcal{B}, k = 1, 2, \ldots, |\mathcal{B}|$ **faça**
3: $\quad i \leftarrow i + 1$
4: $\quad t_i \leftarrow \emptyset$
5: $\quad t_i \leftarrow \{v_{k1}\}$
6: $\quad$ **para** $v_{kj} \in b_k, j = 2, 3, \ldots, |b_k|$ **faça**
7: $\quad\quad$ **se** $\mathrm{hi}(v_{kj}) - \mathrm{hf}(t_i) \geq \mathcal{OT}$ **então**
8: $\quad\quad\quad i \leftarrow i + 1$
9: $\quad\quad\quad t_i \leftarrow \emptyset$ {uma nova tarefa vazia}
10: $\quad\quad$ **fim se**
11: $\quad\quad t_i \leftarrow t_i \cup \{v_{kj}\}$
12: $\quad$ **fim para**
13: **fim para**

Com respeito ao Algoritmo 1: O processo de particionamento inicia com um índice $i = 0$ afim de gerenciar o número de tarefas que serão geradas até o final do processo. Em uma primeira iteração toma-se o índice $k = 1$, que significa que está a se analisar o primeiro bloco de viagens, $i = 1$ que neste momento tem uma única tarefa, que por sua vez encontra-se vazia $t_i = \emptyset$ (linha 4). A primeira viagem v_{k1} do bloco b_k é atribuída à tarefa t_i. Toma-se um índice $j = 2$, que representa a segunda viagem v_{kj} do veículo b_k. Se o intervalo entre a tarefa t_i e a viagem v_{kj} representa uma oportunidade de troca de tripulação, então o índice i é incrementado fazendo $i \leftarrow i + 1$, isso significa que uma nova tarefa $t_i = \emptyset$ é iniciada, e a viagem em questão é a ela diretamente atribuída; caso contrário, o índice i permanece inalterado e a viagem v_{kj} é atribuída à tarefa t_i. O procedimento realizado entres as linhas (6)–(12) persiste até que $j = |b_k|$, ou seja, todas as viagens do bloco sejam contempladas (linha 6). O processo como um todo persiste até que $k = |\mathcal{B}|$, ou seja, todos os blocos sejam plenamente contemplados (linha 2). Vale acrescentar que a cada viagem atribuída a uma tarefa (linha 11), a tarefa em construção herda as propriedades daquela viagem, conforme ilustrado na Figura 2.3.

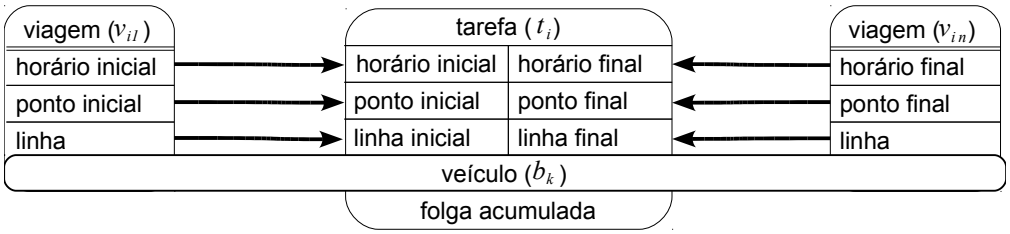


Figura 2.3: Propriedades herdadas das viagens

Onde a tarefa t_i herda de v_{i1}, seu horário de início, ponto inicial e linha; herda da viagem v_{in} (última) horário final, ponto final e linha. Além disso o intervalo entre todas as viagens em t_i são somados,

$$\sum_{j=1}^{|t_i|-1} (\mathrm{hi}(v_{j+1}) - \mathrm{hf}(v_j)) ,$$

e o valor obtido é então chamado de *folga acumulada*, o qual torna-se também uma das propriedades da tarefa (ver Figura 2.3).

Se a cardinalidade de uma tarefa é $|t_i| = 1$, obviamente todas as propriedades são idênticas às da única viagem que é v_{i1}, sendo que no caso da propriedade linha, esta torna-se para t_i, ao mesmo tempo linha inicial e final.

Através do Algoritmo 1 e das demais informações expostas ao longo desta seção nota-se que dadas duas quaisquer tarefas t_i e t_{i+1} consecutivas, desde que executadas por um mesmo veículo, não há sobreposição entre elas, ou seja, neste caso assegura-se:

$$\mathrm{hf}(t_i) < \mathrm{hi}(t_{i+1}), i = 1, 2, \ldots, n-1,$$

onde n neste caso representa o número de tarefas geradas a partir do bloco de viagens de um único veículo arbitrário.

Nota-se também que a ordenação temporal das tarefas de um mesmo veículo é assegurada. Além disso as tarefas encontram-se concentradas com respeito ao atributo que representa o veículo.

Vale enfatizar que, deste passo por diante, a *tarefa* torna-se o principal elemento de trabalho na resolução do PPT, assunto abordado ao longo da próxima seção.

2.2 Programação de Tripulações

Considera-se que todas as rotas já foram planejadas, bem como todas as viagens que compõem cada uma dessas rotas, seus pontos de repouso e troca de tripulação, e ainda seus horários e frequências. Considera-se também que os veículos já foram associados a cada uma das viagens, de forma que já se tem conhecimento de quais os funcionários são habilitados para cada viagem, de acordo com sua habilitação e experiência no tipo de veículo a ser utilizado. Por final considera-se também que, baseando-se na lista de viagens, todas as tarefas foram geradas (ver seção 2.1). Agora de posse de todos esses dados, o problema consiste então em se determinar a escala de trabalho de cada tripulante, definindo sua jornada de trabalho, ou seja, a questão é nesse momento, que tarefas ele deve cumprir ao longo de um determinado dia de trabalho [1].

Este é um dos mais importantes problemas do setor, isso independentemente da especifidade do meio de transporte, do qual se trata. Programação de Tripulações [38, 39, 35, 40], Escalonamento de Tripulações [7, 41] e Alocação de Tripulações [1, 42], estes são alguns dos termos mais usuais encontrados nos trabalhos presentes na literatura para referir-se a este problema.

O problema consiste em distribuir tarefas aos tripulantes (pilotos, comissários de bordo, maquinistas, motoristas, cobradores, etc.) durante vários trechos de viagens a serem cobertos por empresas aéreas, ferroviárias (metroviário) ou rodoviárias gerando assim jornadas individuais passíveis de serem atribuídas à cada uma das tripulações disponíveis. Tudo isso deve ser feito com o menor custo possível. Em contrapartida, esse custo deve ser minimizado sem violar quaisquer das restrições do problema, como por exemplo, normas sindicais e leis trabalhistas, as quais regulam essencialmente o relacionamento entre as companhias e suas tripulações (funcionários). Tais restrições influenciam diretamente no que concerne ao tempo a ser despendido para se resolver o problema.

A solução do PPT é induzida por um conjunto

$$s : s = \{\mathcal{J}_1, \ldots, \mathcal{J}_n\}$$

de jornadas (ver Figura 2.4). Onde, cada jornada é representada por um conjunto

$$\mathcal{J} : \mathcal{J} = (t_1, \ldots, t_n)$$

de tarefas. Assume-se para este contexto que cada jornada está implicitamente vinculada a uma tripulação

$$\mathcal{J} : \mathcal{J} \to \mathcal{C},$$

logo neste trabalho, hora pode-se falar em jornadas, hora pode-se falar em tripulações referindo-se à mesma entidade. A Figura 2.4 ilustra um exemplo de escala de tripulações.

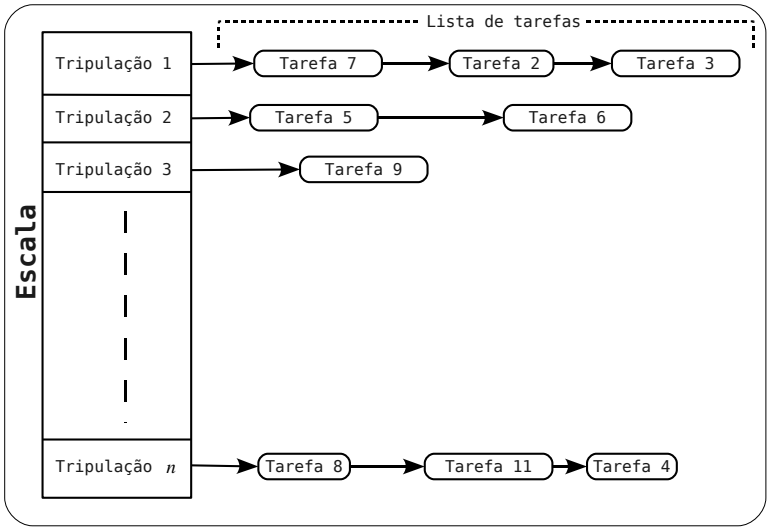


Figura 2.4: Programação de tripulações.

O processo de formação das jornadas de trabalho sujeita-se a um conjunto de regras particulares a uma organização. Tais regras são comumente fruto da composição das regras nacionais e locais [43], podendo ser obrigatórias ou facultativas.

Tipicamente há restrições com respeito ao tempo máximo de trabalho, período de tempo que pode ser trabalhado sem intervalo para descanso ou refeição, ao longo da realização da jornada (duração entre o início e o fim da jornada), etc. Algumas destas restrições encontram-se ilustradas na Figura 2.5.

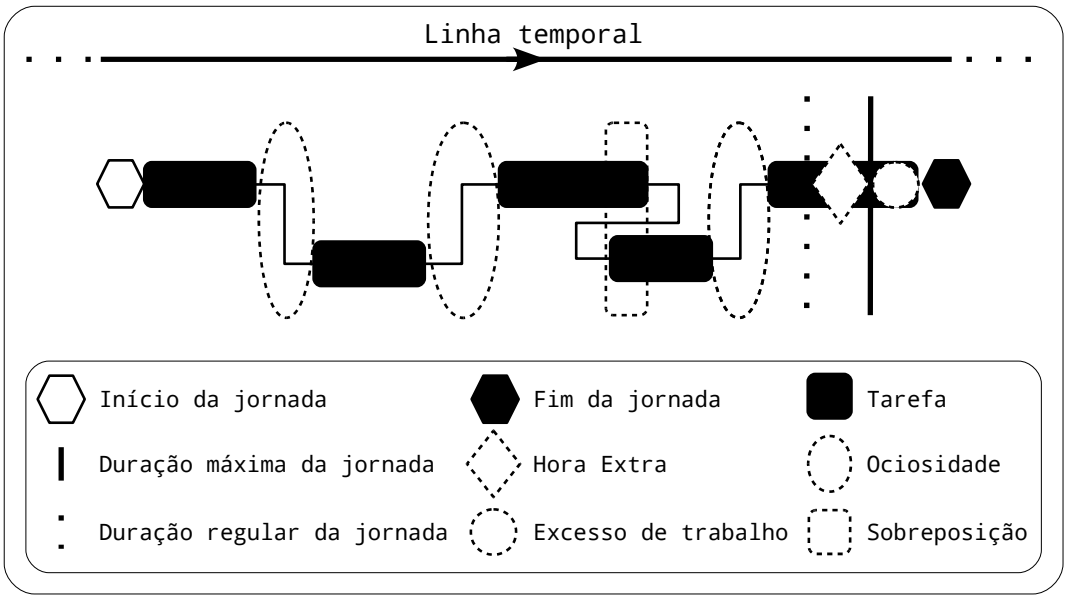


Figura 2.5: Jornada de trabalho e suas restrições básicas

Todos os detalhes envolvidos neste estudo de caso são devidamente apresentados no Capítulo 3.

Com o proposito de solucionar o PPT, várias linhas de investigação foram segmentadas, sendo as principais técnicas envolvendo programação matemática: *Branch-and-Bound* [44, 45], *Branch-and-Price* [30, 46], *Branch-and-Cut* [47]. Em [27] utiliza-se relaxação lagrangeana e otimização por sub-gradiente. Geralmente a dimensão dos problemas encontrados na prática são bem maiores do que a que estes trabalhos conseguiram solucionar.

Uma variedade de meta-heurísticas também foram propostas *Simulated Annealing* [33], *Variable Neighborhood Search* [48], *Grasp* [22]. Algoritmos Genéticos [49], Colonia de Formigas [25]. Alguns como [50] propõem metodologias híbridas que combinam algumas destas meta-heurísticas entre si. Neste último tipo de abordagem, o elevado número de parâmetros a serem harmonizados é um dos principais gargalos.

Marinho [35] apresenta uma abordagem baseada na meta-heurística Busca Tabu (BT), e variações denominadas Busca Tabu com Primeira Melhora (BT-FI-IC) e Busca Tabu com Melhor Vizinho (BT-BI-IC), Busca Tabu com primeira melhora e Relaxação Adaptativa (BTAR-FI-IC), Busca Tabu com primeira melhora e reconexão por caminhos (BTPR-FI-IC). O autor testa os métodos desenvolvidos e os compara com o método de VNS-RTL proposto em [48], que, até aquele momento se-

gundo o autor apresentava os melhores resultados para as instâncias fornecidas pela empresa de transporte. O autor também compara seus resultados com os da empresa. Dos métodos sugeridos, o método BT-FI-IC apresentou melhora na solução e melhora no tempo computacional. Todas as heurísticas propostas produziram redução na quantidade de horas extras, que é um dos componentes que representam maior custo para a empresa. O autor conclui que a versão BTAR-FI-IC apresentou soluções com uma média de desvio de 0,26% das melhores soluções conhecidas considerando todas as instâncias. Foi considerada uma única instância real e dez artificiais as quais foram geradas a partir da real. Quando classificadas entre reais e artificiais, a média do desvio passa a ser de 1,93% e 0,11% respectivamente.

Há ainda aqueles trabalhos como [38, 51, 32] que propõe métodos híbridos onde técnicas de programação matemática e meta-heurísticas são combinadas.

Em [1] parte-se da típica premissa onde o PPT é decomposto em um problema mestre e um subproblema, responsáveis respectivamente por selecionar o melhor conjunto de jornadas, e gerar novas jornadas. No entanto o foco principal daquele trabalho foi a solução do subproblema, pois embora ambos sejam da classe $\mathcal{NP}$-*Difícil*, o problema mestre pode ser rapidamente resolvido pelos atuais pacotes de otimização, desde que o número de jornadas consideradas não seja grande, o que normalmente acontece quando se usa geração de colunas. Neste trabalho são usadas uma heurística gulosa e as meta-heurísticas *Grasp* e Algoritmo Genético para acelerar a geração de colunas, assegurando-se a otimalidade da solução combinando essas heurísticas com programação linear inteira. O autor conclui que sua proposta é capaz de resolver de forma exata problemas de programação de tripulação mais rapidamente que valendo-se do que métodos "puramente" exatos.

Vários sistemas (Carmen [52],[53]; CREW-OPT [54], Rucus [55], HOT [56], HASTUS [57], entre outros) foram propostos objetivando apoiar as decisões dos responsáveis por planejar as atividades das tripulações. Grande parte destes sistemas tornaram-se comerciais. Em sua maioria são utilizados nos países da Europa [7].

Mais detalhes com respeito ao PPT, e métodos de solução podem ser encontrados em trabalhos tais como [5] e [58].

2.3 Formulações para o PPT

O texto exposto nesta seção baseia-se em [1]. Supondo agora, que o conjunto viável de todas as possíveis jornadas de trabalho é conhecido, o problema então passa a ser selecionar dentre os elementos (jornadas) desse conjunto, uma jornada para cada um dos tripulantes, de tal forma que todas as tarefas sejam cumpridas por algum dos tripulantes.

Em suma, o problema consiste em se selecionar um conjunto de jornadas de trabalho de tal forma que se cubra todas as tarefas, com o menor custo operacional total possível. Tal problema é conhecido como *Set Covering Problem* (SCP), onde há um conjunto de linhas a serem cobertas, e um conjunto de colunas com custos associados, cada uma cobrindo um subconjunto dessas linhas. O objetivo é então selecionar um subconjunto de colunas de tal forma que se cubra todas as linhas, com o menor custo possível. O SCP é um problema $\mathcal{NP}$-*Difícil* [59, 11] extensamente estudado na literatura.

- Denota-se por meio de e o vetor coluna formado apenas por $1's$, ou seja, $e^T = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \end{pmatrix}$.

Segue-se então para a formulação do SCP.

$$\min c^T x \tag{2.2}$$

$$\text{sujeito a:} \tag{2.3}$$

$$Ax \geq e \tag{2.4}$$

$$x \in \mathbb{B}^n \tag{2.5}$$

Na solução do SCP, já refletindo com respeito ao PPT, se alguma tarefa é coberta por mais de uma jornada selecionada como solução, na verdade, na prática o que de fato ocorre é que, parte da tripulação viaja como passageiro, enquanto outra efetivamente cobre (cumpre) a tarefa em questão. Desta forma economizam-se custos de hospedagem, e além disso, uma tripulação pode iniciar o trabalho em um local distinto daquele onde havia concluído sua última tarefa, ainda que este esteja a milhas de distância. Tal procedimento é conhecido como *deadhead*, e é comum em empresas aéreas. Todavia não é indicado para empresas rodoviárias, pois caso

a distância seja muito longa é possível que grande parte do horário de trabalho da tripulação seja consumida em deslocamentos. Logo o mais adequado para tais casos, é resolver o PPT como um *Set Partitioning Problem* (SPP) .

$$\min c^T x \tag{2.6}$$

$$\text{sujeito a:} \tag{2.7}$$

$$Ax = e \tag{2.8}$$

$$x \in \mathbb{B}^n \tag{2.9}$$

Nota-se que o SPP e SCP, sob o ponto de vista das formulações apresentadas divergem essencialmente no que concerne à (2.4) e (2.8), pois para o SPP cada uma das linhas deve ser coberta por uma, e apenas uma, coluna. Este detalhe dificulta muito a resolução do SPP em relação ao SCP, pois, até mesmo verificar se existe uma solução viável para o SPP é também $\mathcal{NP}$-*Difícil* [59].

Através das formulações (2.2)–(2.5) e (2.6)–(2.9) nota-se que toda solução viável do SPP implica em uma solução viável do SCP. Em contrapartida a solução ótima do SCP pode nem ao menos representar uma solução viável no SPP correspondente.

Entretanto, de uma forma geral, as mesmas técnicas de solução podem ser empregadas com o objetivo de resolver quaisquer dos dois problemas, sendo mais difícil de se manter a viabilidade da solução quando trabalha-se com o SPP, o que degrada consideravelmente o desempenho de tais métodos.

Para o caso de programação de tripulações, aqui abordado, cada jornada de trabalho é delegada a um tripulante, ou uma equipe única (tripulação), não admitindo assim múltipla cobertura, como o SCP permite. No estudo de caso que aqui faz-se, *deadheads* não são admissíveis, por conseguinte o SPP é o problema com a formulação mais adequada (compatível).

Quanto ao emprego desta formulação, sob o estudo de caso do PPT em questão, este é devidamente discutido ao longo da seção 4.9. Maiores detalhes teóricos com respeito ao SCP e SPP podem ser encontrados em [59].

2.4 Meta-heurísticas baseadas em busca local

O procedimento de busca local representa o pilar básico para os métodos pertencentes a esta classe. Em linhas gerais, uma busca local começando de uma solução inicial s^0 navega pelo espaço de pesquisa, através de um movimento m passando de uma solução à outra que seja sua vizinha vislumbrando assim melhorar o valor de uma função de avaliação $f : s \in S \rightarrow \mathbb{R}$ a ser otimizada, tanto quanto isso seja possível, ou seja vislumbra-se encontrar ao final, uma solução s tal que forneça o melhor custo com respeito a f. Define-se o conjunto de soluções vizinhas de s como:

$$\mathcal{N}(s) = \{s' : s' \text{ está nas proximidades de } s\}.$$

Movimento Denomina-se movimento uma transformação/modificação m tal que, quando aplicada transforma uma solução qualquer s em uma outra, s', a qual esteja em sua vizinhança, ou seja, esteja em suas proximidades. Denota-se por meio de $m \oplus s \Rightarrow s'$ tal operação, logo pode-se então reescrever o conjunto de soluções que representam uma vizinhança de s sob a forma:

$$\mathcal{N}(s) = \{s' : m \oplus s \Rightarrow s'\}. \tag{2.10}$$

Movimento reverso Vale enfatizar que para cada movimento m gerado a partir de uma dada solução s sempre existe um outro movimento m^{-1} reverso, tal que seja capaz de gerar novamente a solução s anterior.

$$\mathcal{N}^{-1}(s') = \{s : m^{-1} \oplus s' \Rightarrow s\}. \tag{2.11}$$

Este conceito é muito útil quando trata-se da aplicação de meta-heurísticas que possuem memória, a Busca Tabu, por exemplo.

Estruturas de vizinhança Geralmente é possível extrair de um mesmo problema vários tipos "distintos" de estrutura de vizinhança, ou seja, diversas famílias de movimentos. Normalmente tem-se uma família elementar e a partir desta compõem-se as demais. Contudo vale enfatizar que, o quão maior for a "complexidade" da transformação, maior é o custo computacional de gerenciamento para tratá-las.

O conjunto que define a vizinhança localmente completa de um dado problema pode ser escrito sob a forma:

$$\mathcal{N}(s) = \left\{ \bigcup_{k=1}^{r} \mathcal{N}_{(s)}^{(k)} \right\},$$

onde r representa o número de vizinhanças distintas de um problema, ou sob uma forma mais geral, apenas aquelas com as quais se deseja trabalhar.

Ademais, os tipos de estrutura de vizinhança para o PPT aqui consideradas encontram-se devidamente definidos ao longo da seção 4.3.

Movimentos redundantes Algumas vezes durante a análise de uma vizinhança qualquer é possível deparar-se em uma situação tal que um movimento ao ser aplicado à solução incumbente gere a própria solução. Tais movimentos conduzem a uma redundância, e devem ser descartados, além de inúteis podem ainda confundir a busca, bem como aumentar o esforço necessário para análise da vizinhança. Embora tais soluções não sejam estritamente iguais para com respeito a estrutura, elas são simétricas.

Recomenda-se uma boa análise do quanto irá se despender em busca de movimentos redundantes, pois caso forem muito particulares certamente não valerão tal esforço, exceto quando for possível definir uma estrutura tal que seja capaz de tratá-los implicitamente.

Independente do método de busca local selecionado deve-se também optar por uma estratégia de exploração de vizinhança, ou seja deve-se também definir como a vizinhança da solução atual, isto é, aquele que se tem posse em uma dada iteração ou passo qualquer deve ser explorada.

Estratégias de exploração de vizinhança Dentre um conjunto de opções, a heurística mais conhecida, e mais sistemática é a chamada *Melhor Vizinho* (*Best Improvement*) (BI). Aplicar a heurística BI significa que, dado uma solução analisa-se toda a sua vizinhança em busca do vizinho que ofereça o melhor custo.

$$s'' = \underset{s' \in \mathcal{N}(s)}{\operatorname{argmin}} \{f(s')\}$$

Contudo, a depender do problema abordado, a busca pelo melhor vizinho pode vir a ser computacionalmente muito onerosa. De fronte a existência deste complicador é comum objetivando dar celeridade ao processo, o emprego de outras heurísticas, tais como:

1. *First Improvement* realiza a busca pela primeira solução de melhora, ou seja, interrompe a busca local tão cedo quanto encontre uma solução com custo melhor que a solução atual. Contudo esta heurística torna-se equivalente a BI, quando a solução que ofereça tal propriedade é a última, ou se simplesmente não há um vizinho que ofereça tanto.

2. Busca restrita a um certo percentual da vizinhança;

3. Empregar a alternativa 2, no entanto incrementar o fator percentual, gradativamente conforme a qualidade das soluções produzidas;

4. Busca restrita a um número fixo de vizinhos gerados sob forma aleatória.

Não há comprovação de qual dentre as estratégias supracitadas comporta-se de forma mais adequada, pois a qualidade de tal escolha pode estar associada a estrutura do problema. Tendo isso em vista, recomenda-se experimentos iniciais utilizando estratégias que considerem vizinhanças mais amplas, e assim estimar se haverá ganho ou não ao lançar mão de parte do espaço de soluções vizinhas.

Condição de parada Finalmente resta dizer que devido ao não-determinismo destas metodologias requer-se então que uma condição de parada seja estabelecida, por meio de algum critério. A condição de parada destes métodos dá-se geralmente sob formas tais como:

- o tempo de computação;
- número de iterações;
- alcance de uma solução tão boa quanto um custo que já se conheça, ou seja, um limitante (*bound*).

Ao longo das duas próximas seções (2.4.1) e (2.4.2) apresentam-se respectivamente as meta-heurísticas Busca Tabu e *Iterated Local Search*, as quais prestam-se como pilares para o desenvolvimento das abordagens aqui propostas.

2.4.1 Busca Tabu

A meta-heurística Busca Tabu (BT) proposta originalmente em [13] caracteriza-se principalmente por dispor do artifício de se admitir soluções de qualidade inferior durante o processo de busca, na tentativa de escapar de ótimos locais, para que possam ser explorados locais distantes vislumbrando um ótimo global ou ao menos uma solução de boa qualidade.

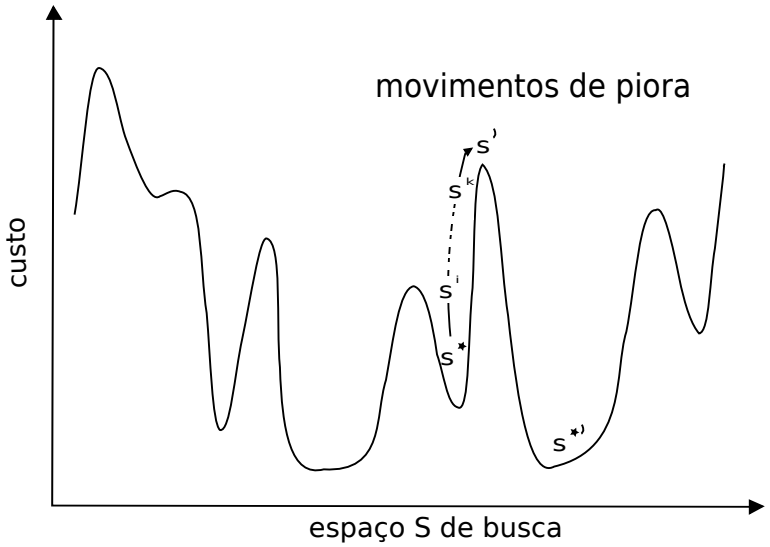


Figura 2.6: Comportamento esperado pelo uso da Busca Tabu.

Através da Figura 2.6 ilustra-se o comportamento vislumbrado pela aplicação da Busca Tabu. Supondo um mínimo local, a busca prossegue selecionando o melhor dentre os vizinhos, daquela solução.

Ainda por meio da Figura 2.6 observa-se que o "gargalo" ao se admitir soluções que piorem o custo é possibilidade de incidência de ciclos, o que conduz o método a um espaço de busca visitado anteriormente. Com o objetivo de reduzir as chances de que isso ocorra, uma estrutura chamada lista tabu é empregada.

A lista tabu consiste em uma memória de curto prazo onde os últimos movimentos referentes ao caminho percorrido pela busca são armazenados. Tais movimentos são considerados proibidos e assim permanecem por um certo período chamado prazo tabu (*tabu tenure*). Duas formas mais comuns de configuração de prazo tabu são empregadas: a estática e a dinâmica. A configuração é dita dinâmica se seu valor é estabelecido através de características extraídas ao longo da busca. A configuração é dita estática, caso o prazo tabu permaneça fixo durante toda a execução do método.

A lista tabu pode eventualmente fazer com que a busca prive-se de soluções promissoras, por conseguinte provocar a estagnação da busca. Tendo isso em vista estabelece-se uma condição chamada critério de aspiração. O critério de aspiração revoga a proibição de um movimento, se este conduz a uma solução que ofereça algum atrativo.

Comumente o critério de aspiração é defino por objetivo (função de avaliação), ou seja, ainda que um movimento seja tabu, este é aceito caso conduza a uma solução $s^{'}$ melhor que a solução s^{*} obtida até então, onde obviamente tal solução ainda não foi visitada. Outros critérios de aspiração mais "complexos" foram propostos em trabalhos tais como [60, 61], entretanto raramente são empregados [16].

Dentre as principais propriedades da Busca Tabu estão:

- *lista tabu*;
- *critério de aspiração*;

Tais propriedades devem estar bem definidas, para aumentar as chances de que uma solução de qualidade seja alcançada ao final do método.

Baseando-se em [16] apresenta-se a meta-heurística Busca Tabu, através do Algoritmo 2. Neste Algoritmo, além da notação já introduzida no início desta seção emprega-se:

- $\mathcal{T}$, lista tabu;
- $\tilde{\mathcal{N}}(s) \subseteq \mathcal{N}(s)$, subconjunto admissível de $\mathcal{N}(s)$ (ou seja, não tabu, ou aceitos por aspiração).

Algoritmo 2 Busca Tabu

requer solução inicial s^0

assegura solução final $s^* : f(s^*) \leq f(s^0)$

1: $s \leftarrow s^0$

2: $s^* \leftarrow s^0$

3: $\mathcal{T} \leftarrow \emptyset$

4: **enquanto** condição de parada não satisfeita **faça**

5: $s \leftarrow \underset{s' \in \tilde{\mathcal{N}}(s)}{\operatorname{argmin}} \{f(s')\}$

6: **se** $f(s) < f(s^*)$ **então**

7: $s^* \leftarrow s$

8: **fim se**

9: $\mathcal{T} \leftarrow \mathcal{T} \cup m^{-1}$, onde {além disso apaga movimento mais antigo, se necessário.}

10: **fim enquanto**

Com respeito ao algoritmo 2. A busca Tabu dá-se sob os seguintes passos. O método recebe como entrada uma solução s^0 inicial. Considera-se inicialmente s^0 como sendo a melhor solução s^* e solução s incumbente (linhas 1 e 2).

A busca consiste na avaliação da vizinhança restrita $\tilde{\mathcal{N}}(s)$ da solução corrente, ou seja seleciona-se o melhor dentre os vizinhos, tal que este, não seja tabu ou atenda ao critério de aspiração (linha 5). A solução vizinha selecionada torna-se então a solução s corrente (linha 5). A nova solução corrente é avaliada em relação a melhor solução obtida até então (linha 6). Se a nova solução é $s : f(s) < f(s^*)$, a solução corrente torna-se também a melhor solução fazendo $s^* \leftarrow s$ (linha 7). O movimento m^{-1} reverso a aquele que induziu a nova solução corrente é então armazenado como tabu. Além disso o movimento mais antigo é eliminado da lista, se for o caso (linha 9). Este processo é repetido até que se satisfaça uma dada condição de parada. A solução final fornecida pelo método é $s^* : f(s^*) \leq f(s^0)$.

Aspectos teóricos com respeito à meta-heurística Busca Tabu foram investigados por [62] e [63]. Todavia segundo Pigatti [64], até aquele momento não havia sido apresentada nenhuma explicação formal verificável com respeito ao bom comportamento da Busca Tabu.

2.4.2 Iterated Local Search

A meta-heurística *Iterated Local Search* (ILS) baseia-se na simples ideia de que uma solução produzida por um procedimento de busca local pode ser melhorada gerando-se novas soluções de partida. Em outras palavras, pode-se dizer que a ILS é um método de busca local baseado na ideia de *multi-start*, o qual trabalha com várias soluções de partida. Ao longo dos anos esta simples ideia vem sendo redescoberta pelos mais diversos trabalhos presentes na literatura [16].

A grande diferença entre o método *multi-start* tradicional e a ILS é que na primeira, uma solução inicial é sempre gerada de forma totalmente aleatória, portanto não possui qualquer relação com as outras soluções de início, ao passo que a ILS tenta melhorar a solução produzida pelo procedimento de busca local provocando perturbação neste mínimo local, e o reconsidera no momento em que a nova solução inicial é gerada. A ILS então foca sua busca não no espaço completo de soluções, mas em um subespaço definido por soluções que são ótimas locais, fornecidas por um procedimento de busca local.

Através da Figura 2.7 é possível observar o comportamento esperado com a aplicação da meta-heurística ILS. Partindo de um mínimo local s^*, aplica-se então o procedimento de *perturbação* lançando-se à uma solução s'. Após a aplicação de um procedimento de *busca local*, encontra-se um novo mínimo $s^{*\prime}$, do qual vislumbra-se uma solução com o custo melhor, em relação às demais já visitadas até então.

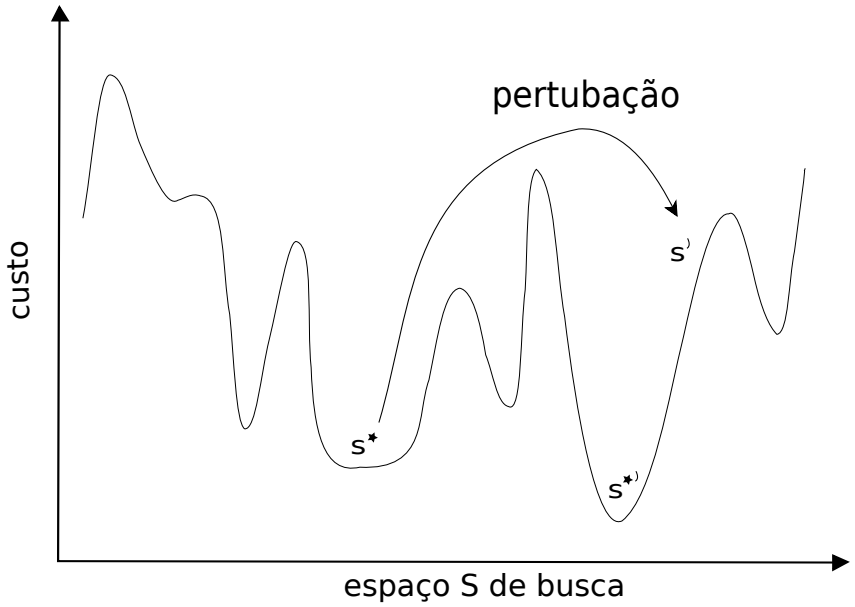


Figura 2.7: Comportamento da ILS

Observa-se que o processo da ILS dá-se efetivamente sempre que se está de fronte a um mínimo local. O procedimento de perturbação é ativado toda vez que isso acontece. Dentre os requisitos para uma boa perturbação, ou seja, uma perturbação que a posteriori sob o efeito da busca local conduza a uma nova solução, e que esta seja de boa qualidade, a perturbação necessita ser suficientemente forte para permitir uma boa diversificação e portanto que a busca local explore diferentes soluções, em contrapartida deve também ser ténue o suficiente para evitar um reinício aleatório.

Além dos procedimentos mais comuns exigidos por métodos baseados em busca local, outros 3 (três) fazem parte do arquétipo utilizado na ILS. Seguem tais componentes, bem como suas respectivas descrições:

- perturbação, modifica uma solução s^* guiando a uma solução intermediária s';
- histórico, armazena informações com respeito a evolução alcançada ao longo do processo. Este item é pouco empregado na prática, pois geralmente é computacionalmente custoso [16];
- critério de aceitação, o qual decide em que solução a próxima perturbação será aplicada.

Os passos envolvidos na meta-heurística ILS são apresentados através do Algoritmo 3.

Algoritmo 3 Iterated Local Search

requer solução inicial s^0

assegura solução final $s^* : s^* \leq s^0$

1: $s^* \leftarrow$ buscaLocal(s^0)

2: **repita**

3: $s' \leftarrow$ pertubação(s^*, histórico)

4: $s^{*'} \leftarrow$ buscaLocal(s')

5: $s^* \leftarrow$ critérioAceitação($s^*, s^{*'}$, histórico)

6: **até que** condição de parada satisfeita

Com respeito ao Algoritmo 3. Definido tais elementos dá-se então os passos envolvidos no método. Dada uma solução inicial, uma busca local é então aplicada como passo preambular (linha 1). De fronte a uma solução s^*, a qual representa um "mínimo local", o método inicia. Produz-se a solução s' por meio de perturbação em s^* (linha 3). O procedimento de busca local é então aplicado na solução s', por meio disso $s^{*\prime}$ é produzida. O *critério de aceitação* decide, se essa nova solução pode, ou não ser aceita (linha 5), caso positivo passa a ser a nova solução incumbente; caso contrário mantêm-se a solução s^*, que já se tinha posse. Este processo persiste até que uma condição de parada seja satisfeita (linha 6).

O próximo capítulo dedica-se aos detalhes dos elementos envolvidos no Problema de Programação de Tripulações, sob um estudo de caso de uma empresa de transporte público coletivo que opera na cidade de Belo Horizonte.

Capítulo 3

Problema de Programação de Tripulações de Ônibus Urbano

De acordo com [51], no Brasil, Belo Horizonte (BH) é uma das 5 (cinco) cidades mais populosas, onde nesta vivem atualmente mais de 3 (três) milhões de pessoas em toda área metropolitana. A cidade atualmente, conta com apenas uma só linha de transporte do tipo metroviário, pois é uma cidade que possui uma geografia expressivamente montanhosa o que caracteriza um sério complicador e dificulta assim a implantação desse tipo de transporte público coletivo. Por conseguinte o transporte público naquela é predominantemente composto por ônibus.

Ainda com respeito à cidade de BH. Há várias diferentes linhas de ônibus, algumas dessas partem de um bairro para o centro e retornam ao mesmo bairro, e há outras que partem de uma região da cidade para outra, atravessando a cidade. Diferentes empresas operam tais linhas, no entanto as mesmas regras são válidas para quaisquer dessas. Cada linha é controlada por apenas uma empresa, entretanto uma empresa geralmente tem mais de uma linha nesse caso [65]. Vale acrescentar que, a *BHTRANS* é a empresa gerenciadora de transporte coletivo daquele município [66].

Conforme anteriormente discutido (ver Capítulo 2) antes de se realizar a etapa que diz respeito à programação de tripulação é necessário particionar os blocos de viagens dos veículos em tarefas. Para o estudo de caso que aqui se faz, cada uma das tarefas consiste em sequencias de viagens cujos intervalos entre elas é de no máximo 5 (cinco) minutos. Intervalos maiores que 5 (cinco) minutos caracterizam oportunidades de troca, ou seja, uma tripulação pode ser substituída por outra.

Apresenta-se através da Tabela 3.1, um exemplo de composição de tarefas.

(a) viagens

viagem	ponto	horário	ponto	horário		
(v_j)	inicial		final		linha	
0	0	253	1	260	332	t_1
1	1	260	1	282	332	
2	1	302	1	325	332	t_2
3	1	332	1	357	332	t_3
4	1	362	1	390	332	
5	1	392	1	426	332	
6	2	426	2	453	331	
7	2	454	2	483	331	
8	2	485	2	513	331	
9	2	525	2	557	331	t_4
10	2	575	2	606	330	t_5
0	2	606	0	613	331	

(b) tarefas

tarefa	veículo	horário		ponto		folga	linha		grupo de linha	
(t_i)		inicial	final	inicial	final	acumulada	inicial	final	inicial	final
1	12	253	282	0	1	0	332	332	0	0
2	12	302	325	1	1	0	332	332	0	0
3	12	332	513	1	2	10	332	331	0	0
4	12	525	557	2	2	0	331	331	0	0
5	12	575	613	2	0	0	331	331	0	0

Tabela 3.1: Bloco de viagens de um veículo particionado em tarefas

Conforme exemplo exibido através da Tabela 3.1, a tarefa 3 reúne as viagens 3–8. Isto ocorre pois o intervalo entre elas é inferior ao mínimo tido como suficiente para que ocorra troca de tripulações, operação esta, que exige no mínimo 6 minutos conforme mencionado anteriormente. Por conseguinte, a tarefa 3 acumula então, uma folga de 10 minutos, a qual representa a soma dos intervalos entre todas as viagens envolvidas. Essa mesma Tabela ilustra ainda a composição das tarefas 1, 2, 4 e 5. Observa-se que sempre a primeira tarefa é composta de no mínimo 2 (duas) viagens, pois a primeira viagem do bloco representa o deslocamento entre a garagem (ponto 0), onde se encontra o veículo até o primeiro ponto. O mesmo ocorre com a última tarefa, já que a última viagem do bloco é o deslocamento do veículo até a garagem.

Vale ressaltar que a modelagem do PPT apresentada neste trabalho considera dados de uma empresa de transporte público que opera na cidade de Belo Horizonte, Minas Gerais, Brasil. Segundo [50], as regras operacionais e restrições são relativas a Convenção Coletiva de Trabalho 2002/2003 firmada entre o Sindicato das Empresas de Transporte de Passageiros de Belo Horizonte (SETRABH) e o Sindicato dos Trabalhadores em Transporte Rodoviário de Belo Horizonte (STTRBH).

Como descrito anteriormente o PPT possui diversas restrições, as quais devem ser contempladas ao longo do processo de alocação das tarefas que compõem as jornadas. A seção seguinte lista e descreve cada umas das restrições inerentes ao estudo de caso que aqui se aborda.

3.1 Restrições/requisitos do PPT

Usualmente em trabalhos tais como [35, 67, 48] classificam as restrições deste problema basicamente em dois subgrupos: essenciais e não-essenciais. As restrições essenciais são aquelas que não podem ser violadas pela solução final do método, ou seja, devem ser satisfeitas para que uma solução seja factível. As restrições não-essenciais, mesmo que sejam violadas, ainda assim podem compor uma solução dita factível, mas a sua violação deve ser evitada para que seja possível obter uma solução final do método otimizada.

Entretanto, antes de definir as restrições que devem ser consideradas na solução do problema, algumas definições devem ser introduzidas:

Pegada Simples : Jornada que possui absolutamente todos os intervalos entre suas tarefas menores que 2 (duas) horas.

Dupla Pegada : Um intervalo do tipo *Dupla Pegada* é um intervalo superior a 2 (duas) horas entre duas tarefas consecutivas pertencentes a uma mesma jornada. Uma jornada é então dita do tipo *Dupla Pegada*, se existe nela, um e apenas um, intervalo do tipo *Dupla Pegada*. Vale acrescentar que tal intervalo não é contabilizado como tempo trabalhado, ou seja, não é remunerado.

Duração regular de uma jornada: A duração regular de uma jornada de trabalho diário é de:

- 7 (sete) horas e 10 (dez) minutos para as tripulações do tipo *Pegada Simples*;
- 6 (seis) horas e 40 (quarenta) minutos para aquelas do tipo *Dupla Pegada*.

O fundamento desta distinção está associado diretamente ao tempo de descanso, a que uma tripulação possui direito. Jornadas do tipo *Dupla Pegada* já possuem um intervalo longo, então supõe-se que este já é suficiente para tanto. Tal propriedade se clarifica por meio da descrição das restrições vinculadas a ociosidade.

Todas as restrições do problema são apresentadas por meio das próximas seções.

3.1.1 Restrições essenciais

As seguintes restrições essenciais são consideradas neste trabalho:

1. Sobreposição: uma tripulação não é capaz de executar mais que uma tarefa ao mesmo tempo, portanto uma jornada de trabalho não deve conter duas ou mais tarefas com horários sobrepostos;

2. Hora excedente: a duração de uma jornada de trabalho não deve exceder em mais do que 2 (duas) horas a duração de uma jornada regular;

3. Tempo de descanso: se uma jornada de trabalho é do tipo *Pegada Simples*, a tripulação associada tem direito a trinta minutos de folga (descanso, alimentação, etc...). Este período pode eventualmente ser fragmentado, entretanto ao menos um destes fragmentos deve conter no mínimo quinze minutos. Tal período é, normalmente computado como tempo de trabalho;

4. Troca de linha proibida: a jornada de uma tripulação não pode conter duas tarefas consecutivas de tal forma que a linha final da primeira e a linha inicial da seguinte sejam distintas, se estas não pertencerem a um mesmo conjunto chamado *grupo de linhas*;

5. Troca de ponto proibida: a jornada de uma tripulação não pode conter duas tarefas consecutivas de tal forma que o ponto final da primeira e ponto inicial da seguinte sejam distintos, exceto em intervalos que caracterizam *Dupla Pegada*. Tal propriedade é proibida, pois supõe-se que são necessários ao menos 2 (duas) horas para que uma tripulação possa deslocar-se de um ponto a outro.

6. Intervalo diário deficiente: entre o final de uma jornada de trabalho de uma tripulação e o seu início no dia seguinte deve haver um intervalo mínimo de 11 (onze) horas. Supõe-se que uma jornada é executada pela mesma tripulação todos os dias (ver Figura 3.1).

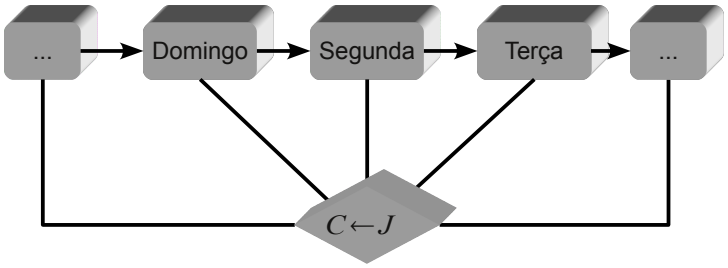


Figura 3.1: Intervalo de descanso diário

7. Excesso de *Dupla pegada*: o número de jornadas do tipo *Dupla Pegada* não deve exceder um certo limite imposto pela empresa;

3.1.2 Restrições não-essenciais

Os seguintes itens constituem as restrições não-essenciais, ou seja, aqueles que são aceitos na solução do problema, mas penalizados na função de avaliação, de forma tal que, o número de ocorrências destes itens seja minimizado.

1. Hora extra: tempo em que a duração da jornada excede a duração regular de trabalho. Este tempo é limitado em 120 (cento e vinte) minutos. O que ultrapassa isso é hora excedente conforme restrição 2, seção 3.1.1;

2. Tempo ocioso: tempo em que o funcionário fica ocioso, ou seja, não realiza nenhuma tarefa;

3. Troca de veículo: quando há troca de veículos entre tarefas consecutivas dentro de uma mesma jornada;

4. Troca de linha permitida: quando em uma mesma jornada, há troca de linhas entre duas tarefas consecutivas, onde ambas linhas pertençam ao mesmo grupo.

5. Troca de ponto permitida: quando em uma mesma jornada, há troca de pontos entre duas tarefas consecutivas, e o intervalo entre elas é do tipo *Dupla Pegada*;

Nota-se por meio da discussão realizada ao longo deste Capítulo que o PPT, bem como grande parte dos problemas reais de otimização possuem vários objetivos. É notável o caráter multiobjetivo intrínseco a este problema. Sabe-se que em uma função $f : s \to \mathbb{R}$ possui-se uma relação de ordem, ou seja, dadas duas soluções quaisquer sabemos qual dentre as duas é de fato a melhor. Entretanto quando se trata de uma função sob a forma $f : s \to \mathbb{R}^n$ que vislumbra contemplar vários objetivos, não há uma relação de ordem bem definida.

O presente trabalho não se adentra por este tema, entretanto, mais com respeito a Programação Multiobjetivo pode ser encontrado em trabalhos tais como [68].

De fronte a complicadores tais como o supra mencionado, o presente trabalho atem-se a uma abordagem mono-objetiva. Compõe-se então uma função de avaliação $f : s \to \mathbb{R}$ para o problema, esta deve ser minimizada, e corresponde ao número de jornadas na solução, acrescido do termo que penaliza a ocorrência dos itens descritos ao longo desta seção. A função aqui empregada baseia-se em [35]. A função de avaliação, bem como a forma com que cada um dos requisitos deve ser contabilizado e penalizado são devidamente apresentadas através da seção 4.8.

Capítulo 4

Métodos Propostos

Neste capítulo são apresentadas as metodologias empregadas para a resolução do PPT, bem como sua modelagem. Para a resolução com sucesso, todas as restrições do problema devem ser bem definidas.

A solução do PPT é dada por um conjunto de jornadas, sendo que as jornadas de cada uma das tripulações são compostas por tarefas e cada tarefa contém uma série de propriedades, são elas: número de identificação, grupo de linhas, horário de início e término, linha inicial e final, ponto inicial e final, número de identificação do veículo e folga acumulada. Todos estes atributos foram obtidos a partir do procedimento descrito na seção 2.1. A Tabela 4.1 apresenta a estrutura completa de uma tarefa.

tarefa (t)	
horário inicial	horário final
linha inicial	linha final
ponto inicial	ponto final
folga acumulada	veículo

Tabela 4.1: Estrutura de uma tarefa

Através da Tabela 4.2 apresenta-se a estrutura de jornada. Nesta estrutura, além do número de identificação, há também a contabilização correspondente a cada uma das restrições do problema (ver Capítulo 3). A notação apresentada nesta mesma tabela é empregada estritamente na seção 4.2.

jornada de trabalho ($\mathcal{J}$)			
troca de veículo	$\mathrm{tv}(\mathcal{J})$	hora extra	$\mathrm{he}(\mathcal{J})$
tempo ocioso	$\mathrm{to}(\mathcal{J})$	hora excedente	$\overline{\mathrm{he}}(\mathcal{J})$
sobreposição	$\mathrm{so}(\mathcal{J})$	dupla pegada	$\mathrm{dp}(\mathcal{J})$
troca de ponto permitida	$\mathrm{tp}(\mathcal{J})$	troca de ponto proibida	$\overline{\mathrm{tp}}(\mathcal{J})$
troca de linha permitida	$\mathrm{tl}(\mathcal{J})$	troca de linha proibida	$\overline{\mathrm{tl}}(\mathcal{J})$
duração	$\mathrm{du}(\mathcal{J})$	duração regular	$\mathrm{dr}(\mathcal{J})$

Tabela 4.2: Estrutura de uma jornada

Nota-se também, que tais valores contabilizados são dinâmicos, e conforme novos padrões (combinações) de alocações vão sendo "experimentados", tais valores modificam-se, e por conseguinte conduzem à composição da função de avaliação. A descrição da função de avaliação é delegada à seção 4.8.

Inicialmente são definidas as seguintes etapas: Tratamento da entrada de dados (seção 4.1), Geração da solução inicial (seção 4.2), Estruturas de vizinhança do PPT (seção 4.3), Heurística para controle do número de jornadas (seção 4.4). A forma como aqui foi definido, tais etapas são comuns à todas aquelas metodologias propostas que tomam como base meta-heurísticas. Em seguida são descritos os aspectos inerentes à cada uma das proposições: BT (seção 4.5) e ILS (seção 4.7).

Nas seções correspondentes às proposições, os procedimentos envolvidos são apresentados em ordem de independência. A condição de parada é a mesma para todos os métodos, e encontra-se definida no próximo capítulo.

A seção 4.8 descreve como cada um dos requisitos do PPT deve ser contabilizado, bem como os custos associados a cada um deles. Assim a função de avaliação é apresentada ao final daquela seção.

Ao final deste capítulo, na seção 4.9 apresenta-se a abordagem exata, a qual baseia-se na formulação do Problema de Particionamento de Conjuntos.

4.1 Entrada de dados

Assumindo que a entrada de dados para o problema (PPT) é sempre gerada como produto de uma solução viável para o problema anterior (PPV) (ver Figura 2.1) é possível assegurar que:

Para tarefas realizadas por um mesmo veículo pode-se afirmar:

- as mesmas encontram-se agrupadas na entrada de dados;
- dispostas em ordem crescente, com respeito ao horário em que devem ser realizadas;
- não se sobrepõem;
- entre quaisquer pares de tarefas consecutivas não há troca de pontos, pois o veículo sempre se desloca imediatamente até o ponto onde será realizada a próxima tarefa.

Tendo tais fatores em vista é possível construir facilmente, uma solução inicial viável para o problema avaliando-se isoladamente cada um dos blocos de tarefas pertencentes a um mesmo veículo. A geração da solução inicial é então o assunto tratado ao longo da próxima seção.

4.2 Geração da solução inicial

Conforme exposto na seção 2.4 do Capítulo 2, para inicializar metodologias tais como as que aqui se propõem é necessário gerar uma solução inicial. A geração de uma solução inicial para este problema pode ser feita de inúmeras maneiras, a maioria delas baseia-se na forma com que a programação das jornadas outrora era manualmente realizada [40].

Para construção da solução inicial, nesse trabalho propõe-se uma heurística sequencial, onde privilegia-se a viabilidade e o baixo custo computacional. Tal heurística é apresentada através do Algoritmo 4. A notação utilizada neste algoritmo está disposta na Tabela 4.2.

Algoritmo 4 Método para geração da solução inicial

requer $\mathcal{E}$: lista de tarefas (entrada de dados).

assegura solução inicial s^0.

1: $i \leftarrow 1$

2: $\mathcal{J}_i \leftarrow \emptyset$

3: **para** $t_j \in \mathcal{E}, j = 1, 2, \ldots, |\mathcal{E}|$ **faça**

4: **se** $\mathcal{J}_i = \emptyset$ **então**

5: $\mathcal{J}' \leftarrow \mathcal{J}_i \cup \{t_j\}$

6: **se** $\mathrm{du}(\mathcal{J}') > \mathrm{dr}(\mathcal{J}') \vee \mathrm{dp}(\mathcal{J}') > 0 \vee \mathrm{tl}(\mathcal{J}') + \overline{\mathrm{tl}}(\mathcal{J}') > 0 \vee \mathrm{tv}(\mathcal{J}') > 0$ **então**

7: $i \leftarrow i + 1$

8: $\mathcal{J}_i \leftarrow \emptyset$ {nova jornada vazia}

9: **fim se**

10: **fim se**

11: $\mathcal{J}_i \leftarrow \mathcal{J}_i \cup \{t_j\}$ {aloca tarefa}

12: **fim para**

No Algoritmo 4. Inicializa-se a construção de uma solução inicial tomando-se um índice $i \leftarrow 1$ uma jornada vazia $\mathcal{J}_i \leftarrow \emptyset$ (haverá no mínimo uma jornada, pois considera-se que a entrada de dados nunca é vazia). Em uma primeira iteração o índice $j = 1$, naturalmente $\mathcal{J}_i = \emptyset$, logo a primeira tarefa é alocada na jornada i. Já na segunda iteração tomando $j = 2$, a jornada i possui uma tarefa, uma jornada auxiliar $\mathcal{J}'$ simula a atribuição da tarefa t_j à jornada $\mathcal{J}_i$ através de $\mathcal{J}' \leftarrow \mathcal{J}_i \cup \{t_j\}$. Se $\mathcal{J}'$ ultrapassa a duração regular de uma jornada, provoca *Dupla Pegada*, *troca de linha*, ou *troca de veículo*, o índice i é incrementado fazendo $i \leftarrow i + 1$, ou seja uma nova jornada vazia é inicializada e assim faz-se: $\mathcal{J}_i \leftarrow \mathcal{J}_i \cup \{t_j\}$ atribuindo a tarefa incumbente a esta nova jornada; caso contrário, o índice i permanece inalterado e a tarefa t_j é então atribuída à jornada $\mathcal{J}_i$. O processo persiste até que $j = |\mathcal{E}|$, ou seja, até que todas tarefas da entrada de dados sejam contempladas (linha 2).

Dentre as vantagens apresentadas pelo uso desta técnica estão: garantia de que a solução inicial não viole quaisquer das restrições essenciais, e além disso, exceto pelo requisito *tempo ocioso*, as restrições não-essenciais também não são violadas. Em contra partida a solução inicial é sempre idêntica se os dados de entrada (programação dos veículos) forem os mesmos.

Conforme discutido ao longo do Capítulo 2, meta-heurísticas tais como a Busca Tabu e *Iterated Local Search* realizam sua busca basicamente explorando um espaço de soluções através de uma busca local. Cada solução encontrada nesse espaço é denominada solução vizinha. O espaço explorado pela busca, ou seja, as soluções vizinhas que serão analisadas, é definido pela *Estrutura de Vizinhança*. A definição das estruturas de vizinhança para o caso de PPT aqui tratado é feita na próxima seção.

4.3 Estruturas de vizinhança

Considerando-se s como sendo uma dada solução, uma solução s' é dita vizinha de s, se é possível chegar em s', com apenas um só movimento (ver seção 2.4).

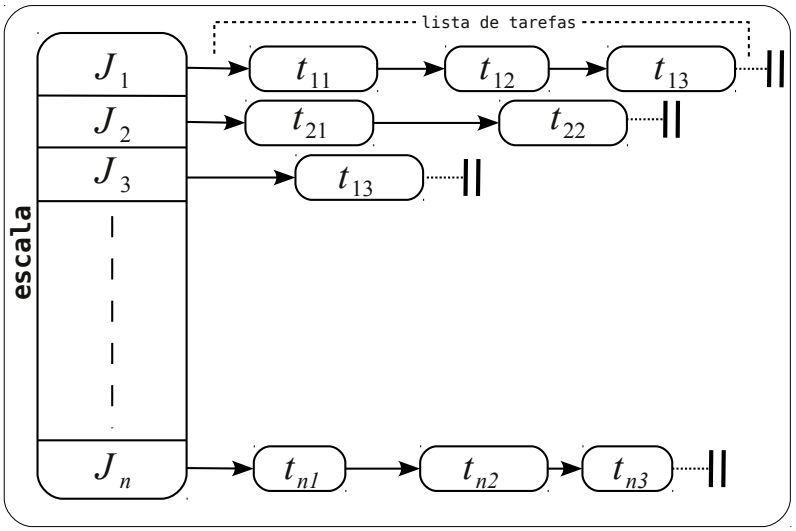


Figura 4.1: Estrutura de escala

Baseando-se na estrutura do problema (ver Figura 4.1), duas possibilidades básicas de movimentação para o problema são mais evidentes. Estes dois tipos de movimentos elementares distintos são então os movimentos empregados: *Realocação* $\mathcal{N}_{(s)}^{(\mathcal{R})}$ e *Troca* $\mathcal{N}_{(s)}^{(\mathcal{T})}$.

Delimitados os tipos de estrutura vizinhança de trabalho, logo tem-se algo como 4.1, a qual denota a *vizinhança "completa"* de uma solução s qualquer.

$$\mathcal{N}(s) = \left\{ \mathcal{N}_{(s)}^{(\mathcal{R})} \bigcup \mathcal{N}_{(s)}^{(\mathcal{T})} \right\} \tag{4.1}$$

Nas próximas seções os tipos de movimentos são plenamente descritos. Além disso conforme supra mencionado é realizada a análise de movimentação redundante, para cada tipo de estrutura de vizinhança.

4.3.1 Movimentos de realocação

O movimento de *Realocação* $\mathcal{N}_{(s)}^{(\mathcal{R})}$ consiste em realocar uma tarefa t pertencente a uma jornada $\mathcal{J}$, a uma outra jornada $\mathcal{J}'$, ou seja:

$$\mathcal{J}' \leftarrow \mathcal{J}' \cup \{t\} : t \in \mathcal{J}; \mathcal{J} \leftarrow \mathcal{J} \setminus \{t\}.$$

Tal movimento é ilustrado através da Figura 4.2.

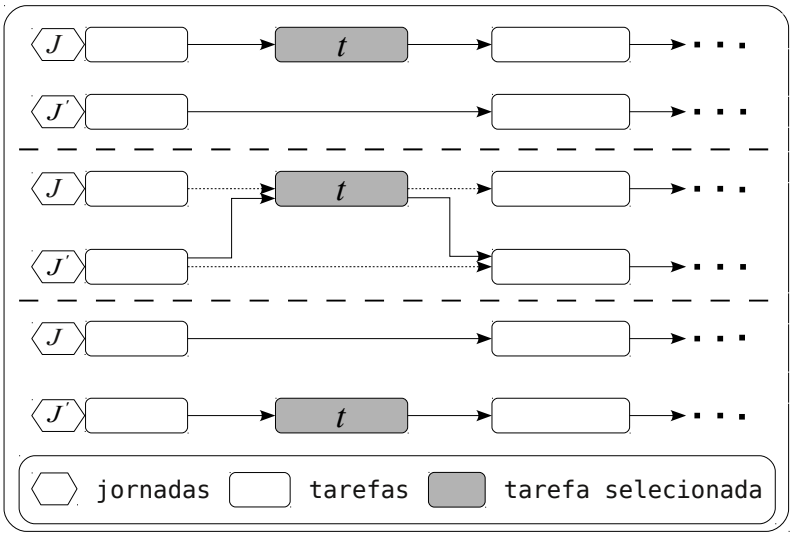


Figura 4.2: Movimento de *Realocação* $\mathcal{N}_{(s)}^{(\mathcal{R})}$

Suponha agora que:

$$\mathcal{J}, \mathcal{J}' \in s, |\mathcal{J}| = 1, \mathcal{J}' = \emptyset, \tag{4.2}$$

onde a jornada $\mathcal{J}$ possui uma única tarefa e a jornada $\mathcal{J}'$ encontra-se vazia, ou seja, completamente ociosa. Se um movimento tal como este for realizado incide em uma solução simétrica, logo todos movimentos desta família são desprezados. Tal situação é também ilustrada por meio da Figura 4.3.

Figura 4.3: Movimento redundante de *Realocação* $\mathcal{N}_{(s)}^{(\mathcal{R})}$

4.3.2 Movimentos de troca

O movimento de *Troca* $\mathcal{N}_{(s)}^{(\mathcal{T})}$ consiste em permutar-se duas tarefas quaisquer t e t' entre duas tripulações distintas $\mathcal{J}$ e $\mathcal{J}'$:

$$m_T = \begin{cases} m_R = \mathcal{J} \leftarrow \mathcal{J} \cup \{t'\} : t' \in \mathcal{J}'; \mathcal{J}' \leftarrow \mathcal{J}' \setminus \{t'\} \\ m'_R = \mathcal{J}' \leftarrow \mathcal{J}' \cup \{t\} : t \in \mathcal{J}; \mathcal{J} \leftarrow \mathcal{J} \setminus \{t\} \end{cases}$$

Tal movimento é ilustrado através da Figura 4.4.

Nota-se que também é possível fazer analogia com o movimento de *Realocação*, por conseguinte concluir que o movimento de *Troca* é simplesmente: produto da composição de 2 (dois) movimentos de *Realocação*.

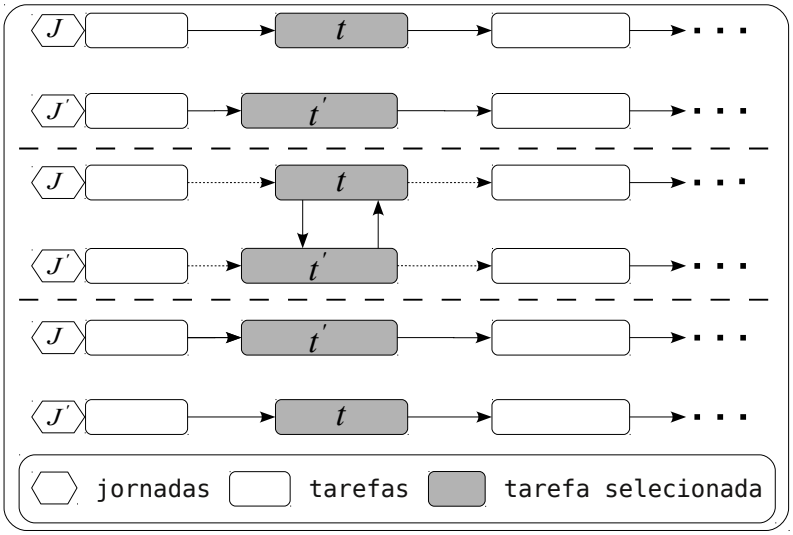


Figura 4.4: Movimento de *Troca* $\mathcal{N}_{(s)}^{(\mathcal{T})}$

Semelhante ao caso anterior (ver seção 4.3.1). Para todos os casos onde $\mathcal{J}$ e $\mathcal{J}'$ possuem uma única tarefa cada, tais movimentos devem ser sumariamente desprezados, pois conduzem à soluções simétricas.

$$\mathcal{J}, \mathcal{J}' \in s, |\mathcal{J}| = |\mathcal{J}'| = 1 \tag{4.3}$$

A Figura 4.5 ilustra o tipo de movimento redundante obtido através da vizinhança $\mathcal{N}_{(s)}^{(\mathcal{T})}$.

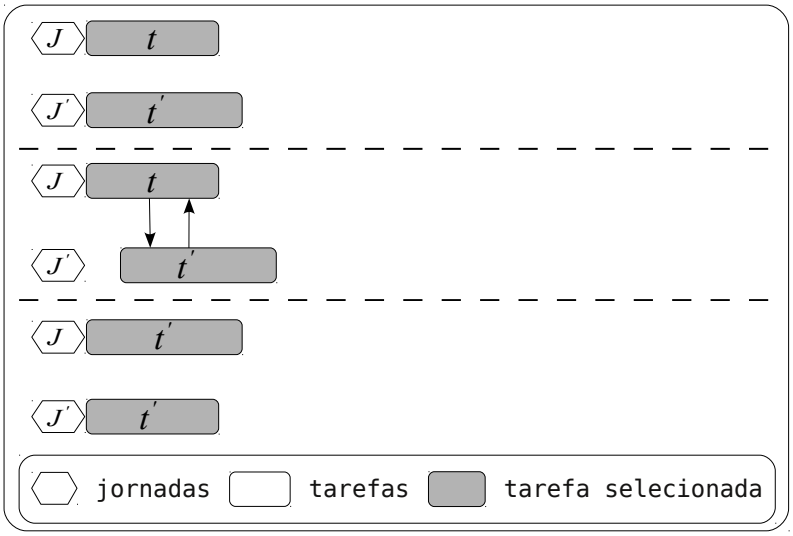


Figura 4.5: Movimento redundante de *Troca* $\mathcal{N}_{(s)}^{(\mathcal{T})}$

4.4 Heurística para controle do número de jornadas

Uma das coisas da qual vislumbra-se ao resolver o PPT é obviamente a minimização do número de jornadas, entretanto devido a estrutura com que se trabalha, a medida em que movimentos de realocação vão sendo aplicados, jornadas presentes na solução tendem a ficar vazias (completamente ociosas).

Eliminar as jornadas vazias da estrutura da solução parece atrativo, pois assim evita-se a geração de combinações desnecessárias. Em contrapartida corre-se o risco de eliminar tantas jornadas, a ponto da solução ficar com um número menor do que o de uma solução de qualidade superior. Observe que da forma como foi aqui definida, durante a execução de uma busca local, o número de jornadas pode diminuir, mas nunca aumenta. Vislumbrando reduzir as chances de que isso ocorra, impõe-se a seguinte regra: toda vez em que duas jornadas estiverem vazias, uma delas permanece e a outra é eliminada da estrutura da solução.

Ademais, sempre que um movimento de realocação é feito, em seguida, o que se faz é verificar se a jornada que fornece a tarefa torna-se vazia, e se assim for verifica-se, se já existe uma outra jornada vazia, em caso afirmativo eliminada-se a última. Tal procedimento é ilustrado por meio da Figura 4.6.

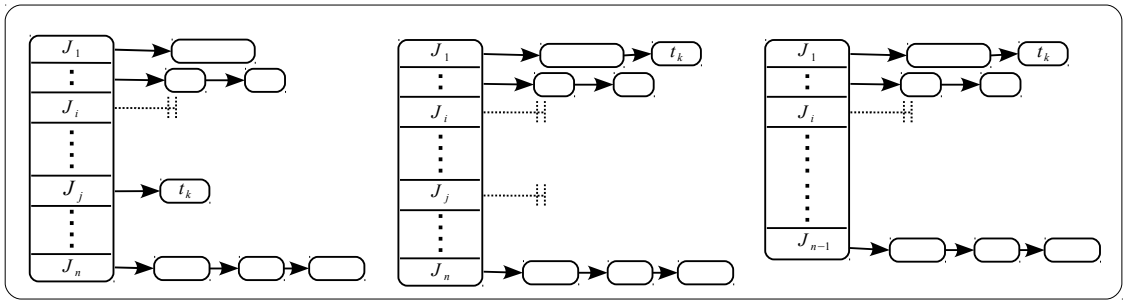


Figura 4.6: Eliminação de excesso de jornadas vazias

Vale destacar a importância em se tratar este aspecto, pois, a cada jornada eliminada da estrutura do problema, um número de combinações igual ao tamanho da entrada, deixam de ser consideradas na análise da vizinhança, o que pode acelerar a busca significativamente.

Uma desvantagem é que em um dado momento pode-se substituir a melhor das soluções por uma nova solução que à princípio pode parecer muito atrativa.

Entretanto o número de jornadas a qual restringe-se naquele momento pode ser inferior ao de soluções de qualidade superior, as quais naquele momento obviamente, ainda não se conhece.

4.5 Busca Tabu

Essa seção apresenta a abordagem que considera a Busca Tabu como base. A implementação proposta envolve lista tabu estática, busca local baseada na heurística *Best Improvement* e critério de aspiração baseado em função de avaliação. Todos estes procedimentos e estruturas envolvidos são descritos à seguir, inicia-se pela lista tabu, da qual sua definição independe dos demais procedimentos.

Lista Tabu Definiu-se aqui, que a lista tabu comporta-se exatamente como uma fila circular, ou seja, os movimentos mais antigos são eliminados à medida em que novos movimentos são adicionados e a fila se encontra cheia. O comportamento desse algoritmo é ilustrado através da Figura 4.7.

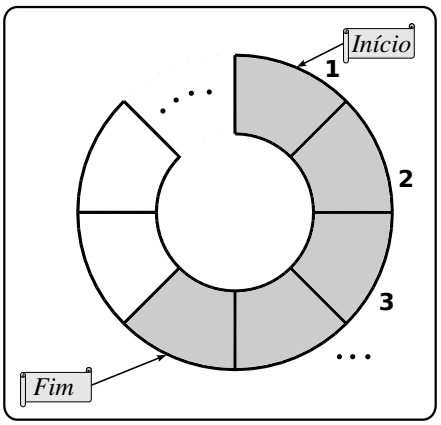


Figura 4.7: Tabu circular (fila circular)

O armazenamento dos movimentos na *lista tabu* fica a cargo dos passos descritos no Algorítimo 5, onde a notação adicional empregada é:

- $\mathcal{T}_i$, representa a i-ésima posição na lista tabu;
- $|\mathcal{T}|_{\max}$, representa a cardinalidade máxima do conjunto que define a lista tabu. Seu valor é definido a priori.

Algoritmo 5 Lista Tabu

requer m, movimento.

assegura $\mathcal{T}$ lista tabu atualizada.

1: $\mathcal{T}_i$ {representa a i-ésima posição na lista tabu.}

2: $i \leftarrow \min\{i+1, |\mathcal{T}|_{\max}\}$ {calcula posição do novo movimento.}

3: $\mathcal{T}_i \leftarrow m^{-1}$ {atualiza lista tabu.}

No Algoritmo 5. Nota-se que o índice i é sempre calculado de modo à respeitar a cardinalidade máxima. Além disso vale enfatizar que o custo de inserção nesta lista é constante, e não há custo em remoção pois, o movimento mais antigo é substituído pelo mais novo, a partir do momento que a lista encontra-se cheia.

Busca local Apresenta-se por meio do Algoritmo 6 a busca local. A expressão $\neg\mathcal{T}(s \oplus m)$ significa que o movimento m não se encontra na lista tabu ($m \notin \mathcal{T}$), e a solução s pode ser aceita.

Algoritmo 6 Busca local tabu

requer $\mathcal{T}$:lista tabu, s:solução corrente, s^*:melhor solução conhecida.

assegura $s' : \underset{s' \in \tilde{\mathcal{N}}(s)}{\operatorname{argmin}}\{f(s')\}$

1: $s' \leftarrow \text{random}\{\mathcal{N}(s)\}$

2: **para todo** $s'' : s'' \in \mathcal{N}(s) \setminus \{s'\}$ **faça**

3: **se** $f(s'') < f(s') \wedge \left(\neg\mathcal{T}(s'') \vee \underbrace{f(s'') < f(s^*)}_{\text{critério de aspiração}} \right)$ **então**

4: $s' \leftarrow s''$

5: **fim se**

6: **fim para**

7: $\mathcal{T} \leftarrow \mathcal{T} \cup \{m^{-1}\}$, onde $s' = s \oplus m$

No Algoritmo 6, o objetivo é selecionar a melhor solução vizinha de s. Isso é feito iterativamente, da seguinte forma: toma-se aleatoriamente uma solução vizinha de s, como referência (linha 1). A cada iteração um vizinho de s é analisado. Faz-se $s' \leftarrow s''$ (linha 4), se é melhor que a última melhor solução vizinha encontrada, e não é tabu ou se é melhor que a melhor solução conhecida s^* (linha 3). Este procedimento encerra-se quando todos os vizinhos de s forem analisados (linha 2).

O último passo (linha 7) referece-se ao Algoritmo 5 que é responsável por gerenciar a lista tabu.

Ainda por meio do Algoritmo 6 observa-se, que o *critério de aspiração* empregado (linha 3) guia-se estritamente pelo valor da função de avaliação.

Apresenta-se pelo Algoritmo 7 o procedimento de nível mas elevado da proposição, o que quer dizer que este é o algoritmo responsável por gerenciar os demais procedimentos.

Algoritmo 7 Busca Tabu

requer solução inicial s^0

assegura solução final $s^* : f(s^*) \leq f(s^0)$

1: $s \leftarrow s^0$

2: $s^* \leftarrow s^0$

3: $\mathcal{T} \leftarrow \emptyset$

4: **enquanto** condição de parada não satisfeita **faça**

5: $s \leftarrow \underset{s' \in \tilde{\mathcal{N}}(s)}{\operatorname{argmin}} \{f(s')\}$

6: **se** $f(s) < f(s^*)$ **então**

7: $s^* \leftarrow s$

8: **fim se**

9: **fim enquanto**

O algoritmo requer uma solução s^0 inicial (ver seção 4.2). Tal solução é inicialmente considerada a melhor das soluções $s^* \leftarrow s^0$ (linha 2), é também considerada a solução corrente $s \leftarrow s^0$ (linha 1). A lista tabu inicialmente encontra-se vazia (linha 3). Após a inicialização destes atributos, a busca é iniciada. A busca ocorre enquanto a condição de parada não for alcançada (linha 4).

A cada iteração do método uma nova solução corrente é produzida, segundo as especificações expostas no Algoritmo 6 de busca local. Se a nova solução oferece melhor custo do que a melhor solução conhecida, esta então toma o seu lugar fazendo $s^* \leftarrow s$ (linha 7).

4.6 *Variable Neighborhood Descent*

Nesta seção é apresentado o Método de Descida em Vizinhança Variável (*Variable Neighborhood Descent*, VND) [69], este é empregado como método de busca local na proposição que baseia-se na meta-heurística *Iterated Local Search*, tal proposição é apresentada na próxima seção.

O VND é um método de busca local que explora o espaço de soluções por meio de trocas sistemáticas de estruturas de vizinhança, aceitando somente soluções de melhora da solução corrente e retornando à primeira estrutura, toda vez em que há melhora.

$$\mathcal{N}_{(s)}^{(k)} = \left\{\mathcal{N}_{(s)}^{(1)}, \mathcal{N}_{(s)}^{(2)}, \ldots, \mathcal{N}_{(s)}^{(r)}\right\}$$

- Explora o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança;
- Explora vizinhanças gradativamente mais "distantes";
- Sempre que há melhora em uma certa vizinhança, retorna-se à vizinhança "menos distante"[1], ou seja, $\mathcal{N}_{(s)}^{(1)}$ (o menor índice está diretamente associado ao tipo de estrutura de vizinhança dita "menos distante").

As características supra mencionadas podem ser observadas por meio da Figura 4.8.

[1] Alguns autores empregam esse termo para dizer que um tipo de estrutura de vizinhança aplica menos sub-transformações na solução.

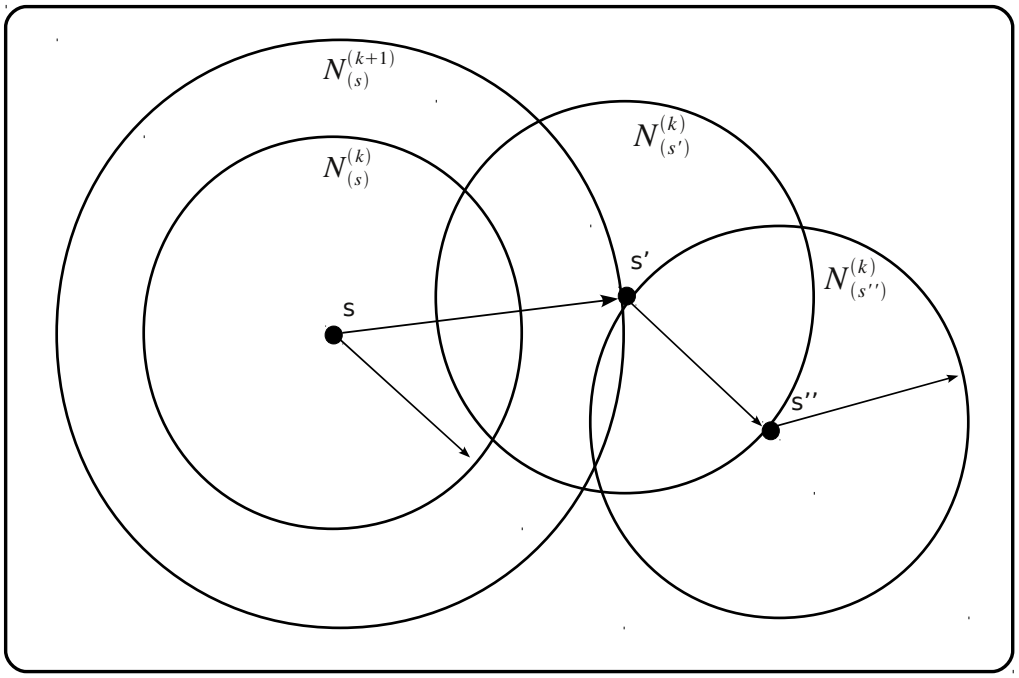


Figura 4.8: Comportamento da heurística VND

A VND é apresentada através do Algoritmo 8.

Algoritmo 8 Variable Neighborhood Descent

requer solução inicial s^0

assegura solução final $s^* : f(s^*) \leq f(s^0)$

1: $s^* \leftarrow s^0$
2: $k \leftarrow 1$
3: **enquanto** $k \leq r$ **faça**
4: $s \leftarrow \underset{s' \in \mathcal{N}_{(s^*)}^{(k)}}{\operatorname{argmin}} \{f(s')\}$
5: **se** $f(s) < f(s^*)$ **então**
6: $s^* \leftarrow s$
7: $k \leftarrow 1$ {reinicia na primeira estrutura de vizinhança}
8: **se não**
9: $k \leftarrow k + 1$
10: **fim se**
11: **fim enquanto**

A busca persiste enquanto o valor de k for menor que o valor da constante r. Para a abordagem aqui proposta $r = 2$, pois 2 (dois) tipos de vizinhança são considerados

(ver seção 4.3).

A proposta do VND baseia-se sob três pilares básicos:

- Não há nada que assegure que: um ótimo local associado a um tipo de estrutura $\mathcal{N}_{(s)}^{(k)}$ corresponda a um ótimo local com respeito a qualquer um dos demais tipos de estrutura;
- Um ótimo global corresponde a um ótimo local para todos os demais tipos de estrutura de vizinhança;
- Vislumbra-se que para uma variedade de problemas, ótimos locais com respeito a um tipo de estrutura de vizinhança encontram-se nas proximidades, em relação aos demais tipos de estruturas de vizinhança.

Empiricamente observa-se que um ótimo local frequentemente fornece algum tipo de informação com respeito ao ótimo global. Este é o caso em que os ótimos locais e o ótimo global compartilham uma parcela relativamente considerável de variáveis com mesmo valor, portanto sugere-se uma investigação sistemática da vizinhança de um ótimo local até a obtenção de uma nova solução de melhor valor.

4.7 *Iterated Local Search* com *Variable Neighborhood Descent*

Apresenta-se nesta seção a proposição que adota a meta-heurística Iterated Local Search. Considera-se cada um dos elementos definidos por meio da seção 2.4.2, bem como a descrição de cada um dos aspectos envolvidos. Nesta implementação estão envolvidos: procedimento de perturbação baseado na estrutura de vizinhança, busca local baseada em *Variable Neighborhood Descent* com heurística *Best Improvement*, e critério de aceitação baseado na função de avaliação. Para esta proposição o procedimento de perturbação é o mais independente dos demais, e por isso é apresentado à seguir.

Perturbação No procedimento de perturbação, sucessivos movimentos aleatórios baseados na estrutura de vizinhança são empregados. A Figura 4.9 ilustra o procedimento empregado.

Neste caso define-se que, perturbar a solução significa caminhar a cada iteração, aleatoriamente de vizinho para vizinho, sem que a solução construída seja examinada sob qualquer aspecto.

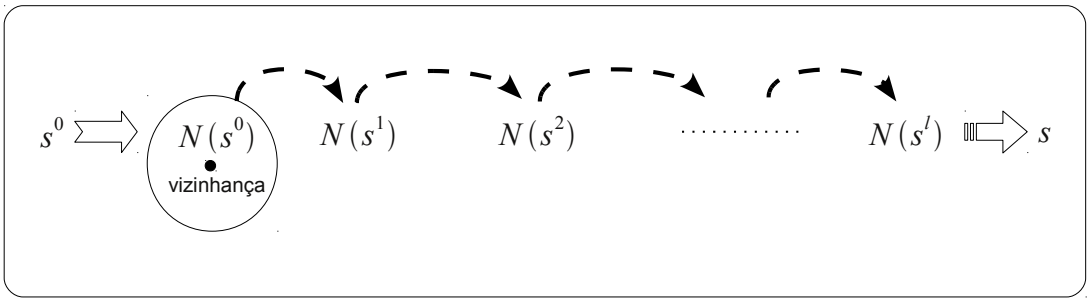


Figura 4.9: Perturbação

O procedimento de perturbação é exposto por meio do Algoritmo 9.

Algoritmo 9 Perturbação

requer s^0: Solução inicial/base; l: nível de pertubação.

assegura s: Solução perturbada em um nível l.

1: $s \leftarrow s^0$

2: **para** $i = 1, \ldots, l$ **faça**

3: $\quad s' \leftarrow \text{random}\,\{\mathcal{N}(s)\}$ {seleciona aleatoriamente uma solução vizinha de s.}

4: $\quad s \leftarrow s'$

5: **fim para**

O procedimento supra apresentado toma como parâmetros uma solução s^0 e nível l de perturbação, onde este último é um valor que representa em quantas vizinhanças sucessivas caminha-se.

Busca local Emprega-se como método de busca local a heurística *Variable Neighborhood Descent*, a qual encontra-se descrita por meio da seção 4.6. Vale acrescentar que as estruturas de vizinhança são consideradas na mesma ordem em que foram apresentadas neste trabalho, ou seja, considera-se primeiro a vizinhança de realocação e em seguida a de troca, pois a primeira é a mais elementar.

Critério de aceitação Baseando-se na melhor solução conhecida decide-se, se a próxima perturbação será aplicada na nova solução ou na melhor solução conhecida,

ou seja, o critério só aceita a nova solução s, se ela for melhor que s^*, que representa a melhor solução encontrada. Se por este critério decidir-se pela melhor solução conhecida, o nível de pertubação é então incrementado, caso contrário ele é reiniciado. O nível l de pertubação encontra-se definido no intervalo $l_{\min} \leq l \leq l_{\max}$, onde $l_{\min}$ e $l_{\max}$ representam nível de perturbação mínimo e máximo respectivamente, e são definidos a priori.

Algoritmo 10 Critério de aceitação

requer s: solução candidata, s^*:melhor solução conhecida, l:nível de pertubação.

assegura $s' : s' \leq s^*$

1: **se** $f(s) < f(s^*)$ **então**
2: $l \leftarrow l_{\min}$ {reinicia nível de perturbação}
3: $s' \leftarrow s$ {aceita a nova solução}
4: **se não**
5: $l \leftarrow \min\{l+1, l_{\max}\}$ {incrementa nível de perturbação}
6: $s' \leftarrow s^*$ {mantém solução}
7: **fim se**

Se uma dada solução s oferece um custo melhor que s^*, essa nova solução é aceita fazendo $s' \leftarrow s$, além disso faz-se $l \leftarrow l_{\min}$, reiniciando assim o nível de perturbações; caso contrário, $l \leftarrow \min\{l+1, l_{\max}\}$ fazendo com que o nível de pertubação seja incrementado, além disso faz-se $s' \leftarrow s^*$, ou seja, a solução retornada pelo procedimento é a mesma que lhe foi passada como parâmetro. Note que se $l = l_{\max}$, as próximas pertubações persistem neste nível até que uma solução de melhor custo seja alcançada, e assim o nível de perturbação é reiniciado com valor $l_{\min}$.

Dado que todos os procedimentos agora encontram-se bem definidos, estes são invocados por meio do Algoritmo 11, o qual os coordena.

Algoritmo 11 Iterated Local Search para o PPT

requer $l_{\min}$: nível de pertubação mínimo, $l_{\max}$: nível de pertubação máximo.

requer solução inicial s^0

assegura solução final $s^* : f(s^*) \leq f(s^0)$

1: $s^* \leftarrow \text{VND}(s^0)$ {busca Local}

2: $l \leftarrow l_{\min}$

3: **repita**

4: $s' \leftarrow \text{pertubação}(s^*, l)$

5: $s^{*'} \leftarrow \text{VND}(s^{'})$ {busca Local}

6: $s^* \leftarrow \text{critérioAceitação}(s^*, s^{*'}, l)$

7: **até que** condição de parada satisfeita

Com respeito ao Algoritmo 11. Obtém-se uma solução s^* por meio da busca local VND feita sob uma dada solução s^0 inicial (linha 1). O procedimento inicia-se de fato gerando-se a solução s' por meio de l movimentos de perturbação na solução corrente (linha 4). A busca local VND é feita a partir de s' gerando a solução $s^{*\prime}$ (linha 5). A depender do critério de aceitação (ver algoritmo 10), o nível l de perturbação e solução s^* são devidamente atualizados. O processo como um todo (linhas 4–6) persiste até que uma condição de parada (linha 7) seja satisfeita.

4.7.1 Memória local de perturbações

Uma variação de ILS é proposta. Nesta emprega-se um mecanismo, para o qual adota-se o termo *Memória Local de Perturbações* (MILS). Este consiste em uma memória de curto prazo a qual impede que em uma mesma sessão de perturbações, uma perturbação realize um movimento reverso a outro que já havia sido realizado naquela sessão o que caracterizaria em uma exclusão mútua entre aqueles. Este mecanismo é semelhante ao empregado na lista tabu da Busca Tabu. Apresenta-se tal metodologia por meio do Algoritmo 12.

Algoritmo 12 Pertubação com memória local

requer s^0: Solução inicial/base, l: nível de pertubação.

assegura s, Solução perturbada em um nível l.

1: $s \leftarrow s^0$

2: $\mathcal{T} \leftarrow \emptyset$

3: **para** $i = 1, \ldots, l$ **faça**

4: $\quad s'' \leftarrow \underset{s' \in \mathcal{N}(s) \setminus \{\mathcal{T}\}}{\text{random}} \{s'\}$

5: $\quad \mathcal{T} \leftarrow \mathcal{T} \cup \{m^{-1}\}$

6: $\quad s \leftarrow s''$

7: **fim para**

No Algoritmo 12 a cada iteração, a pertubação aplicada é atribuída a um conjunto $\mathcal{T}$ (linha 5). Além disso observa-se também, que esta memória (tabu) é mantida apenas localmente, ou seja, após o termino de uma seção de pertubações, esta é completamente eliminada. O procedimento utilizado para gerenciar esta memória (linha 5) é exatamente o mesmo apresentado no Algoritmo 5, da Busca Tabu (seção 4.5).

4.8 Função de Avaliação

Esta seção apresenta a função de avaliação empregada para mensurar a qualidade das soluções produzidas pelas metodologias propostas. A função empregada neste trabalho baseia-se em [35]. São também aceitas soluções inviáveis, entretanto a ocorrência de inviabilidades é penalizada de tal forma a privilegiar sua eliminação [48].

A função de avaliação é apresentada em duas parcelas, onde a primeira $f' : s \rightarrow \mathbb{R}$ corresponde às restrições essenciais, e a segunda $f'' : s \rightarrow \mathbb{R}$, por sua vez às restrições não-essenciais. A forma com que os requisitos devem ser contabilizados também é apresentada juntamente com sua respectiva lista de restrtições.

Seja uma solução $s : s = \{\mathcal{J}_1, \ldots, \mathcal{J}_n\}$ formada por um conjunto de jornadas

$\mathcal{J}_k, k = 1, \ldots, n$.

$$f'(s) = \sum_{k=1}^{n} \sum_{i=1}^{5} \beta_i \times x_{ki} + \beta_6 \times x_6 \tag{4.4}$$

onde,

- x_{k1} : quantidade de tempo em que uma tarefa acontece ao mesmo tempo que outra em uma mesma jornada k;
- x_{k2} : quantidade de tempo em que a duração de uma jornada k excede além da hora extra;
- x_{k3} : tempo que falta para o intervalo mínimo diário (entre o final de uma jornada k diária de trabalho e o seu início no dia seguinte);
- x_{k4} : número vezes em que há *troca de linha proibida* dentro de uma mesma jornada k;
- x_{k5} : número vezes em que há *troca de ponto proibida* dentro de uma mesma jornada k;
- x_6 : quantidade de jornadas do tipo *Dupla Pegada* excedentes.

Os valores referentes a estas penalidades encontram-se dispostos nas Tabelas 4.3.

	requisito	custo
x_{k1}	sobreposição	$\beta_1 = 40$
x_{k2}	hora excedente	$\beta_2 = 40$
x_{k3}	deficiência do intervalo entre jornadas	$\beta_3 = 40$
x_{k4}	troca de linha proibida	$\beta_4 = 35$
x_{k5}	troca de ponto proibida	$\beta_5 = 50$
x_6	excesso de *Dupla Pegada*	$\beta_6 = 45$

Tabela 4.3: Lista de restrições essenciais e seus respectivos custos

Segue então a parcela com respeito às restrições não-essenciais.

$$f''(s) = \sum_{k=1}^{n} \sum_{i=1}^{3} \alpha_i \times y_{ki} + \sum_{k=1}^{n} (\xi(y_{k4}) + \vartheta(y_{k5})) \tag{4.5}$$

onde,

- y_{k1} : número de vezes em que há *troca de linha permitida* dentro de uma mesma jornada k;
- y_{k2} : número de vezes em que há *troca de ponto permitida* dentro de uma mesma jornada k.
- y_{k3} : número de vezes em que há *troca de veículos* dentro de uma mesma jornada k;
- y_{k4} : quantidade de tempo em que a duração de uma jornada k excede a duração regular, ou seja, a contabilização das horas extras (vale relembrar que as horas excedentes não entram neste cálculo);
- y_{k5} : quantidade de tempo que uma tripulação fica ociosa, ao longo da sua jornada k de trabalho. Há três tipos distintos de ociosidade:
  - *folga acumulada* (ver seção 2.1);
  - entre pares de tarefas consecutivas, cujo intervalo entre elas é do tipo *Pegada Simples*;
  - entre a última tarefa e o tempo de trabalho regular (ver seção 3.1).

Os tipos de ociosidade encontram-se ilustrados por meio da Figura 4.10.

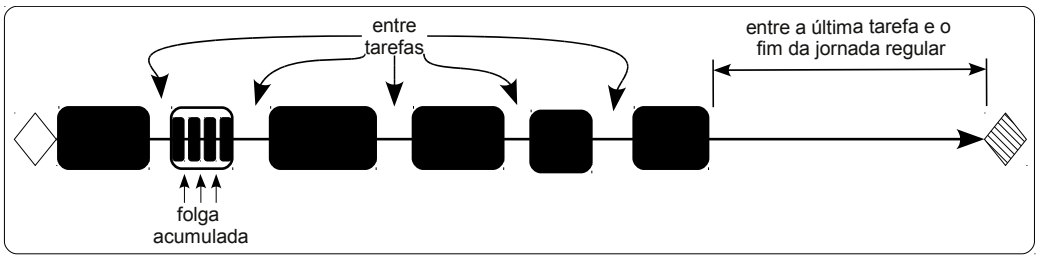


Figura 4.10: Ociosidade na jornada

Os valores referentes as penalidades das restrições não-essenciais encontram-se dispostos na Tabela 4.4.

	requisito	custo
y_{k1}	troca de linha permitida	$\alpha_1 = 5$
y_{k2}	troca de ponto permitida	$\alpha_2 = 5$
y_{k3}	troca de veículo	$\alpha_3 = 19$
y_{k4}	hora extra	$\xi(y_{k4}) = \lceil 1.357 \times y_{k4}^{0.6} \rceil$
y_{k5}	tempo ocioso	$\vartheta(y_{k5}) = \begin{cases} \lceil 0.347 \times y_{k5}^{0.8} \rceil, & \text{se } y_{k5} \leqslant 120; \\ 18, & \text{caso contrário.} \end{cases}$

Tabela 4.4: Lista de restrições não-essenciais e seus respectivos custos

A função de avaliação $f : s \to \mathbb{R}$ para o PPT é composta pela soma das duas parcelas supra apresentadas, e é dada pela expressão 4.6.

$$z = \min f(s) = f'(s) + f''(s) \tag{4.6}$$

Mais detalhes com respeito à esta função de avaliação e a contabilização das restrições para este problema podem ser vistos em [35].

4.9 Programação Matemática e Enumeração Exaustiva

Baseando-se na discussão apresentada na seção 2.3, sabe-se que o PPT pode ser modelado como um problema de programação inteira 0-1. Mais especificamente, ele pode ser modelado como um problema semelhante ao de particionamento de conjuntos (*Set Partitioning*). Neste modelo, considera-se todas as possíveis jornadas que podem construídas a partir das tarefas dadas.

Optou-se neste trabalho, por gerar todas as jornadas que não violem quaisquer das restrições essenciais do problema (ver seção 3.1.1) através de um procedimento de Enumeração Exaustiva (*Backtracking*).

Seja: $\mathcal{M} = \{1, \ldots, m\}$ o conjunto das tarefas dadas e $\mathcal{N} = \{1, \ldots, n\}$ o conjunto de todas as jornadas geradas. Defina para todo $i \in \mathcal{M}$ e $j \in \mathcal{N}$,

$$a_{ij} = \begin{cases} 1, & \text{se a tarefa } i \text{ pertence a jornada } j \\ 0, & \text{caso contrário} \end{cases}$$

e c_j como o custo associado a jornada j, o qual deve levar em consideração as penalizações descritas através das seções 3.1.2 e 4.8, o custo associado a cada uma das jornadas é calculado através expressão 4.5, naturalmente os valores dos parâmetros de penalização são os mesmos empregados nas abordagens baseadas em meta-heurística (ver Tabela 4.4).

Considere as seguintes variáveis de decisão:

$$x_j = \begin{cases} 1, & \text{se a jornada } j \text{ pertence à solução} \\ 0, & \text{caso contrário.} \end{cases}$$

Considere ainda:

$$a_{m+1j} = \begin{cases} 1, & \text{se a jornada } j \text{ é do tipo } \textit{Dupla Pegada} \\ 0, & \text{caso contrário.} \end{cases}$$

Pode-se, então, modelar o PPT como o problema abaixo, no qual o objetivo é selecionar as jornadas que atendam a todas as tarefas dadas com o menor custo possível.

$$\min \sum_{j=1}^{n} c_j x_j \tag{4.7}$$

sujeito a:

$$\sum_{j=1}^{n} a_{ij} x_j = 1 \qquad ,\forall i \in \mathcal{M} \tag{4.8}$$

$$\sum_{j=1}^{n} a_{m+1j} x_j \leq \mathcal{D}_{\max} \tag{4.9}$$

$$x_j \in \{0, 1\} \qquad ,\forall j \in \mathcal{N} \tag{4.10}$$

Observa-se que a linha $m + 1$ da matriz representa a restrição que diz respeito ao limite de jornadas do tipo *Dupla Pegada*, a constante $\mathcal{D}_{\max}$ é definida a priori.

Através do exemplo ilustrado na Figura 4.11 é então possível observar a atuação por parte de cada uma das entidades, sob o modelo.

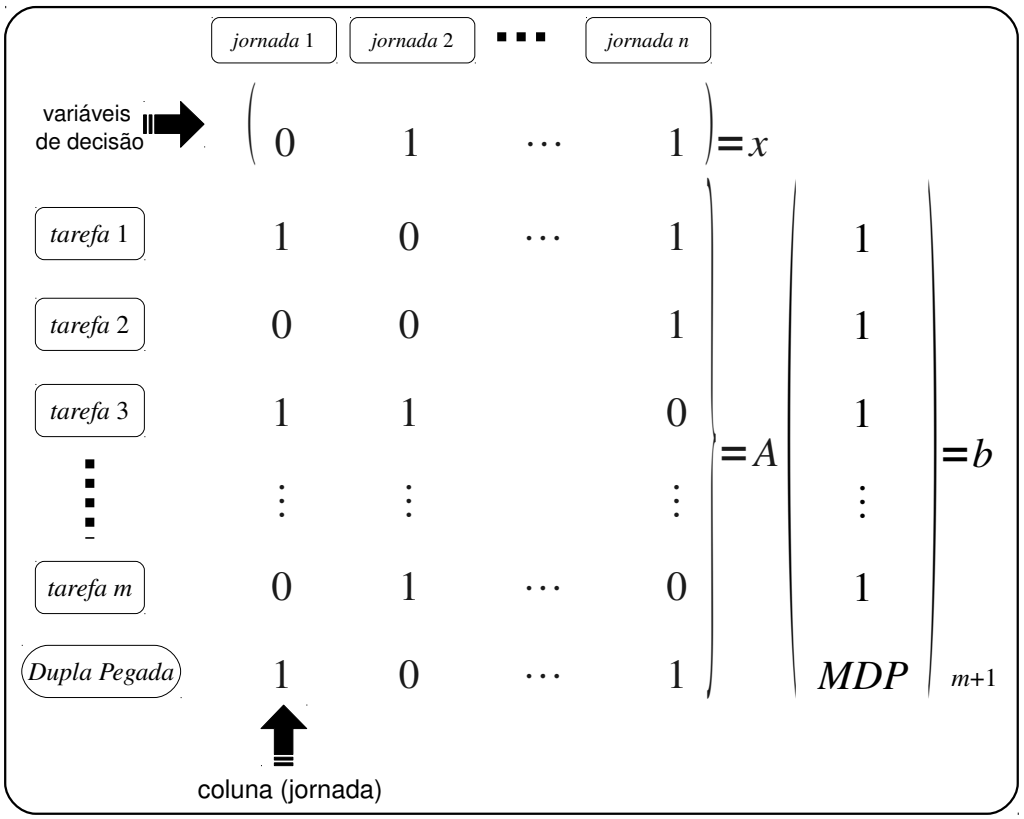


Figura 4.11: Modelagem

Capítulo 5

Resultados e Discussão

Todos experimentos mencionados nesse trabalho foram realizados no laboratório de otimização da PESC/COPPE/UFRJ. A máquina utilizada para tais possui a seguinte configuração: processador Intel® Xeon® X5472 3.00GHz, 16 gigabytes de memória RAM, sistema operacional Linux. Para a compilação dos algoritmos foi utilizado o compilador g++/GCC (*Gnu Compiler Collection*) com opção de otimização de compilação nível 3 (três) (otimizações agressivas), e para resolver o problema de programação matemática (4.7)–(4.10), foi utilizado o pacote *Dash Optimization*, módulos: *Xpress-Optimizer* e *Xpress-BCL*.

A Tabela 5.1 descreve sucintamente as metodologias implementadas.

método	descrição
BT	Busca Tabu.
ILS	*Iterated Local Search* com *Variable Neighborhood Search.*
MILS	*Iterated Local Search* com *Variable Neighborhood Search*, e memória local de perturbações.
PM	metodologia para obter a solução exata do modelo de programação matemática com o uso do *Xpress.*

Tabela 5.1: Descrição das metodologias

Vale ressaltar que, todas as metodologias aqui implementadas adotam *Best Improvement* (BI) como estratégia de exploração de vizinhança, pois essa foi a estratégia que apresentou os melhores resultados produzidos pelos experimentos preliminares, para quaisquer das abordagens propostas. Os resultados foram comparados com as estratégias *First Improvement* (FI) e *Método Randômico de Descida* (MRD).

5.1 Instâncias teste

Considera-se um conjunto de 32 (trinta e duas) instâncias de teste, as quais foram geradas a partir de dados reais de uma empresa que opera no município de Belo Horizonte/MG. As principais propriedades de tais instâncias encontram-se dispostas através da Tabela 5.2.

	instância	tarefas	veículos	pontos	linhas	dupla pegada
1	DOM-27	27	5	1	1	1
2	SEG-27	53	14	1	1	3
3	SEX-27	54	13	1	1	3
4	SEX-115	93	28	1	1	7
5	SEG-115	93	26	1	1	6
6	SAB-27	50	8	1	1	2
7	SEG-02	132	41	2	2	10
8	DOM-115	64	13	1	1	3
9	SEX-02	142	40	2	2	10
10	DOM-02	72	10	2	2	2
11	SAB-02	100	23	2	2	5
12	SAB-115	86	14	1	1	3
13	SEX-69	299	64	7	7	16
14	SEX-46	230	50	4	4	12
15	SEG-46	245	52	4	4	13
16	SEG-69	302	62	7	7	15
17	DOM-46	170	22	4	4	5
18	SEX-99	154	25	5	5	6
19	SAB-46	203	29	4	4	7
20	DOM-69	220	26	6	6	6
21	SAB-69	274	32	6	6	8
22	DOM-99	99	5	3	3	1
23	SEG-99	170	25	5	5	6
24	SAB-99	125	13	4	4	3
25	SEG-48	256	33	7	7	8
26	SEX-61	502	74	13	13	18
27	SEG-61	515	73	13	13	18
28	SAB-61	375	46	13	13	11
29	DOM-48	214	19	7	7	4
30	SEX-48	268	33	7	7	8
31	SAB-48	242	24	7	7	6
32	DOM-61	371	24	13	13	6

Tabela 5.2: Propriedades das instâncias de teste

Na última coluna há o número de *Dupla Pegada* permitida, este valor é fixo. Como o número de *Dupla Pegada* não faz parte da entrada optou-se por empregar: $0,1 \times 2,5 \times$ veiculos, que corresponde a uma estimativa para que 10% dos jornadas possam ser do tipo *Dupla Pegada* (estima-se que para cada veículo são necessárias $2,5$ jornadas). As demais propriedades contidas na Tabela 5.2 são dadas.

Quaisquer dos dados relacionados a tempo de processamento presentes nas tabelas apresentadas neste Capítulo encontram-se em "segundos" como unidade de medida.

A Tabela 5.3 apresenta o resumo de desempenho (valor da função de avaliação e tempo de processamento) da abordagem PM, ademais o número de variáveis contidas em cada um dos modelos gerados. Observa-se ainda, que os resultados encontram-se dispostos por ordem do tempo de processamento despendido para se encontrar a melhor das soluções, tal ordenação é utilizada como referência para as demais tabelas e figuras dispostas ao longo deste Capítulo.

Tais resultados representam o valor da solução ótima para 30, dentre os 32 problemas aqui considerados. Por conseguinte estes foram empregados como condição de parada para os demais experimentos, além disso emprega-se também com a finalidade de obter-se uma melhor análise, percepção e referência com respeito a qualidade das soluções produzidas pelas meta-heurísticas aqui implementadas.

	i	z^*	tempo (t)	variáveis
	1	170		510
	2	442		1687
	3	479		1976
	4	984	< 1	2788
	5	1007		4061
	6	275		7467
$\mathcal{A}$ $t \leq 60$	7	1234		12695
	8	398		10905
	9	1239		19078
	10	389	≈ 1	22205
	11	668		22693
	12	510		28972
	13	2266	19	200027
	14	1720	44	337381
	15	1625	53	553767

	i	z^*	tempo (t)	variáveis
	16	2132	62	590353
	17	850	76	695903
	18	800	80	753941
	19	1057	84	714260
	20	1024	291	1549063
	21	1135	303	2033172
$\mathcal{B}$ $60 < t \leq 3600$	22	257	328	1561491
	23	884	346	2067751
	24	498	409	2537833
	25	1193	1340	5579179
	26	2786	2200	7133283
	27	2669	2270	7338649
	28	1697	2294	9016961
	29	764	10566	15675395
$\mathcal{C}$ $t > 3600$	30	1205	12370	8562245

Tabela 5.3: Propriedades da solução ótima para as intâncias 1–30, obtidos por PM

	i	$\overline{z}$	Variáveis
$\mathcal{D}$ $t' \approx 3600$	31	865	44942535
	32	1073	73846881

Tabela 5.4: Limitantes obtidos por meta-heurísticas

Como pode ser observado através das tabelas supra apresentadas, não houve sucesso na busca por uma solução ótima para as instâncias pertencentes a classe $\mathcal{D}$ (casos 31 e 32). Tal fato se deve principalmente ao elevado número de variáveis

geradas. Todas as tentativas com esse propósito resultaram na exaustão de recursos de memória, onde não foi possível nem ao menos concluir a fase de geração das colunas para a construção do modelo. Todavia um limitante superior pôde ser encontrado por meio das metodologias propostas, as quais baseiam-se em meta-heurísticas, tais resultados podem ser observados através da Tabela 5.4.

5.2 Desempenho das metodologias

Para cada par instância/metodologia foram realizados 10 (dez) experimentos. Cada um dos quais considera como condição de parada 1 (uma) hora de processamento, além disso o experimento é finalizado caso alcance uma solução com o valor igual ao limitante (z^*), os quais foram produzidos por meio dos resultados da abordagem PM (ver Tabela 5.3).

Com respeito aos parâmetros utilizados para as abordagens baseadas em meta-heurísticas, para aquela que baseou-se em Busca Tabu, o comprimento máximo da lista foi fixado em 55. Na metodologia baseada em Iterated Local Search por sua vez foi utilizado nível de perturbação inicial $l_{\min} = 3$ e $l_{\max} = 20$. Verificou-se empiricamente por meio de sessões de exaustivos experimentos que estes valores forneciam bons resultados, e por isso foram os empregados.

Através da Tabela 5.5 são apresentados os seguintes dados estatísticos extraídos por meio dos experimentos realizados: distância média (*gap* médio) e a distância mais acentuada (*gap* máximo) em relação às melhores soluções conhecidas, percentual de casos onde foi alcançada a melhor solução conhecida para todos os dez experimentos ("melhor*"), percentual de casos onde foi alcançada a melhor solução conhecida em ao menos um dos experimentos ("melhor$^+$"), desvio médio (desvio médio) e o desvio mais acentuado (desvio máximo) em relação a solução produzida pelo melhor dos experimentos. Vale enfatizar que o limite superior apresentado na Tabela 5.4 é considerado no levantamento destas estatísticas.

	gap				desvio	
método	médio	máximo	melhor*	melhor$^+$	médio	máximo
BT	0,68	2,19	18,75	46,88	0,25	1,00
ILS *	0,35	1,69	28,12	68,75	0,23	1,00
MILS	0,37	1,69	21,88	68,75	0,28	1,62

Tabela 5.5: Resumo de desempenho

A meta-heurística ILS notavelmente se destaca dentre as demais implementadas, pois apresentou melhores resultados, produziu em média soluções a uma distância de apenas $0,35\%$ das melhores soluções conhecidas, alcançou a melhor solução conhecida em um maior percentual de casos, e finalmente por sua robustez observada através dos baixos desvios. Vale destacar ainda as soluções produzidas para as instâncias pertencentes a classe $\mathcal{C}$, pois nota-se que a metodologia que vem se destacando (ILS) foi capaz de produzir soluções a uma média de distância de $0,347\%$, em um tempo aproximadamente 3 (três) vezes menor que o tempo demandado pela abordagem PM, para se encontrar a solução ótima (ver Tabela 5.6).

Emprega-se (5.1) com o objetivo de mensurar o quanto de desvio ocorrera entre valor médio, valor da melhor solução encontrada e melhor solução conhecida independentemente.

$$\sigma(z, z') = \frac{z - z'}{z'} \times 100 \tag{5.1}$$

A Tabela 5.6 apresenta em detalhes os dados que deram origem a Tabela 5.5. Em ordem, as colunas contidas em tais tabelas representam respectivamente: sigla da metodologia, valor médio e melhor valor encontrado, desvio entre o valor médio e valor da melhor solução conhecida, tempo médio de processamento, tempo médio de processamento para se alcançar um valor alvo (sub-ótimo) que se encontra a uma distância (desvio) de 5% em relação a melhor solução conhecida. Essa última tem como por objetivo analisar o comportamento das metodologias quando lhes é dado um valor alvo de mínima qualidade. Os valores de z^* presentes nesta tabela são os mesmos da Tabela 5.3, novamente dispostos apenas para facilitar a comparação.

Observa-se que para quaisquer das instâncias de teste, as metodologias aqui implementadas, apresentaram em tempos quase desprezíveis, soluções a uma distância de não mais que 5% em relação a melhor solução conhecida, em quaisquer dos dez experimentos realizados.

instância bound(z^*)	método	z		desvio (%)		tempo médio (seg.)	
		média	melhor	melhor	gap (z^*)	parada	sub-ótimo
1 z^*=170	BT	170,00	170,00	0,000	0,000	0,20	0,000
	ILS	170,00	170,00	0,000	0,000	0,10	0,100
	MILS	170,00	170,00	0,000	0,000	0,00	0,000
2 z^*=442	BT	442,10	442,00	0,023	0,023	362,40	0,000
	ILS	442,00	442,00	0,000	0,000	5,20	0,000
	MILS	442,00	442,00	0,000	0,000	6,10	0,000
3 z^*=479	BT	479,00	479,00	0,000	0,000	0,40	0,000
	ILS	479,00	479,00	0,000	0,000	0,30	0,000
	MILS	479,00	479,00	0,000	0,000	0,60	0,200
4 z^*=984	BT	984,00	984,00	0,000	0,000	136,50	0,400
	ILS	984,00	984,00	0,000	0,000	295,30	0,300
	MILS	985,80	984,00	0,183	0,183	1231,80	3,200
5 z^*=1007	BT	1007,70	1007,00	0,070	0,070	2526,20	0,000
	ILS	1008,10	1007,00	0,109	0,109	2839,30	0,500
	MILS	1008,80	1007,00	0,179	0,179	3206,40	0,400
6 z^*=275	BT	275,80	275,00	0,291	0,291	1800,10	0,000
	ILS	275,10	275,00	0,036	0,036	360,30	0,000
	MILS	275,10	275,00	0,036	0,036	360,30	0,000
7 z^*=1234	BT	1241,60	1235,00	0,534	0,616	3600,00	2,000
	ILS	1235,90	1234,00	0,154	0,154	3077,70	1,400
	MILS	1236,10	1234,00	0,170	0,170	3227,10	1,400
8 z^*=398	BT	399,00	398,00	0,251	0,251	1440,60	0,100
	ILS	398,20	398,00	0,050	0,050	368,70	0,100
	MILS	398,00	398,00	0,000	0,000	11,60	0,000
9 z^*=1239	BT	1241,10	1239,00	0,169	0,169	2529,10	2,100
	ILS	1239,90	1239,00	0,073	0,073	1690,30	2,000
	MILS	1240,20	1239,00	0,097	0,097	2080,60	2,000
10 z^*=389	BT	389,00	389,00	0,000	0,000	0,00	0,000
	ILS	389,00	389,00	0,000	0,000	2,00	0,000
	MILS	389,00	389,00	0,000	0,000	11,90	0,000
11 z^*=668	BT	668,20	668,00	0,030	0,030	376,60	0,400
	ILS	668,00	668,00	0,000	0,000	49,30	0,100
	MILS	668,00	668,00	0,000	0,000	102,30	2,400
12 z^*=510	BT	515,10	510,00	1,000	1,000	3240,80	0,400
	ILS	510,00	510,00	0,000	0,000	4,50	1,400
	MILS	510,80	510,00	0,157	0,157	723,40	1,300
13 z^*=2266	BT	2277,10	2275,00	0,092	0,490	3600,00	19,700
	ILS	2281,80	2278,00	0,167	0,697	3614,30	19,000
	MILS	2280,30	2275,00	0,233	0,631	3609,20	17,200
14 z^*=1720	BT	1743,60	1743,00	0,034	1,372	3600,00	8,200
	ILS	1720,90	1720,00	0,052	0,052	2421,70	8,600
	MILS	1721,00	1720,00	0,058	0,058	2607,00	8,500
15 z^*=1625	BT	1654,00	1653,00	0,060	1,785	3600,00	16,200
	ILS	1650,30	1634,00	0,998	1,557	3605,60	9,800

instância bound(z^*)	método	z		desvio (%)		tempo médio (seg.)	
		média	melhor	melhor	gap (z^*)	parada	sub-ótimo
	MILS	1652,40	1626,00	1,624	1,686	3604,70	9,100
$\underset{z^*=2132}{16}$	BT	2171,10	2163,00	0,374	1,834	3600,00	27,000
	ILS	2156,00	2143,00	0,607	1,126	3609,50	14,900
	MILS	2153,00	2136,00	0,796	0,985	3610,80	14,400
$\underset{z^*=850}{17}$	BT	858,60	854,00	0,539	1,012	3600,00	1,100
	ILS	851,20	850,00	0,141	0,141	2400,80	13,400
	MILS	852,00	850,00	0,235	0,235	2394,00	5,900
$\underset{z^*=800}{18}$	BT	802,20	800,00	0,275	0,275	2663,00	3,400
	ILS	802,60	800,00	0,325	0,325	2684,80	4,300
	MILS	801,80	800,00	0,225	0,225	1945,80	5,400
$\underset{z^*=1057}{19}$	BT	1073,00	1065,00	0,751	1,514	3600,00	10,800
	ILS	1060,40	1057,00	0,322	0,322	3575,10	27,400
	MILS	1062,00	1058,00	0,378	0,473	3602,00	34,700
$\underset{z^*=1024}{20}$	BT	1037,90	1036,00	0,183	1,357	3600,00	3,800
	ILS	1031,70	1025,00	0,654	0,752	3601,60	2,600
	MILS	1031,20	1025,00	0,605	0,703	3601,80	2,100
$\underset{z^*=1135}{21}$	BT	1145,20	1143,00	0,192	0,899	3600,00	15,200
	ILS	1137,50	1136,00	0,132	0,220	3604,10	82,300
	MILS	1138,30	1135,00	0,291	0,291	3375,50	68,700
$\underset{z^*=257}{22}$	BT	257,00	257,00	0,000	0,000	11,80	0,000
	ILS	257,00	257,00	0,000	0,000	461,40	0,000
	MILS	257,00	257,00	0,000	0,000	632,10	1,100
$\underset{z^*=884}{23}$	BT	887,70	884,00	0,419	0,419	3278,70	1,000
	ILS	887,70	884,00	0,419	0,419	3256,20	0,500
	MILS	887,70	884,00	0,419	0,419	3439,40	5,400
$\underset{z^*=498}{24}$	BT	498,00	498,00	0,000	0,000	205,30	0,100
	ILS	498,00	498,00	0,000	0,000	531,10	0,100
	MILS	498,30	498,00	0,060	0,060	2093,30	2,900
$\underset{z^*=1193}{25}$	BT	1213,70	1206,00	0,638	1,735	3600,00	30,500
	ILS	1196,90	1193,00	0,327	0,327	3603,00	130,800
	MILS	1196,00	1193,00	0,251	0,251	2390,40	140,000
$\underset{z^*=2786}{26}$	BT	2842,40	2833,00	0,332	2,024	3601,20	90,800
	ILS	2833,20	2824,00	0,326	1,694	3616,90	58,600
	MILS	2833,10	2820,00	0,465	1,691	3623,80	57,000
$\underset{z^*=2669}{27}$	BT	2679,10	2676,00	0,116	0,378	3601,50	95,500
	ILS	2688,40	2678,00	0,388	0,727	3622,00	63,400
	MILS	2690,40	2681,00	0,351	0,802	3617,60	61,200
$\underset{z^*=1697}{28}$	BT	1703,40	1699,00	0,259	0,377	3600,20	24,800
	ILS	1709,40	1698,00	0,671	0,731	3609,10	14,900
	MILS	1710,90	1703,00	0,464	0,819	3606,30	14,500
$\underset{z^*=764}{29}$	BT	770,10	765,00	0,667	0,798	3600,00	4,800
	ILS	766,00	765,00	0,131	0,262	3600,60	1,000
	MILS	767,30	765,00	0,301	0,432	3600,80	1,000
$\underset{z^*=1205}{30}$	BT	1209,90	1208,00	0,157	0,407	3600,00	7,000
	ILS	1210,20	1206,00	0,348	0,432	3603,50	4,500
	MILS	1208,70	1206,00	0,224	0,307	3606,60	10,300
$\underset{\overline{z}=865}{31}$	BT	870,20	868,00	0,253	0,601	3600,00	12,500
	ILS	871,80	865,00	0,786	0,786	3605,40	42,800
	MILS	869,50	865,00	0,520	0,520	3607,20	49,600
$\underset{\overline{z}=1073}{32}$	BT	1096,50	1092,00	0,412	2,190	3600,00	22,800

instância bound(z^*)	método	z		desvio (%)		tempo médio (seg.)	
		média	melhor	melhor	gap (z^*)	parada	sub-ótimo
	ILS	1076,20	1073,00	0,298	0,298	3603,10	109,000
	MILS	1078,40	1073,00	0,503	0,503	3603,40	96,800

Tabela 5.6: Desempenho metodologias × instâncias

5.3 Gráfico das evoluções

As Figuras (5.1)–(5.4) apresentam graficamente a evolução da solução durante a execução do experimento que dentre o conjunto dos dez executados alcançou melhor resultado, para cada par instância/metodologia. Na legenda de cada um dos gráficos encontram-se em destaque a sigla das metodologias que para aquela instância alcançaram uma solução com o mesmo valor da melhor solução conhecida com respeito a função de avaliação, além disso mostra-se também o *gap* (distância) para os casos onde tal fato não ocorreu. A análise de tais figuras deixa ainda mais claro o quão próximas as soluções lá representadas encontram-se da melhor solução conhecida.

Através destas e dos resultados anteriormente apresentados e discutidos, verifica-se a eficiência e rapidez com que as metodologias aqui implementadas convergem em direção a valores muito próximos aos das melhores soluções conhecidas.

Através das figuras, observa-se ainda, que apesar da BT para muitos dos casos aqui considerados, evoluir mais rapidamente nos primeiros instantes de execução as versões que aplicam ILS apresentam um comportamento mais eficiente em um prazo ligeiramente maior, alcançando soluções de qualidade superior. O que leva a concluir que a ILS demonstra-se mais eficiente em escapar de mínimos locais.

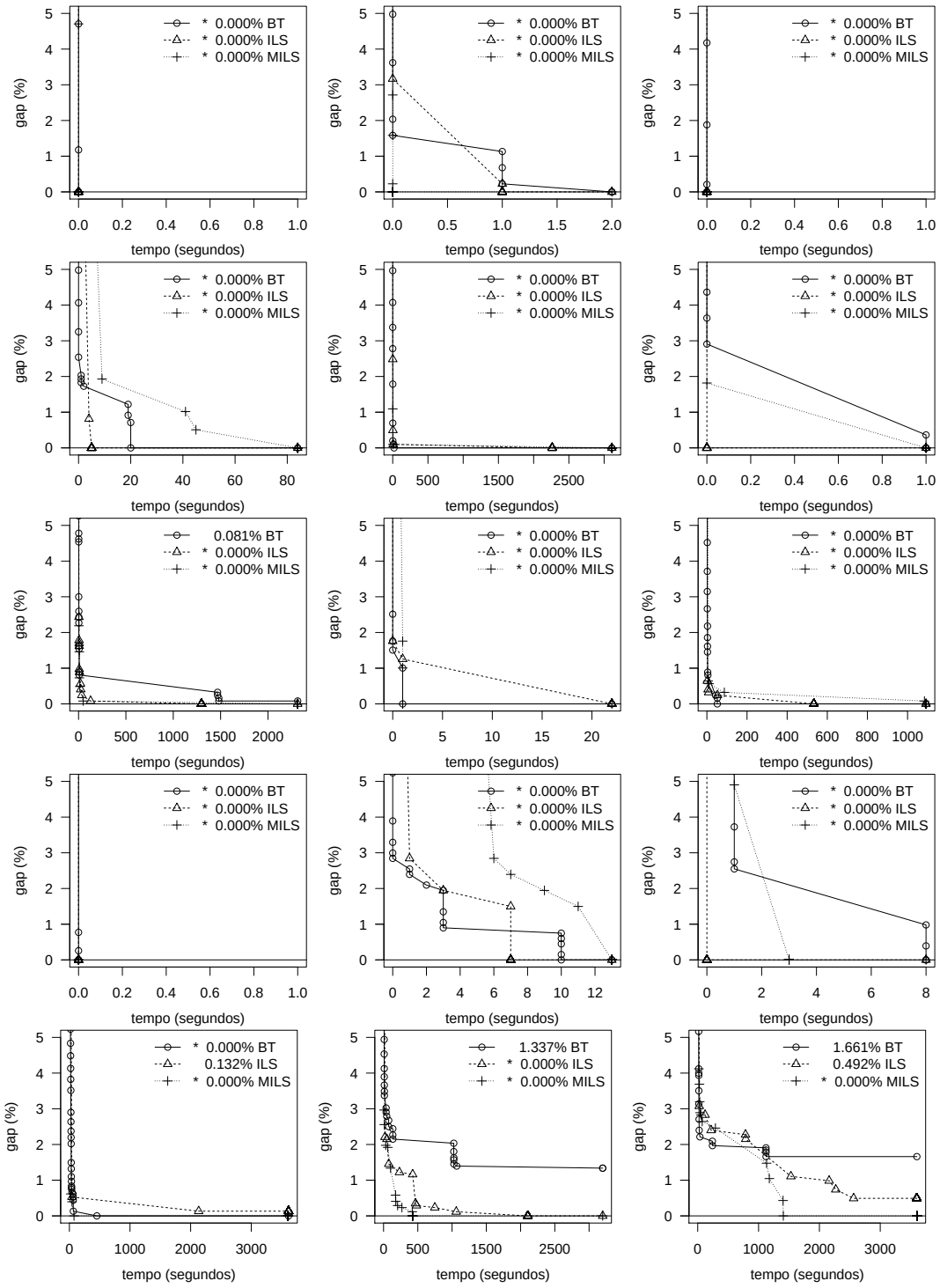


Figura 5.1: Evolução do melhor experimento em cada instância classe $\mathcal{A}$ × métodos

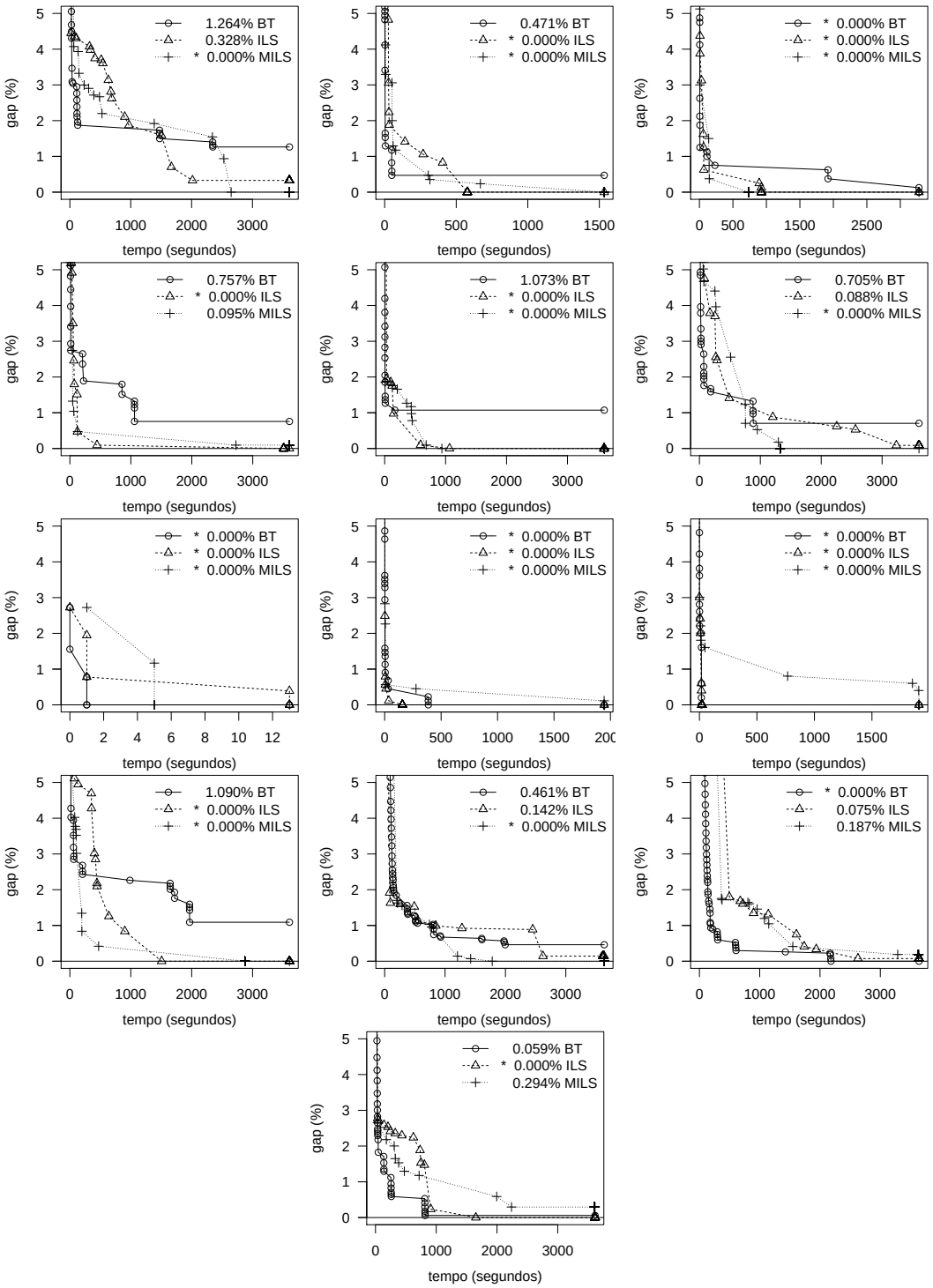


Figura 5.2: Evolução do melhor experimento em cada instância classe $\mathcal{B}$ × métodos

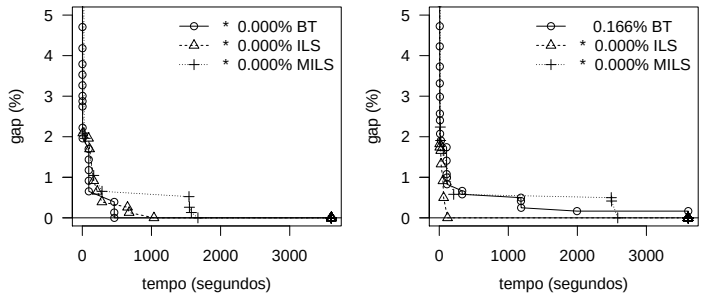


Figura 5.3: Evolução do melhor experimento em cada instância $\mathcal{C}$ × métodos

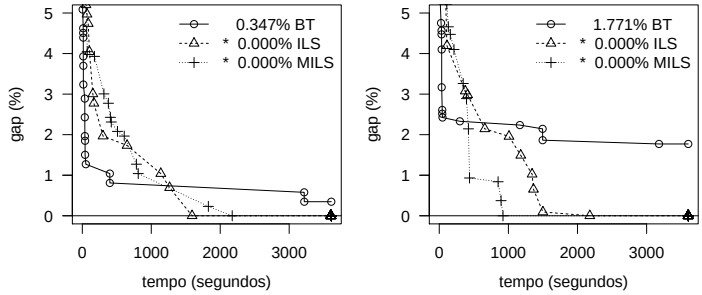


Figura 5.4: Evolução do melhor experimento em cada instância $\mathcal{D}$ × métodos

5.4 Análise de distribuição de probabilidade empírica

Com o propósito de se analisar a velocidade com que cada umas das metodologias produz soluções de mínima qualidade, neste momento leva-se em conta a distribuição empírica de probabilidade de se atingir um determinado valor alvo definido a priori, ou seja, considera-se o tempo necessário para produção de soluções com custo menor ou igual ao valor em questão. Os valores alvo considerados nos experimentos foram definidos de forma a permitir que até mesmo a mais lenta das metodologias alcance-o em um tempo computacionalmente razoável. Nesse experimento foram avaliadas as versões dos métodos ILS e BT que se destacaram considerando a análise realizada na seção anterior, sendo que os tempos de execução para 100 (cem) execuções em cada foram computados.

Para cada par instância/alvo, os dados que representam o tempo despendido para se alcançar o valor alvo em cada uma das execuções são ordenados em forma

crescente. Associa-se então o i-ésimo menor tempo de execução t_i com a probabilidade:

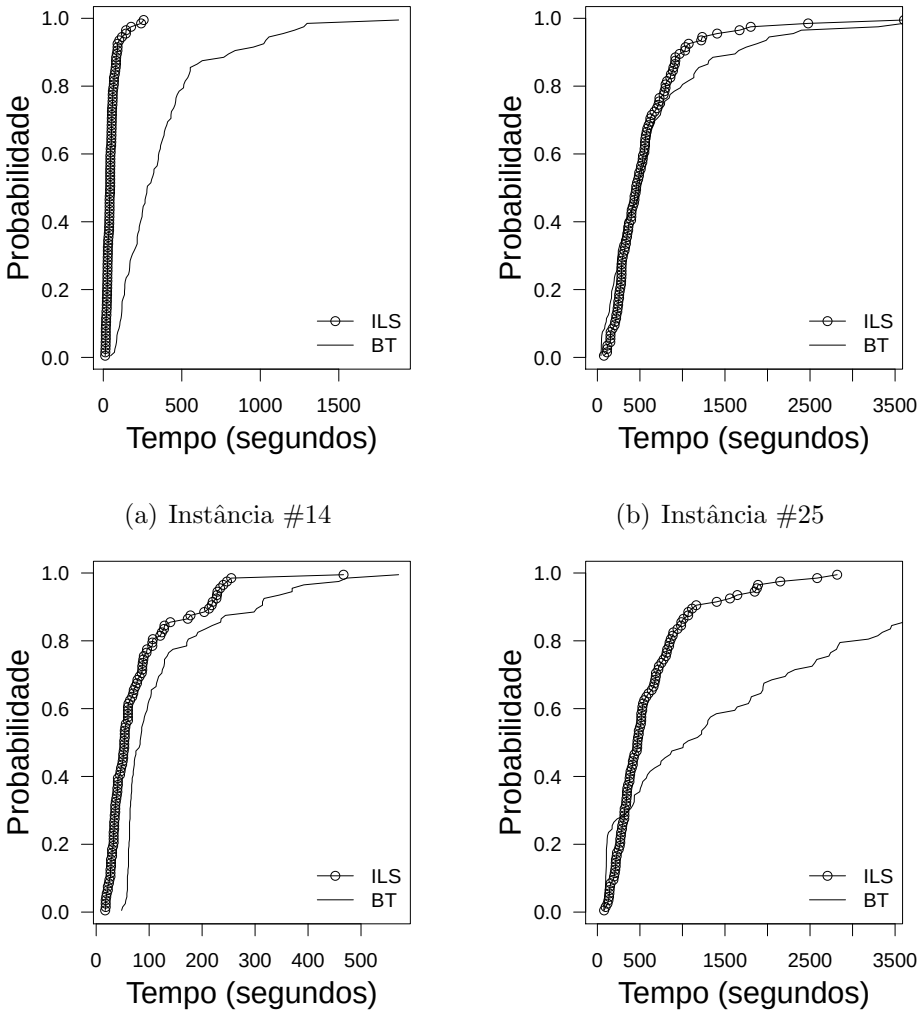
$$p_i = \frac{i - \frac{1}{2}}{n}, 1 \leq i \leq n,$$

gerando assim, os pontos:

$$z_i = (t_i, p_i), 1 \leq i \leq n,$$

onde n representa o número de experimentos realizados.

Esta metodologia de análise proposta em [70] baseia-se em [71]. Os resultados produzidos encontram-se graficamente representados através da Figura 5.5.



(a) Instância #14

(b) Instância #25

(c) Instância #30

(d) Instância #32

Figura 5.5: Distribuição empírica de probabilidade

Observa-se por meio dos resultados apresentados por meio da Figura 5.5, que em poucos instantes torna-se cada vez mais provável encontrar o valor alvo utilizando a metodologia baseada em ILS.

Além disso, nota-se também que a BT sinaliza com maior probabilidade no intervalo que representa os primeiros instantes, todavia em poucos instantes a ILS a ultrapassa.

Capítulo 6

Conclusões e Trabalhos futuros

O Problema de Programação de Tripulações (PPT) é um problema de grande importância na prática do planejamento das operações de empresas vinculadas ao setor transporte. Segundo relatos de trabalhos presentes na literatura como [7] e [35], algumas empresas desse setor ainda geram as escalas (jornadas) de trabalho de suas tripulações manualmente. Por conseguinte há uma grande necessidade de trabalhos como o presente, os quais auxiliem na difusão de técnicas como as que aqui foram aplicadas com o propósito de resolução deste problema.

Para a resolução do PPT, neste trabalho foram implementadas duas versões da meta-heurística *Iterated Local Search* e uma versão de Busca Tabu para efeitos de comparação. A seleção de tais, vem a ser justificada por manter um histórico de destaque e sucesso em abordagens direcionadas a este, e outros problemas de natureza combinatória e de difícil resolução, além disso pelo baixo número de parâmetros a serem harmonizados e também pela sua boa robustez, quando comparadas a outras meta-heurísticas presentes na literatura.

Também foi implementada uma versão de ILS , a qual incorpora um mecanismo que consiste em uma memória de curto prazo onde visa-se impedir que em uma mesma sessão de perturbações, uma perturbação realize um movimento reverso a outro que já havia sido realizado naquela sessão. Através dos resultados concluiu-se que para esta proposição, não houve benefício quando comparadas às que não o faziam, além disso o custo de gerenciamento da memória empregada mostrou-se oneroso.

Com respeito a harmonização do parâmetro de pertubação inicial foi possível

observar que, aquelas versões de ILS que empregam a ideia de pertubações iniciando-se do nível 3 (três) com o proposito de se evitar ciclos imediatos, mostraram-se eficientes apresentando resultados de alta qualidade.

Com o propósito de se mensurar de forma mais apurada a qualidade das soluções produzidas pelas metodologias aqui aplicadas, foi implementado um método exato (PM), onde então aplicou-se um método de *Enumeração Exaustiva* o qual para cada uma das instâncias, neste caso, foi incumbido de realizar independentemente a geração do conjunto de todas as possíveis colunas (enumeração completa), ou seja, foram consideradas todas as combinações de tarefas a serem cumpridas as quais implicassem em jornadas viáveis, e isto foi feito empregando-se um procedimento baseado em *Backtracking*. A cada um dos conjuntos de colunas aplicou-se então o modelo de particionamento de conjuntos, sendo este submetido a um pacote resolvedor de otimização. Concluiu-se então através dos resultados que as metodologias (meta-heurísticas) apresentavam desvios em média relativamente baixos com respeito ao valor da melhor solução conhecida para a grande maioria das instâncias.

O presente trabalho destaca-se por considerar instâncias geradas a partir de dados reais, as quais contemplam regras operacionais deste país. Destaca-se também por obter sucesso na validação de suas proposições, pois baseia-se como referência de comparação em uma metodologia exata, com a qual provou-se otimalidade para $93,75\%$ dos casos de teste aqui considerados, dentre os quais alguns destes, possuem dimensões relativamente elevadas, chegando a um número superior a 15 (quinze) milhões de variáveis.

Em suma, os resultados mostram de forma clara e veemente o potencial da abordagem aqui proposta, onde soluções de alta qualidade são obtidas, em tempos de processamento expressivamente baixos, para problemas (instâncias) reais, de dimensões relativamente grandes. Mostrando assim a grande eficácia das meta-heurísticas, em tratar de problemas tais como esse, onde encontram-se repletos de objetivos e regras operacionais.

Dentre outras possibilidades, como extensão desse trabalho pode-se propor o emprego de um método híbrido, o qual por exemplo poderia combinar a meta-heurística ILS a um método exato. Inspirado em [38], uma possível proposta poderia considerar como um tipo de pertubação para a ILS, a resolução de um sub-problema com

um número restrito de tarefas desalocadas de suas respectivas jornadas, este então seria submetido a um procedimento de programação matemática como o descrito na seção 4.9.

Referências Bibliográficas

[1] SANTOS, A. G., *Método de Geração de Colunas e Meta–heurísticas para Alocação de Tripulações*, Ph.D. Thesis, UFMG – Universidade Federal de Minas Gerais, 2008.

[2] BOUZADA, C. F., *Análise das despesas administrativas no custo do transporte a coletivo por ônibus no municipio de Belo Horizonte*, Master's Thesis, Escola de Governo, Fundação João Pinheiro, Belo Horizonte, 2002.

[3] CABRAL, L. D. A. F., FREITAS, M. J., MACULAN, N., et al., "An Heuristic Approach for Large Scale Crew Scheduling Problems at Rio-Sul Airlines", *40th International Symposium of the AGIFORS*, pp. 16, August 2000.

[4] ERNST, A. T., JIANG, H., KRISHNAMOORTHY, M., et al., "Staff scheduling and rostering: A review of applications, methods and models", *European Journal of Operational Research*, v. 153, n. 1, pp. 3–27, 2004, Timetabling and Rostering.

[5] WREN, A., *Scheduling Vehicles and their Drivers — Forty Years' Experience*, Tech. Rep. 2004.03, University of Leeds, School of Computing Research, April 2004.

[6] ELIAS, S. E. G., *The use of digital computers in the economic scheduling for both man and machine in public transportation*, Tech. rep., Manhattan: Kansas State University, Special Report 49, 1964.

[7] MAURI, G. R., *Novas Heurísticas para o Problema de Escalonamento de Tripulações*, Master's Thesis, INPE - Instituto Nacional de Pesquisas Espaciais, 2005.

[8] MANINGTON, P. D., WREN, A., "Experiences with a bus scheduling algorithm which saves vehicles". In: *Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services*, Chicago, 1975.

[9] WREN, A., ROUSSEAU, J. M., "Bus driver scheduling: an overview". In: *Daduna, J.R.; Branco, I.; Paixão, J. M. P. (Eds.), Computer-Aided Scheduling of Public Transport, Lecture Notes in Economics and Mathematical Systems*, pp. 173–187, 1995.

[10] MINGOZZI, A., BOSCHETTI, M. A., RICCIARDE, S., et al., "A Set Partitioning Approach to the Crew Scheduling Problem", *Oper. Res.*, v. 47, n. 6, pp. 873–888, 1999.

[11] GAREY, M. R., JOHNSON, D. S., *Computers and Intractability : A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. San Francisco, W. H. Freeman., January 1979.

[12] GOLDBERG, D. E., *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston, MA, USA, Addison-Wesley Longman Publishing Co., Inc., 1989.

[13] GLOVER, F., "Future paths for integer programming and artificial intelligence", *Computers and Operations Research, Amsterdam*, v. 13, pp. 533–549, 1986.

[14] FEO, T., RESENDE, M., "A probabilistic heuristic for a computationally difficult set covering problem", *Operations Research Letters*, v. 8, pp. 67–71, 1989.

[15] KIRKPATRICK, S., GELATT, C. D., VECCHI, M. P., "Optimization by simulated annealing", *Science*, v. 220, pp. 671–680, 1983.

[16] GLOVER, F. W., KOCHENBERGER, G. A., *Handbook of Metaheuristics (International Series in Operations Research & Management Science)*. January 2003.

[17] LOURENÇO, H. R., PAIXÃO, J. P., PORTUGAL, R., "Multiobjective Metaheuristics for the Bus Driver Scheduling Problem", *Transportation Science*, v. 35, n. 3, pp. 331–343, 2001.

[18] FRELING, R., *Models and Techniques for Integrating Vehicle and Crew Scheduling*, Ph.D. Thesis, Erasmus University, Rotterdam, The Netherlands, 1997.

[19] LOURENÇO, H. R., PINTO, J., PORTUGAL, R., *Metaheuristics for The Bus-Driver Scheduling Problem*, Economics Working Papers 304, Department of Economics and Business, Universitat Pompeu Fabra, July 1998.

[20] CEDER, A., "Urban transit scheduling: framework, review and examples", *Journal of Urban Planning and Development*, v. 128(4), pp. 225–244, 2002.

[21] LEUNG, Y.-T. J., *Handbook of Scheduling: Algorithms, Models and Performance Analysis*. Boca Raton, Florida, Chapman & Hall/CRC, 2004.

[22] BASSI, H. V., SILVA, G. P., MARINHO, E. H., et al., "GRASP and Path-relinking Applied to the Independent Crew Scheduling Problem. In: 10th International Conference on Computers in Urban Planning and Urban Management", *10th International Conference on Computers in Urban Planning and Urban Management, Reviewed papers in the Conference Proceedings of CUPUM*, 2007.

[23] SOUZA, M. J. F., SILVA, G. P., SIMÕES, E. M. L., "Programação de Ônibus Urbano: uma Abordagem Heurística". In: *Confederação Nacional do Transporte (CNT), Associação Nacional de Pesquisa e Ensino em Transporte (ANPET)*, v. 1, n. 1, pp. 39–57, Brasília, 2007.

[24] SIMÕES, E. M. L., SOUZA, M. J. F., SILVA, G. P., "Aplicação da Metaheurística Iterated Local Search à Programação de Veículos no Sistema de Transporte Público". In: *XXXVIII Simpósio Brasileiro de Pesquisa Operacional*, Goiânia, 2006.

[25] FORSYTH, P., WREN, A., "An Ant System for Bus Driver Scheduling". In: *7th International Workshop on Computer-Aided Scheduling of Public Transport*, Boston, August 1997.

[26] WREN, A., FORES, S., KWAN, A., et al., "A flexible system for scheduling drivers", *J. of Scheduling*, v. 6, n. 5, pp. 437–455, 2003.

[27] BEASLEY, J. E., CAO, B., "A tree search algorithm for the crew scheduling problem", *European Journal of Operational Research*, v. 94, n. 3, pp. 517–526, 1996.

[28] DESAULNIERS, G., "Bus and Driver Scheduling in Urban Mass Transit Systems". In: *Travel and Transportation Workshop, Institute for Mathematics and its Applications*, Minneapolis, University of Minnesota, 2002.

[29] SHEN, Y., KWAN, R. S. K., "Tabu search for driver scheduling". In: *CASPT*, 2001.

[30] DESROCHERS, M., SOUMIS, F., "A Column Generation Approach to the Urban Transit Crew Scheduling Problem", *TRANSPORTATION SCIENCE*, v. 23, n. 1, pp. 1–13, 1989.

[31] DALLAIRE, A., FLEURENT, C., ROUSSEAU, J.-M., "Dynamic Constraint Generation in CrewOpt, a Column Generation Approach for Transit Crew Scheduling". In: *COMPUTER-AIDED SCHEDULING OF PUBLIC TRANSPORT (CASPT)*, pp. 73–90, 2004.

[32] MAURI, G. R., LORENA, L. A. N., "A New Hybrid Heuristic for Driver Scheduling", *International Journal of Hybrid Intelligent Systems, IOS Press - Amsterdam*, v. 4 (1), pp. 39–47, 2007.

[33] MAURI, G. R., *Resolução do Problema de Programação de Tripulações de um Sistema de Transporte Público via Simulated Annealing*, Tech. rep., Departamento de Ciência da Computação - Universidade Federal de Ouro Preto, 2003.

[34] VALOUXIS, C., HOUSOS, E., "Combined bus and driver scheduling", *Computers and Operation Research Journal*, v. 29, n. 3, pp. 243–259, 2002.

[35] MARINHO, E. H., *Heurísticas Busca Tabu para o Problema de Programação de Tripulações de Ônibus Urbano*, Master's Thesis, UFF - Universidade Federal Fluminense, 2005.

[36] MOREIRA, J. M., *Travel time prediction for the planning of mass transit companies: a machine learning approach*, Ph.D. Thesis, Faculty of Engineering of University of Porto, 2008.

[37] SILVA, G. P., SOUZA, M. J. F., ALVES, J. M. D. C. B., "Simulated Annealing Aplicado à Programação da Tripulação no Sistema de Transporte Público". In: *XXII Encontro Nacional de Engenharia de Produção*, 2002.

[38] GONÇALVES, T. L., FAMPA, M. H. C., DOS SANTOS, A. G., et al., "Metaheurística Busca Tabu e Programação Matemática Uma Abordagem Híbrida Aplicada ao Problema de Programação de Tripulações". In: *Anais XXXI CNMAC - Congresso Nacional de Matemática Aplicada e Computacional*, 2008.

[39] GONÇALVES, T. L., SILVA, J. M. N., DOS SANTOS, A. G., "Estratégia Paralela para Busca Tabu Aplicada ao Problema de Programação de Tripulações". In: *Anais XIV CLAIO – Latin Ibero–American Congress on Operations Research*, Cartagena de Índias, Setembro 2008.

[40] MARINHO, E. H., OCHI, L. S., DRUMMOND, L. M. A., et al., "Busca Tabu Aplicada ao Problema de Programação de Tripulações de Ônibus Urbano", *XXXVI SBPO - Sociedade Brasileira de Pesquisa Operacional*, 2004.

[41] CALVI, R., *Um Algoritmo para o Problema de Escalonamento de Tripulação em Empresas de Ônibus*, Master's Thesis, Universidade Estadual de Maringá, 2005.

[42] GONÇALVES, T. L., *Estudo e implementação de algoritmos multiobjetivos para o problema de alocação de tripulação*, Iniciação científica, CNPq – Conselho Nacional de Desenvolvimento Científico e Tecnológico, 2006.

[43] DIAS, M. T. G., *A new approach to the bus driver scheduling problem using multiobjective genetic algorithms*, Ph.D. Thesis, Faculty of Engineering of University of Porto, 2005.

[44] FORES, S., PROLL, L., WREN, A., "An Improved ILP System for Driver Scheduling", *Wilson, N. H. M. (ed.) Computer-Aided Transit Scheduling*, pp. 43–61, 1999.

[45] SMITH, B. M., WREN, A., "A bus crew scheduling system using a set covering formulation", *Transportation Research Part A: General*, v. 22, n. 2, pp. 97–108, 1988.

[46] BARNHART, C., JOHNSON, E. L., NEMHAUSER, G. L., et al., "Branch-and-Price: Column Generation for Solving Huge Integer Programs", *OPERATIONS RESEARCH*, v. 46, n. 3, pp. 316–329, 1998.

[47] HAASE, K., FRIBERG, C., "An exact branch and cut algorithm for the vehicle and crew scheduling problem", *N. H. M. Wilson, ed. Computer-Aided Transit Scheduling. Springer Verlag, Berlin, Germany*, pp. 63–80, 1999.

[48] SOUZA, M. J. F., CARDOSO, L. X. T., SILVA, G. P., et al., "Metaheurísticas Aplicadas ao Problema de Programação de Tripulações no Sistema de Transporte Público", *Tendências em Matemática Aplicada e Computacional (TEMA)*, v. 5, n. 2, pp. 357–368, 2004.

[49] CARVALHO, M. A. M. D., SANTOS, A. G. D., MATEUS, G. R., "Seleção de Colunas no Problema de Escalonamento de Tripulações utilizando Algoritmo Genético". In: *XXXVII Simpósio Brasileiro de Pesquisa Operacional*, pp. 2512–2519, Gramado, RS, 2005.

[50] SOUZA, M. J. F., CARDOSO, L. X. T., SILVA, G. P., "Programação de Tripulações de Ônibus Urbano uma Abordagem Heurística", *XXXV SBPO - Sociedade Brasileira de Pesquisa Operacional*, 2003.

[51] SANTOS, A. G., MATEUS, G. R., "Crew scheduling urban problem: an exact column generation approach improved by a genetic algorithm". In: *IEEE Congress on Evolutionary Computation*, pp. 1725–1731, 2007.

[52] WEDELIN, D., "The design of a 0-1 integer optimizer and its application in the Carmen system", *European Journal of Operational Research*, v. 87, pp. 722–730, 1995.

[53] KOHL, N., KARISCH, S. E., "Airline Crew Rostering: Problem Types, Modeling, and Optimization", *Annals of Operations Research 127, 223-257, 2004*, v. 127, pp. 223–257, 2004.

[54] DESROCHERS, M.; GILBERT, J. S. M. S. F., "CREW-OPT : Subproblem Modeling in a Column Generation Approach to Urban Crew Scheduling", *Computer-Aided Transit Scheduling. Lecture Notes in Economics and Mathematical System*, v. 386, pp. 395–406, 1992.

[55] WILHELM, E. B., "Overview of the RUCUS package driver run cutting program (RUNS)". In: *Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services*, Chicago, 1975.

[56] DADUNA, J. R.; MOJSILOVIC, M., "Computer-aided vehicle and duty scheduling using HOT programme system", *Daduna, J. R.; Wren, A. (eds.) Computer-aided transit scheduling. Berlin: Springer-Verlag*, pp. 133–146, 1988.

[57] ROUSSEAU, J. M.; LESSARD, R. B. J. Y., "Enhancements to the HASTUS crew scheduling algorithm", *Computer scheduling of public transport, New York*, pp. 295–310, 1985.

[58] ERNST, A. T., JIANG, H., KRISHNAMOORTHY, M., et al., "An Annotated Bibliography of Personnel Scheduling and Rostering." *Annals OR*, v. 127, n. 1-4, pp. 21–144, 2004.

[59] ROSEN, K. H., MICHAELS, J. G., *Handbook of discrete and combinatorial mathematics*. Boca Raton, FL, CRC Press, 2000.

[60] DE WERRA, D., HERTZ, A., "Tabu search techniques: a tutorial and an application to neural networks", *OR Spektrum*, v. 11, pp. 131—141, 1989.

[61] HERTZ, A., DE WERRA, D., "The tabu search metaheuristic: How we used it", *Annals of Mathematics and Artificial Intelligence*, v. 1, n. 1, pp. 111–121, Sept. 1990.

[62] FAIGLE, U., K, W., "Some Convergence Results for Probabilistic Tabu Search", *ORSA Journal on Computing*, pp. 4:32–37, 1992.

[63] FOX, B. L., "Integrating and accelerating tabu search, simulated annealing, and genetic algorithms", *Annals of Operations Research*, v. 41, n. 1-4, pp. 47–67, 1993.

[64] PIGATTI, A. A., *Modelos e Algoritmos para o Problema de Alocação Generalizada (PAG) e Aplicações*, Master's Thesis, PUC-Rio, 2003.

[65] SANTOS, A. G., MATEUS, G. R., "Hybrid approach to solve a crew scheduling problem: an exact column generation algorithm improved by metaheuristics". In: *Hybrid Intelligent Systems, 2007. HIS 2007. 7th International Conference on*, pp. 107–112, Sept. 2007.

[66] DRUMOND, R. A., BOUZADA, C. F., "Novo Modelo de Gestão e Remuneração das Subconcessionárias do Serviço Público de Transporte Coletivo por Ônibus do Município de Belo Horizonte". In: *XVIII Congresso de Pesquisa e Ensino em Tranportes*, 2004.

[67] VON ATZINGEN, J., DA CUNHA, C. B., SILVA, G. P., "Uma Abordagem Integrada para o Problema de Programação de Veículos e Tripulantes de Ônibus". In: *XXI ANPET - Congresso de Pesquisa e Ensino em Transportes*, pp. 1–12, 2007.

[68] CLÍMACO, J. N., ANTUNES, C. H., ALVES, M. J. G., *Programação Linear Multiobjetivo*. Portugal, Imprensa da Universidade de Coimbra, 2003.

[69] MLADENOVIC, N., HANSEN, P., "Variable neighborhood search", *Computers & Operations Research*, v. 24, n. 11, pp. 1097–1100, 1997.

[70] AIEX, R. M., RESENDE, M. G. C., RIBEIRO, C. C., "Probability distribution of solution time in GRASP: An experimental investigation", *Journal of Heuristics*, v. 8, pp. 200–2, 2000.

[71] CHAMBERS, J. M., *Graphical Methods for Data Analysis (Statistics)*. Boca Raton, Florida, Chapman & Hall/CRC, February 1983.

Apêndice A

Propriedades dos Melhores Experimentos

Neste apêndice apresenta-se através da Tabela A.1, as principais propriedades das soluções produzidas pelo melhor experimento para cada par instância/método. Devido ao baixo desvio, vale a analise de tais propriedades. Observa-se que as propriedades que referem-se a tempo (hora extra e tempo ocioso) encontram-se em "minutos" como unidade de medida. O tempo t de processamento em "segundos", como nos Capítulos anteriores. Além disso encontram-se em destaque ("**negrito**") aqueles onde obteve-se uma solução com o mesmo valor da melhor solução conhecida.

método	z	tempo(t)	tempo ocioso	hora extra	troca			dupla pegada	jornadas
					veículo	linha	ponto		
BT	**170**	**0**	**1059**	**32**	**1**	**0**	**0**	**1**	**12**
ILS	**170**	**0**	**1059**	**32**	**1**	**0**	**0**	**1**	**12**
MILS	**170**	**0**	**1059**	**32**	**1**	**0**	**0**	**1**	**12**
BT	**442**	**2**	**2789**	**331**	**1**	**0**	**0**	**3**	**28**
ILS	**442**	**1**	**2789**	**331**	**1**	**0**	**0**	**3**	**28**
MILS	**442**	**0**	**2789**	**331**	**1**	**0**	**0**	**3**	**28**
BT	**479**	**0**	**3051**	**416**	**1**	**0**	**0**	**3**	**29**
ILS	**479**	**0**	**3051**	**416**	**1**	**0**	**0**	**3**	**29**
MILS	**479**	**0**	**3051**	**416**	**1**	**0**	**0**	**3**	**29**
BT	**984**	**20**	**5174**	**1002**	**3**	**0**	**0**	**7**	**60**
ILS	**984**	**5**	**5174**	**1002**	**3**	**0**	**0**	**7**	**60**
MILS	**984**	**84**	**5174**	**1002**	**3**	**0**	**0**	**7**	**60**
BT	**1007**	**19**	**6417**	**908**	**2**	**0**	**0**	**6**	**63**
ILS	**1007**	**2263**	**6417**	**908**	**2**	**0**	**0**	**6**	**63**
MILS	**1007**	**3110**	**6417**	**908**	**2**	**0**	**0**	**6**	**63**
BT	**275**	**1**	**2229**	**98**	**0**	**0**	**0**	**2**	**19**
ILS	**275**	**0**	**2206**	**75**	**0**	**0**	**0**	**2**	**19**
MILS	**275**	**1**	**2219**	**88**	**0**	**0**	**0**	**2**	**19**
BT	1235	3600	8764	774	5	0	0	10	73
ILS	**1234**	**1298**	**8703**	**713**	**5**	**0**	**0**	**10**	**73**
MILS	**1234**	**2314**	**8703**	**713**	**5**	**0**	**0**	**10**	**73**

método	z	tempo(t)	tempo ocioso	hora extra	troca			dupla pegada	jornadas
					veículo	linha	ponto		
BT	**398**	**1**	**2831**	**255**	**0**	**0**	**0**	**3**	**30**
ILS	**398**	**22**	**2831**	**255**	**0**	**0**	**0**	**3**	**30**
MILS	**398**	**1**	**2831**	**255**	**0**	**0**	**0**	**3**	**30**
BT	**1239**	**52**	**9074**	**605**	**5**	**0**	**0**	**10**	**76**
ILS	**1239**	**533**	**9497**	**598**	**4**	**0**	**0**	**10**	**77**
MILS	**1239**	**1092**	**9074**	**605**	**5**	**0**	**0**	**10**	**76**
BT	**389**	**0**	**2898**	**143**	**1**	**0**	**0**	**2**	**25**
ILS	**389**	**0**	**2898**	**143**	**1**	**0**	**0**	**2**	**25**
MILS	**389**	**0**	**2898**	**143**	**1**	**0**	**0**	**2**	**25**
BT	**668**	**10**	**5136**	**410**	**1**	**0**	**0**	**5**	**44**
ILS	**668**	**7**	**5136**	**410**	**1**	**0**	**0**	**5**	**44**
MILS	**668**	**13**	**5136**	**410**	**1**	**0**	**0**	**5**	**44**
BT	**510**	**8**	**3234**	**414**	**0**	**0**	**0**	**3**	**36**
ILS	**510**	**0**	**3234**	**414**	**0**	**0**	**0**	**3**	**36**
MILS	**510**	**3**	**3234**	**414**	**0**	**0**	**0**	**3**	**36**
BT	2275	3600	18395	1289	4	1	0	16	140
ILS	2278	3615	17864	1188	6	4	2	16	139
MILS	2275	3604	18294	1188	5	3	1	16	140
BT	1743	3600	12045	1333	6	0	2	12	104
ILS	**1720**	**2106**	**10371**	**1379**	**9**	**0**	**3**	**12**	**100**
MILS	**1720**	**422**	**10371**	**1379**	**9**	**0**	**3**	**12**	**100**
BT	1653	3600	12091	1029	6	0	1	13	102
ILS	1634	3600	11755	1123	4	1	1	13	101
MILS	1626	3602	12447	955	4	0	1	13	103
BT	2163	3600	15572	1496	8	1	1	15	127
ILS	2143	3606	15250	1604	7	1	1	15	126
MILS	2136	3608	16283	1347	7	1	1	15	129
BT	854	3600	6944	324	2	0	0	5	56
ILS	**850**	**578**	**6529**	**339**	**3**	**0**	**0**	**5**	**55**
MILS	**850**	**1534**	**6529**	**339**	**3**	**0**	**0**	**5**	**55**
BT	**800**	**3279**	**5784**	**503**	**0**	**1**	**0**	**6**	**52**
ILS	**800**	**924**	**5784**	**503**	**0**	**1**	**0**	**6**	**52**
MILS	**800**	**738**	**5786**	**505**	**0**	**1**	**0**	**6**	**52**
BT	1065	3600	6612	611	3	0	2	7	69
ILS	**1057**	**3507**	**7045**	**614**	**2**	**0**	**1**	**7**	**70**
MILS	1058	3602	7038	607	2	0	2	7	70
BT	1036	3600	8453	626	0	0	0	6	65
ILS	1025	3601	8308	481	1	0	0	6	65
MILS	1025	3603	8308	481	1	0	0	6	65
BT	1143	3600	7783	371	4	0	1	8	78
ILS	1136	3606	7771	359	4	0	0	8	78
MILS	**1135**	**1325**	**8084**	**242**	**4**	**0**	**0**	**8**	**79**
BT	**257**	**1**	**2338**	**31**	**0**	**0**	**0**	**1**	**14**
ILS	**257**	**13**	**2338**	**31**	**0**	**0**	**0**	**1**	**14**
MILS	**257**	**5**	**2338**	**31**	**0**	**0**	**0**	**1**	**14**
BT	**884**	**387**	**7048**	**528**	**1**	**0**	**0**	**6**	**53**
ILS	**884**	**157**	**7048**	**528**	**1**	**0**	**0**	**6**	**53**
MILS	**884**	**1945**	**7048**	**528**	**1**	**0**	**0**	**6**	**53**
BT	**498**	**17**	**3119**	**347**	**1**	**0**	**0**	**3**	**28**
ILS	**498**	**22**	**3127**	**355**	**1**	**0**	**0**	**3**	**28**
MILS	**498**	**1909**	**3119**	**347**	**1**	**0**	**0**	**3**	**28**
BT	1206	3600	8886	646	1	2	2	8	76
ILS	**1193**	**3606**	**8865**	**625**	**0**	**2**	**2**	**8**	**76**
MILS	**1193**	**2874**	**8865**	**625**	**0**	**2**	**2**	**8**	**76**
BT	2833	3602	24778	1440	4	0	2	18	173
ILS	2824	3613	24306	1398	5	0	3	18	172
MILS	2820	3636	24659	1321	5	0	3	18	173
BT	2676	3603	21848	1500	6	3	2	18	162
ILS	2678	3642	20872	1384	8	4	4	18	160
MILS	2681	3632	20872	1384	8	4	3	18	160
BT	1699	3601	12186	1056	5	0	1	11	105
ILS	1698	3614	11745	1045	6	0	1	11	104
MILS	1703	3606	11724	1024	5	2	3	11	104

método	z	tempo(t)	tempo ocioso	hora extra	troca			dupla pegada	jornadas
					veículo	linha	ponto		
BT	765	3600	6665	322	0	0	0	4	47
ILS	765	3601	6666	323	0	0	0	4	47
MILS	765	3600	6666	323	0	0	0	4	47
BT	1208	3600	8019	808	3	1	1	8	75
ILS	1206	3603	9186	685	1	1	1	8	78
MILS	1206	3606	8843	772	1	1	1	8	77
BT	868	3600	6267	378	3	0	0	6	55
ILS	**865**	**3604**	**6693**	**374**	**2**	**0**	**0**	**6**	**56**
MILS	**865**	**3611**	**6694**	**375**	**2**	**0**	**0**	**6**	**56**
BT	1092	3600	9740	213	0	2	2	6	71
ILS	**1073**	**3601**	**8918**	**251**	**0**	**2**	**2**	**6**	**69**
MILS	**1073**	**3602**	**8918**	**251**	**0**	**2**	**2**	**6**	**69**