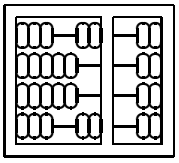

Universidade Estadual de Campinas - UNICAMP

Instituto de Computação - IC

Busca Tabu Paralelizada Aplicada ao Problema da Programação de Viagens

Segundo Relatório Científico

Tony Minoru Tamura Lopes — Orientando

Prof. Dr. Arnaldo Vieira Moura — Orientador

Julho de 2005

Resumo

Este projeto de Iniciação Científica trata da utilização da Metaheurística Busca Tabu Paralelizada aplicada ao Problema da Programação de Viagens. Esse problema é enfrentado pelas empresas de transporte urbano em seu planejamento diário, é de extrema importância financeira, e ainda afeta diretamente a qualidade do serviço. Para abordá-lo, foi construído um algoritmo *Multithread* com controle centralizado, troca de soluções de elite e aproveitamento de informação por recombinação. O algoritmo usou modelagem avançada com "Path Relinking" e oscilação estratégica. Já existe uma interface pronta para visualização ou edição das soluções que, em conjunto com o resolvedor, fornecerá uma ferramenta útil para as empresas.

Palavras-chave: Otimização, Busca Tabu, Paralelismo, Programação de Viagens, Transporte Urbano.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

1 Introdução

O presente projeto de iniciação científica trata do *Problema de Programação de Viagens*, ou *PPV*, aplicando a Metaheurística Busca Tabu Paralelizada. Esse problema é enfrentado por empresas do transporte urbano durante o planejamento das linhas, e boas soluções são de suma importância pois os ganhos delas advindos poderão ser redirecionados para a melhoria do serviço prestado. Sendo um problema de grande dificuldade, classificado como NP-difícil, a aplicação de uma Metaheurística, como a Busca Tabu Paralelizada, foi escolhida como tema deste projeto.

Na primeira fase do projeto, foram realizadas a modelagem do problema e a implementação do algoritmo básico monoprocessado que serviu de base para o algoritmo paralelizado. Em conjunto com este algoritmo, foram desenvolvidos também uma outra heurística, baseada na modelagem, e um algoritmo de busca local, com a finalidade de se realizar comparações e testes. Após a análise de resultados intermediários, foi implementada, como melhoria, uma oscilação estratégica, a qual conseguiu mais qualidade nas soluções obtidas quando comparadas às soluções encontradas pela Busca Tabu tradicional.

Nesta segunda fase do projeto, foram concluídos os estudos sobre algoritmos de Busca Tabu paralelizados. Seguindo uma taxonomia estudada, foi possível definir um modelo de paralelização sobre o algoritmo monoprocessado desenvolvido na primeira fase. No entanto, para a implementação do algoritmo paralelo, foi utilizada a versão sem oscilação estratégica. Isso porque a oscilação, agora, irá se aproveitar do paralelismo, como será descrito no projeto.

A estratégia utilizada nesta paralelização foi a de múltiplas threads que cooperam entre si trocando soluções e recombinando-as. Para tal, foram desenvolvidas as implementações de um "Elite Pool" e de um algoritmo de recombinação, no caso um "Path Relinking". Analisando os resultados obtidos verificou-se a necessidade de uma pós-otimização na distribuição dos horários de partida. Neste sentido, foi implementada uma heurística específica para distribuição de horários e uma adaptação do procedimento de busca local. Por último, foram realizados testes e comparações, que mostraram que a Busca Tabu paralelizada obteve melhores resultados do que sua implementação monoprocessada.

Este relatório será, doravante, organizado da seguinte forma: A seção 2 apresenta o cronograma de trabalho enviado à FAPESP e a descrição das atividades. Na seção 3 é descrito o problema abordado e é apresentada uma formulação para o mesmo. Da seção 4 à seção 10 descrevem-se os trabalhos de modelagem e implementação realizados durante o primeiro semestre de atividades. A seção 11 abrange o segundo semestre, onde a paralelização do algoritmo é descrita. Por fim, a seção 12 demonstra como a distribuição dos horários foi tratada e a seção 13 apresenta os testes e os resultados obtidos para instâncias reais. Na última seção, são apresentadas uma conclusão e uma perspectiva para trabalhos futuros.

2 Cronograma de Trabalho

Nesta segunda fase do projeto, trabalhou-se intensivamente na implementação da versão paralela da Busca Tabu, incluindo o "Elite Pool" e o componente de "Path Relinking" e foi dada atenção à

distribuição dos horários de viagens. Também foram iniciados os esforços em publicações como será descrito abaixo. Desta forma, cumpriu-se integralmente as atividades propostas no cronograma.

Abaixo segue o cronograma atualizado, e logo em seguida a descrição de cada atividade.

Atividades Planejadas		Situação*	**MÊS**											
			01	02	03	04	05	06	07	08	09	10	11	12
1	Estudos PPV	$\surd\surd$	•											
2	Estudos BT	$\surd\surd$	•	•	•									
3	Implementação BT	$\surd\surd$			•	•	•							
4	Testes	$\surd\surd$				•	•	•						
5	Melhorias no BT	$\surd\surd$					•	•	•	•	•	•	•	
6	Testes	$\surd\surd$					•	•	•	•	•	•	•	
7	Implement. Paralelismo	$\surd\surd$						•	•	•				
8	Testes	$\surd\surd$							•	•	•	•		
9	Melhorias Paralelismo	$\surd\surd$								•	•	•		
10	Testes	$\surd\surd$									•	•	•	
11	Relatórios	$\surd\surd$						•						•
12	Publicações	$\surd$											•	•

* $\surd$ = atividade iniciada. $\surd\surd$ = atividade concluída.

Tabela 1: Cronograma de atividades

1. *Estudos PPV:* Foram analisadas as especificidades do problema, as restrições a serem tratadas e as técnicas de modelagem empregadas para resolvê-lo [Vaz03, Rod01, WG97].

2. *Estudos BT:* Finalizaram-se os estudos da Metaheurística Busca Tabu [GL97, HTW92, LAG94, GP02] e também foram realizados estudos em Algoritmos de Busca Tabu aplicados a problemas de agendamento de veículos e motoristas, similares ao problema tratato [LPP01]. Isto com o propósito de conhecer as técnicas para modelagem de soluções, as vizinhanças utilizadas e as estratégias próprias da Busca Tabu. Como suplemento da próxima atividade, foram estudados Frameworks de Busca Local e Busca Tabu, e também artigos de modelagem orientada a objetos para Metaheurísticas [GS01, ACR98].

3. *Implementação da BT:* Baseando-se nos estudos realizados, optou-se por implementar uma enumeração de jornadas. Em seguida, a solução foi modelada em conjunto com outro aluno de iniciação científica, visando facilitar futuras hibridizações. Após o término da modelagem, implementou-se uma heurística para resolução do problema, que serviu em grande parte para realização dos testes das implementações obtidas até o momento. Isso viabilizou a implementação da Busca Local e, conseqüentemente, da Busca Tabu. A modelagem, no entanto, ainda não trata as jornadas do tipo *dupla-pegada*. A abordagem do problema encontra-se detalhada na seção 3 e a implementação final da primeira versão, já com algumas melhorias, pode ser analisada na seção 10.

4. *Testes:* Na primeira iteração de testes utilizou-se a primeira heurística para validar a modelagem das soluções e também foram testadas a Busca Local e a Busca Tabu. Esses testes

utilizaram instâncias reais do transporte urbano metropolitano da cidade de São Paulo, e uma interface criada em trabalhos anteriores [Rod01] para visualização e análise dos resultados.

5. *Melhorias na BT:* Foi implementada uma nova função objetivo que tem todo seu cálculo otimizado, guardando valores que não precisam ser recalculados, e sendo atualizados utilizando-se de informação prévia. A modelagem da Busca Tabu foi feita de modo que se possa lidar com diversas estratégias, do que tirou-se proveito durante esta fase de implementação paralela do projeto. A maior melhoria na Busca Tabu deu-se com a implementação da oscilação estratégica, que conseguiu obter resultados muito bons para todas instâncias do problema.

6. *Testes:* Foram testadas, novamente, a primeira heurística, a Busca Local e Busca Tabu, sobre todas as instâncias fornecidas. Este processo, permitiu, também, ajustar os parâmetros dos vários algoritmos.

7. *Implement. Paralelismo:* Com o objetivo de definir um modelo para a versão paralela, foram realizados estudos em diferentes modelos e aplicações da Busca Tabu Paralelizada que obtiveram bons resultados [BHX02, CTG97, GH02, PR95]. Inicialmente adequou-se o procedimento a um protocolo de comunicação genérico(veja seção 11.4) e foi implementada uma estratégia simples não colaborativa, já nos moldes *master-slave*, que faz parte do modelo paralelo final.

8. *Testes:* A versão paralela simples foi testada e, dessa forma, possibilitou a validação do protocolo de comunicação utilizado e dos aspectos de controle básicos entre as *threads*.

9. *Melhorias Paralelismo:* Neste momento foram desenvolvidos o "Elite Pool" e o componente de "Path Relinking" necessários para o modelo paralelo escolhido. Tendo esses componentes testados, foi implementada a estratégia paralela colaborativa na qual as diferentes *threads* com diferentes parâmetros e pesos nas restrições trocam soluções através do "Elite Pool" e recombinam soluções com o "Path Relinking". Esta versão paralela não incluía a oscilação estratégica pois seus ganhos foram substituídos pela atribuição de diferentes pesos nas restrições para cada *thread* o que denominou-se de *oscilação distribuída*.

   Em conjunto com essas melhorias foi desenvolvida uma heurística para a distribuição dos horários de viagens.

10. *Testes:* O modelo implementado foi testado sobre todas as instâncias fornecidas, possibilitando o ajuste dos novos parâmetros referentes ao modelo paralelizado. Após isso, os resultados obtidos foram comparados em termos de tempo de convergência e qualidade da solução com os resultados obtidos pelo algoritmo monoprocessado.

11. *Relatórios:* Neste período organizou-se o histórico das atividades e a análise dos resultados obtidos durante o semestre para a elaboração dos relatórios e publicações.

12. *Publicações:* Foi confeccionado um pôster que foi apresentado no evento INOVA municípios. Também foi produzido um artigo que concorreu ao CTIC e iniciou-se as atividades para a elaboração de um relatório técnico.

3 O Problema da Programação de Viagens

3.1 Descrição geral

No planejamento operacional de uma empresa de ônibus e transporte urbano, uma situação muito relevante, freqüente e complexa, é o Problema da Programação de Viagens (PPV). Ele consiste em realizar a alocação da frota e dos condutores de uma linha de ônibus, fornecendo-se também os horários de partidas das viagens, para uma dada demanda de transporte. A dificuldade deve-se à grande quantidade de restrições e aos diferentes objetivos a serem atingidos. As restrições vão desde regras e limites no uso da frota, ao cumprimento de leis trabalhistas e acordos sindicais, incluindo restrições no atendimento da demanda dentro de certos padrões de qualidade. Objetivos típicos seriam buscar a minimização do número de veículos e motoristas e do número de horas extras trabalhadas.

Nas empresas, a resolução do PPV geralmente é feita de forma manual, baseando-se na experiência e conhecimentos adquiridos ao longo dos anos de prática. Como o PPV é de grande importância em virtude do transporte de passageiros ser um setor econômico de grande movimentação de recursos, onde quaisquer otimizações, ainda que pequenas, envolvem economias consideráveis, é interessante o uso de ferramentas computacionais para sua resolução. Foi constatado nos trabalhos realizados anteriormente que tais ferramentas conseguem obter resultados muito melhores que as soluções manuais, incentivando o estudo de novos procedimentos.

Para se resolver o PPV alguns dados são necessários, e estão listados abaixo:

- **A Tabela de demanda**: O primeiro estudo realizado sobre uma linha é o da demanda, feito pela própria empresa de ônibus. Para esse fim, ao longo do percurso de uma linha, estabelecem-se Pontos de Controle (PCs). É nesses PCs onde a demanda de passageiros é medida, e isto é feito em cada faixa horária do dia. A divisão do dia em faixas horárias consegue demonstrar a evolução da demanda sobre o dia e é muito importante para o escalonamento e quantatização das viagens.

- **A Tabela de duração de viagens**: O segundo estudo é o da duração das viagens realizadas pelos ônibus entre os PCs durante cada faixa horária do dia e cada sentido do percurso entre PCs. Também é necessário coletar os tempos de viagem entre a garagem e os PCs. Esses tempos serão utilizados para dimensionar as jornadas e conseguir prever quantas viagens serão possíveis.

- **Parâmetros**: Há ainda vários outros parâmetros que devem ser informados, tais como o tamanho da frota, tempo máximo de jornada de motoristas, tempo de descanso, entre outros. A maioria destes parâmetros são decorrentes da existência ou não de restrições que serão vistas mais à frente.

Para facilitar o entendimento do PPV, podemos dividi-lo em três subproblemas, conforme segue:

- Problema da Montagem de Horários **(PMH)** - A entrada deste primeiro problema são os dados de entrada do PPV, ou seja, a tabela de demanda e tabela de duração de viagens. O

objetivo deste problema é determinar a quantidade de viagens por faixa horária e o horário de partida de cada viagem, em cada sentido do percurso. É preferível, neste processo, manter uma boa distribuição de viagens em cada faixa horária, uniformizando o tempo de espera entre as partidas, ao mesmo tempo em que se deve atender à demanda, com a qualidade estipulada.

- Problema do Escalonamento de Veículos **(PEV)** - A escala de ônibus representa a alocação dos veículos disponíveis para realizar todas as viagens resultantes do PMH, de modo que cada uma delas seja alocada a um único veículo. Além disso, os ônibus devem sair da garagem antes da primeira viagem e voltar à garagem após a última viagem. No nosso caso, vamos considerar apenas ônibus de capacidade uniforme e que trafeguem em uma única linha. O objetivo é minimizar a quantidade de veículos utilizados, sem violar restrições operacionais, que listaremos a seguir.

- Problema do Escalonamento de Tripulação **(PET)** - Essa escala atribui viagens aos motoristas, de modo que cada viagem seja alocada a um único par motorista-cobrador. O planejamento será diário, sem considerar folga semanal, férias e feriados. A cada seqüência de viagens associada a um motorista damos o nome de *jornada* . Normalmente, essas jornadas são contínuas. No entanto, pode haver um número restrito de jornadas tipo "dupla pegada", quando as atividades são divididas em dois turnos, separados por no mínimo 2 horas. O objetivo do PET é minimizar a quantidade de motoristas e horas extras trabalhadas, sem violar as restrições operacionais e trabalhistas, listadas a seguir, e cobrindo ainda todas as viagens.

3.2 Restrições e Otimizações

O PPV é um problema de difícil solução, como é bem indicado pela quantidade de restrições e fatores a otimizar, que compoem seu contexto. Abaixo são descritas as restrições tratadas para a realidade da região Metropolitana da Grande São Paulo, que são, em geral, comuns à realidade brasileira.

As restrições mais importantes para a frota são:

- O número de viagens entre PCs deve ser suficiente para atender à demanda com um certo padrão de qualidade. O padrão de qualidade é dado pela lotação máxima de um veículo.

- Os horários de partida devem ser bem distribuídos ao longo do dia.

- O veículo deve sair da garagem antes da primeira viagem e voltar a garagem depois da última viagem.

- Existe um número máximo de veículos disponíveis para uma linha.

- Em um PC não se deve ultrapassar uma capacidade máxima de veículos em espera ao mesmo tempo.

- Existem casos onde não se pode iniciar uma jornada em um PC, e/ou termina-la em um PC.

Para a tripulação, as principais restrições são:

- A jornada de trabalho normal é de 7h20min.
- A jornada de trabalho mínima em geral é de 5h.
- Há um limite de horas extras que um funcionário pode realizar em um dia.
- A quantidade de jornadas tipo "dupla pegada" é limitada a uma porcentagem do número diário de viagens.
- O descanso do motorista pode ser obrigatório, ou não, em uma linha.
- Os motoristas têm direito a um descanso de 30 minutos, entre a segunda e a sexta hora do dia. Caso se escolha não conceder esse descanso, o motorista terá sua jornada encurtada em 30 minutos.
- A rendição, ou troca de veículo ocorrida quando um motorista termina sua jornada e outro assume seu lugar, tem duração de 20 minutos.
- Em alguns PCs e faixas horárias não são permitidas a realização de rendição.

As otimizações a serem realizadas são:

- Minimização do uso de veículos em uma linha.
- Minimização do número de motoristas necessários.
- Minimização das horas extras totais dos motoristas.
- Minimização das horas ociosas totais dos motoristas, ou seja, das horas onde não estão em viagem.
- Minimização dos passageiros atentidos em excesso.
- Minimização do tempo de espera entre as viagens, implicando na boa distribuição das viagens.
- Minimização do número de jornadas do tipo "dupla-pegada".

3.3 Tratamento utilizado

O PPV é um problema recorrente no mundo todo [LPP01] e em cada localidade possui certas características específicas. Existem quatro pontos que realçam bem as diferenças entre as realidades, sendo eles:

1. Existência ou não de troca de veículos entre linhas, conhecida também como "Interlining".
2. Possibilidade de "Changeover", ou troca de veículos durante a jornada de um motorista.

3. Utilização de vários tipos de veículos, com capacidades diferentes.

4. Existência e utilização de diversas garagens que podem servir como ponto de partida e recolhimento dos veículos.

Para a realidade brasileira, como já constatado em [Rod01], cada linha de ônibus é tratada de forma independente, não existindo a troca de veículos entre linhas. Também não há a troca de veículos na jornada de um motorista, implicando, com o tratamento independente das linhas, na impossibilidade de troca de linha por um motorista. Em geral, é utilizada somente uma garagem para abastecer uma linha e os veículos utilizados contém a mesma capacidade.

Essas considerações podem reduzir as possibilidades de reutilização dos veículos e motoristas, mas, em contrapartida, reduzem a complexidade da gerência e operação das linhas.

Na modelagem utilizada neste projeto usaremos as características da realidade brasileira, como forma de facilitar a modelagem e permitir uma maior atenção na melhoria das soluções para as instâncias disponíveis.

A modelagem ficará centralizada nas jornadas dos motoristas. Essas jornadas são compostas por tarefas, que podem ser ida ou volta da garagem, viagem entre PCs, rendição ou descanso. Uma jornada de motorista independe do ônibus no qual se localiza, isto porque os ônibus são todos do mesmo tipo e estão todos na mesma linha, respeitando a realidade descrita. A partir de uma jornada, conseguimos tirar muitas informações referentes à tripulação, como o tamanho da jornada, o número de horas extras e ociosas, e a existência ou não de descanso, inferindo, desse modo, se a jornada respeita as restrições impostas. Sabemos também, com a alocação das tarefas de viagens na jornada, os horários de partida e conseguimos com essa informação verificar o atendimento da demanda nas diversas faixas horárias atendidas. Para formar uma solução para o problema, devemos alocar as jornadas de motoristas em veículos, de forma a atender a demanda, respeitando as restrições e minimizando os recursos necessários. Nessa alocação, podemos ter mais de um motorista em um ônibus, na condição de que as jornadas sejam consecutivas e que não sejam alocadas ao mesmo tempo. Desta forma, conseguimos resolver o PEV, PET e PMH de forma integrada, já que a solução fornece os ônibus utilizados, os motoristas com suas jornadas e as jornadas com as informações do horários de partida e outras restrições.

Na próxima seção será apresentado a implementação do Algoritmo de Busca Tabu monoprocessado. Poderá ser constatado que a implementação, utilizando dos conceitos desta seção, conseguiu tratar todas as restrições e permitiu várias otimizações, obtendo bons resultados.

4 A Enumeração de Jornadas

Como objetivo principal da primeira fase de projeto, buscou-se compreender o problema de forma profunda, para aproveitar ao máximo suas características e especificidades. Deste modo, foi dada grande atenção à modelagem do problema, pois ela, em conjunto com as vizinhanças escolhidas, são fatores de grande impacto nos resultados obtidos pela Busca Tabu.

Desde o início, buscou-se modelar o problema de forma a garantir um espaço de busca de qualidade. Para entender quais aspectos são comuns em soluções de qualidade, foram realizadas

análises sobre as soluções encontradas por outras abordagens ([Rod01, Vaz03, Cir04]), e notados certos pontos que valem citar:

1. As jornadas de motoristas possuem um conjunto limitado e específico de características para cada instância dada. Aparentemente, essas jornadas não divergem muito, e pode-se facilmente descrever a diferença entre elas.

2. As melhores soluções têm menos tempo desperdiçado pelos motoristas em suas jornadas. Ou seja, o motorista fica pouco tempo esperando pela sua próxima viagem. Esse tempo disperdiçado acaba resultando em menos viagens por motoristas e cria a necessidade de um número maior de horas extras e motoristas para o atendimento completo da demanda.

3. As jornadas situadas em um mesmo ônibus não interferem entre si. Isto porque cada motorista tem todas suas tarefas restritas a um único ônibus. Esta independência dos motoristas é importante na modelagem da solução, como será visto.

4. Ao analisarmos uma jornada, não é necessário considerarmos as tarefas de ida e volta à garagem ou de rendição. A tarefa de saída de garagem é necessária para o atendimento da primeira tarefa da jornada, e acaba influenciando somente o tamanho da jornada do motorista. O mesmo ocorre com a tarefa de volta à garagem que deve ser realizada após a última tarefa.

Esses pontos levaram à conclusão de que os elementos principais da solução são as jornadas de motoristas. Então, para conseguirmos soluções de qualidade, devemos ser capazes de encontrar boas combinações de jornadas. A primeira dificuldade para se conseguir este objetivo está em modelar as jornadas de forma a conseguir um grande número de jornadas, que difiram entre si e tenham impactos diferentes na solução. O significado disso para o procedimento de busca iterativa é a escolha de um movimento que consiga explorar de forma eficiente o espaço de busca. Um movimento eficaz para nosso problema deveria alterar propriedades que influenciam diretamente na função objetivo. Como exemplo, devemos conseguir alterar o horário de inicio da jornada, tamanho da jornada, o número de tarefas, inserir descansos, o PC de início da jornada e o PC de fim da jornada.

Uma segunda dificuldade seria conseguir jornadas com poucas diferenças entre si para que consigamos realizar um refinamento da solução. A principal característica com que um movimento deveria lidar neste ponto seria a distribuição dos horários das tarefas dentro uma jornada. No entanto, a alteração das tarefas de uma jornada de motorista é um procedimento complexo, pois são poucas as propriedades que se pode alterar e que não implicam em refazer toda a jornada. Se pensarmos em fazer um deslocamento de alguns minutos em uma tarefa, podemos acabar tendo que deslocar todas as tarefas do motorista, e do motorista seguinte, fazendo com que a demanda, e tempos de trabalho se alterem. É muito difícil avaliar se esse tipo de movimento nos permitirá atingir soluções viáveis.

Dada a dificuldade em se definir os movimentos para alteração das jornadas, foi tomado outro caminho, onde não é necessário realizar alterações na estrutura interna de uma jornada. Para tal geramos uma enumeração de jornadas, e construimos a solução referenciando as jornadas desta enumeração.

4.1 Detalhes da Enumeração

Usar uma enumeração de jornadas pode ser muito vantajoso em diversos aspectos:

- Remove a complexidade de se tratar alterações internas de uma jornada, pois já fornece todas as jornadas que podem estar em uma solução.
- Fornece uma grande diversidade de vizinhos e ainda limita o espaço de busca, deixando-o mais promissor.
- Absorve diversas restrições não permitindo a existência de jornadas inviáveis. Deste modo, podemos retirar esse item do tratamento do algoritmo.
- Permite que certas restrições mais específicas fiquem transparentes ao algoritmo. Como exemplo, se não for permitido que jornadas iniciem em um determinado PC, a enumeração pode absorver essa restrição e não gerar jornadas que a apresentem.

Além disso, o conceito de uma enumeração de jornadas [LPP01] ou de um conjunto de elementos usado na construção de uma solução [PGSH96], já foi utilizado em outros trabalhos semelhantes, com bons resultados. E mais, na paralelização do algoritmo, a utilização da enumeração deixará mais eficiente a troca de soluções entre os algoritmos paralelos. Isso por que a utilização de referências à enumeração na construção da solução é bem mais eficiente que a transferência de toda a estrutura da jornada.

Para se utilizar da enumeração de jornadas nos algoritmos, devemos criar um módulo separado, que faça a geração das possíveis jornadas, dada uma instância do problema.

4.2 Geração da Enumeração

Durante o procedimento de gerar as jornadas, conseguimos estabelecer exatamente algumas de suas propriedades. E esta informação pode ser útil durante a escolha das jornadas que serão utilizadas na solução. Por esse motivo, a enumeração é construída como uma matriz n-dimensional, onde cada dimensão representa uma propriedade da jornada. Um exemplo de como essas dimensões podem ser úteis está na geração da solução inicial que será descrita em seguida. Abaixo, segue uma descrição de cada dimensão da enumeração e como ela pode afetar a solução:

- *1ª Dimensão*: "Horário Inicial". O horário de início da jornada é determinante, em conjunto com a segunda dimensão, para se garantir o atendimento da demanda.
- *2ª Dimensão*: "Tamanho da Jornada em Horas". O número de horas que a jornada possui é importante para a cobertura de demanda, número de horas extras e tempo ocioso.
- *3ª Dimensão*: "Número de Tarefas". A quantidade de tarefas utilizadas influi no tempo ocioso, horas extras e demandas atendidas.
- *4ª Dimensão*: "PC Inicial". O PC de início das viagens é restrição em algumas instâncias. Na enumeração, podemos já eliminar as jornadas que não satisfazem essas restrições.

- *5ª Dimensão*: "PC Final". O PC final também é uma restrição, em algumas instâncias.
- *6ª Dimensão*: "O Horário do Descanso". Existem horários bem definidos para o descanso, quando estes forem necessários.
- *7ª Dimensão*: "Total de Tempo Ocioso". É importante para o número de horas extras e, obviamente, de horas trabalhadas.
- *8ª Dimensão* : "Aleatoriedade dos Inícios das Tarefas". Tenta abrigar as outras propriedades não mapeadas pelas outras dimensões, e é muito útil para distribuir os horários e aumentar o número de jornadas.

A geração da enumeração é um processo demorado que, no entanto, só necessita ser realizado uma vez para cada instância, de forma que o resultado pode ser gravado em arquivo e reutilizado pelos algoritmos nas várias execuções. Pode-se gerar, ainda, enumerações diferentes, isso porque a enumeração recebe como entrada parâmetros que podem ser configurados para limitar ou expandir o número de jornadas e também por usar de aleatoriedade em suas construções. Um mecanismo muito útil para elevar o número de jornadas é modificar a última dimensão da enumeração.

Em suma, o algoritmo recebe como entrada a instância do problema, e intervalos sobre as dimensões citadas anteriormente. Por exemplo: deseja-se construir jornadas que tenham de 3 a 12 horas, com 5 a 10 tarefas, que só comecem no PC 0, e só terminem no PC 0, etc. A instância do problema é necessária para se calcular os tempos das tarefas, e também pode influir diretamente em algumas dimensões, impossibilitando certos valores de intervalos. Essa última afirmação pode ser vista quando desejamos gerar jornadas que iniciem em qualquer um dos dois PCs. Porém, para a instância dada existe uma restrição de inicio de jornada, sendo esta permitida somente no PC 0. As outras considerações importantes para a enumeração são listadas a seguir:

- Caso o motorista não tenha descanso no intervalo definido pela instância, ele não pode trabalhar mais do que a jornada normal menos 30 minutos.
- Existe um número máximo de horas extras definidas pela instância que, se for ultrapassado, implica no descarte da jornada.

Recebidas essas informações, o algoritmo realiza um procedimento simples de combinar todos os valores nos intervalos entre si tentando, com cada combinação, gerar uma jornada que tenha os valores estipulados. Se for possível gerar uma jornada obedecendo aquelas restrições, esta jornada faz parte da enumeração. Tipicamente, observou-se, na prática, uma redução no número de jornadas geradas para, aproximadamente, 0,5% das jornadas possíveis. Como foi dito anteriormente, a enumeração é uma matriz n-dimensional onde cada dimensão representa alguma propriedade. Porém, a matriz construída por este procedimento não é indexada diretamente pelo valor da propriedade, e sim incrementalmente conforme a existência das jornadas. Por exemplo, se a primeira jornada que se consiga gerar começa na décima hora, ela não será acessada com índice 10 na primeira dimensão, e sim com índice 0; a próxima jornada, mesmo estando na vigésima hora, será acessada com índice

1. Esta medida foi usada porque verificou-se que a matriz é muito esparsa, dada a freqüencia de conflitos entre propriedades, e poderia ocupar espaço demasiado da memória.

Para conseguir gerar a jornada, com a combinação de propriedades pedida, foi criado um procedimento que insere as tarefas uma a uma, de forma que no final do processo ela satisfaça a maior parte das propriedades. Este procedimento ainda necessita de uma matriz com as médias de tempo das viagens entre cada dois horários no dia. Essa matriz será necessária para prever tempos de trabalho e descanso, e conseguir distribuir as tarefas pela jornada. O pseudo-código para geração de jornadas é mostrado no algoritmo 1.

Algoritmo 1 Criação de Jornada de Motorista

```
1: procedimento GERARJORNADA(dimensoesPedidas,instanciaDoProblema,mediaTempoViagens)
2:     novaJornada ← jornada vazia
3:     minuto_inicio ← horaIncial * 60 + NroAleatório(0,59)
4:     maximoMinuto ← horaFinal * 60
5:     enquanto Não alcançou tamanhoJornada ou Não alcançou nroTarefas faça
6:         se É a hora do descanso então
7:             ultimaTarefa ← descanso iniciando em minuto_inicio
8:         fim se
9:         se nroTarefasInseridas = nroTarefas − 1 então
10:            minutoFim ← horaFinal * 60 + NroAleatório(0,59)
11:            ultimaTarefa ← nova viagem terminando em minuto_fim
12:        senão
13:            ultimaTarefa ← nova viagem iniciando em minuto_inicio
14:        fim se
15:        Adiciona a ultimaTarefa na novaJornada
16:        nroTarefasInseridas ← nroTarefasInseridas + 1
17:        Atualiza as variáveis sobre a jornada
18:        maximoTempoDisperdisavel ← maximoMinuto - ultimaTarefa.minutoFim -
       mediaTempoViagens[ultimaTarefa.horaInicio][horaFinal]*(nroTarefas - nroTarefasInseridas)
19:        minuto_inicio ← ultimaTarefa.minutoFim + NroAleatório(0,maximoTempoDisperdisavel)
20:    fim enquanto
21:    Verifica se a jornada satisfaz as restrições pedidas
22:    retornar novaJornada
23: fim procedimento
```

Deve-se notar que o algoritmo 1 não inclui as tarefas de ida e volta à garagem, e nem o tempo de rendição. Esses pontos são automaticamente atendidos pela modelagem da solução. O motivo disto vem do quarto ponto levantado no começo desta seção. Todas as jornadas da enumeração já são geradas de forma a satisfazer certas restrições, como por exemplo: em uma seqüência de tarefas, uma tarefa qualquer t_1 tem $minuto\_final_{t1}$ menor que $minuto\_inicial_{t2}$ onde t_2 é a tarefa seguinte. Da mesma forma, t_1 tem como pc_final_{t1} igual a $pc\_inicial_{t2}$. Existem ainda outras garantias a serem exigidas de duas jornadas em seqüência, e que também são diretamente atendidos pela modelagem da solução.

O procedimento para geração da enumeração proposto é heurístico. Portanto, não há garantias da qualidade das jornadas que serão geradas. Os resultados obtidos pelos algoritmos resolvedores foram satisfatórios, o que permite dizer que este procedimento é suficiente. No entanto, uma mo-

delagem matemática poderia ser utilizada para a construção da enumeração, como em [PGSH96]. É importante dar atenção à geração da enumeração, já que, pode-se dizer que a sua "qualidade" é um fator limitante aos algoritmos, pois as soluções serão construídas a partir das jornadas contidas na enumeração.

4.3 Vantagens da Enumeração ao Paralelismo

Outra vantagem da Enumeração é que ela viabiliza a implementação do paralelismo, visto que não é mais necessário transmitir todos os horários de partida de uma solução para reconstruí-la em outro nó. Agora, basta transmitir os índices das dimensões das jornadas na enumeração, considerando que a semente de construção da Enumeração é a mesma para todas as populações.

Empiricamente, foi verificado que isto reduz o *overhead* de transmissão em cerca de 95%. Mais detalhes da implementação do protocolo de comunicações pode ser encontrado na seção 11.4.

5 A modelagem da solução

A modelagem da solução deve ser feita para prover acesso rápido às informações e facilitar as modificações da solução e, se possível, automaticamente garantir a satisfação de certas restrições. Para tal, criou-se uma estrutura de dados adequada. Para o PPV foi escolhida uma modelagem orientada a objetos, que se utiliza de classes tais como: solução, ônibus, motorista, lista de tarefas e tarefa. Essas classes possuem diversas propriedades para acelerar a obtenção de informações. Podemos descrever a relação entre elas de forma sucinta, conforme a seguir:

- A solução possui um vetor de ônibus, sendo que este vetor tem tamanho igual ao número máximo de veículos permitidos na instância dada.
- Os ônibus têm associado a eles uma lista de motoristas e também uma lista de tarefas que são realizadas por aqueles ônibus.
- Para cada motorista temos três apontadores para tarefas que estão situadas na lista de tarefas de seu ônibus. O primeiro apontador é o da primeira tarefa do motorista, podendo ser uma saída da garagem ou uma rendição. O segundo apontador é referente a uma "tarefa jornada", que será explicada a seguir. Já o terceiro apontador indica a última tarefa do motorista, que é uma volta à garagem ou rendição.

Como já mencionado, a responsabilidade pelas tarefas de ida e volta da garagem, e também a rendição, ficaram com a modelagem da solução e estão situadas na lista de tarefas de um ônibus. Uma tarefa contém algumas propriedades básicas como: PC de início, PC de fim, minuto de início, minuto de fim, faixa horária, duração da tarefa e tipo da tarefa. Os tipos de tarefas possíveis, já citados, são: volta da garagem, ida à garagem, viagem, descanso, rendição e jornada. O tipo jornada só aparece em uma tarefa estendida que é a "tarefa jornada". Esta tarefa possui uma propriedade a mais que é a referência a uma jornada da enumeração. Além disso os valores das propriedades de minuto de início e PC de início são referentes à primeira tarefa da jornada na enumeração, enquanto que as propriedades PC de fim e minuto de fim são referentes à última

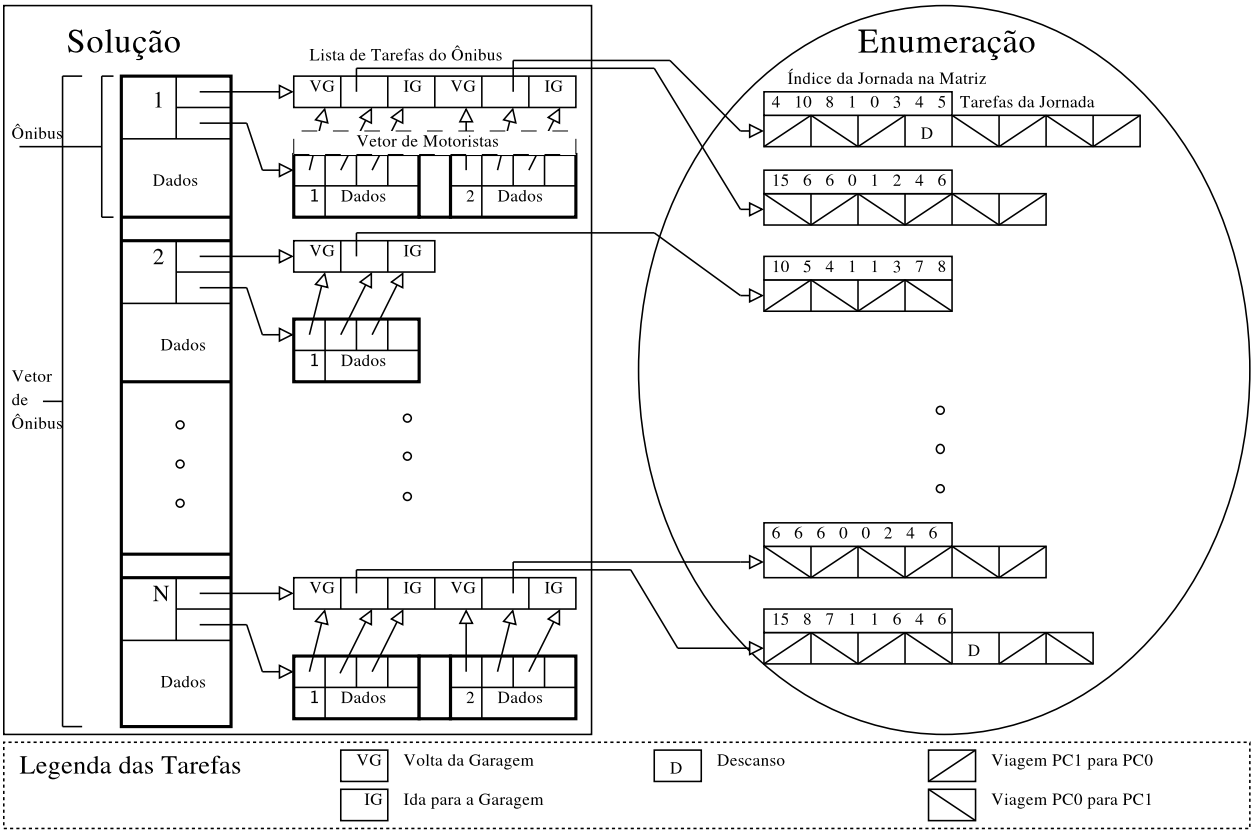

Figura 1: Modelo da estrutura da solução

tarefa da jornada. A duração da tarefa jornada é a duração da jornada referenciada. O modelo da estrutura da solução pode ser visto na figura 1.

A lista de tarefas de um ônibus deve obedecer às mesmas regras de viabilidade de uma jornada.

6 Geração de solução inicial

Os algoritmos de busca iterativa, como a Busca Local e Busca Tabu, necessitam de uma solução inicial para que novas soluções sejam obtidas a partir dos movimentos realizados. Usualmente decide-se entre construir uma solução inicial aleatoriamente, ou cria-se alguma heurística mais inteligente para se obtê-la.

Neste projeto, foi usada uma heurística para a construção da solução inicial baseada no procedimento manual utilizado pelas empresas do transporte urbano. O algoritmo 2 faz uso da enumeração e uma de suas funcionalidades: a obtenção de jornadas que satisfaçam a propriedade de começar ou terminar numa dada faixa horária.

Algoritmo 2 Geração de Solução Inicial

```
1: procedimento GerarSolucaoPPV(Enumeracao,InstanciaPPV)
2:     Solucao ← Nova Solucao
3:     numeroVeiculos ← 0
4:     direcao ← da menor faixa horária para a maior
5:     enquanto numeroVeiculos < InstanciaPPV.maxVeiculos faça
6:         ListaTarefas ← Lista vazia
7:         Veiculo ← Novo Veiculo
8:         enquanto Achar PC e faixaHoraria com demanda pendente na InstanciaPPV para a direcao atual faça
9:             se direcao = "da menor faixa horária para a maior" então
10:                NovaJornada ← Enumeracao.BuscaJornadaQuePrimeiraTarefaInicie( PC, faixaHoraria)
11:            senão
12:                NovaJornada ← Enumeracao.BuscaJornadaQueÚltimaTarefaInicie( PC, faixaHoraria)
13:            fim se
14:            NovaTarefaJornada ← Nova Tarefa Jornada com NovaJornada
15:            Adiciona NovaTarefaJornada a ListaTarefas
16:        fim enquanto
17:        AlocarMotoristasEColocarGaragensERendicoes(ListaTarefas)
18:        se ListaTarefas não ficou vazia então
19:            Atribuir a ListaTarefas ao Veiculo
20:            Adiciona o Veiculo a Solucao
21:        fim se
22:        Inverte direcao
23:    fim enquanto
24:    retornar Solucao
25: fim procedimento
```

7 A Função Objetivo

A função objetivo utilizada pelos algoritmos que foram implementados é uma função linear sobre as restrições e parâmetros de otimização do PPV. Como foi dito, a enumeração absorveu algumas das restrições, sendo desnecessário considerá-las no cálculo da função objetivo. A seguir, será demonstrado como cada uma das restrições e otimizações irá influenciar o cálculo da função objetivo, e quais delas foram evitadas pela enumeração.

Restrições relativas à frota:

Quando uma restrição é violada, aplica-se uma penalidade específica para aquela restrição que é somada ao valor f da função objetivo. Para a frota, a restrição de número máximo de veículos pode ser descartada pois os algoritmos satisfazem uma limitação de veículos igual a esse número máximo. Já a restrição de início ou término de jornadas em determinados PCs é satisfeita pela enumeração, não existindo jornadas que violem esta restrição. A única restrição não automaticamente satisfeita é o atendimento da demanda.

Porém, há instâncias em que o número de veículos disponíveis não é suficiente para o atendimento completo da demanda ou, inversamente, as soluções finais podem conter um número de veículos em excesso. Desta forma, para avaliar a *variação* da demanda em cada faixa horária e PC, foram utilizadas duas penalidades adicionais: *custoPorPassSuperAtend*, aplicada a cada passageiro em excesso, e *custoPorPassSubAtend*, aplicado a cada passageiro ainda não atendido. Isto

induz o algoritmo a buscar soluções mais próximas de um *atendimento ideal* em cada PC, eliminando veículos e motoristas em excesso. Além disso, deve-se estabelecer $custoPorPassSubAtend \gg custoPorPassSuperAtend$, para não prejudicar a qualidade de atendimento da linha.

Restrições relativas à mão-de-obra:

No caso das restrições relativas à mão-de-obra, grande parte não precisou ser calculada pois foram evitadas pela enumeração. A penalidade é aplicada quando:

- Não foi dado descanso ao motorista, quando obrigatório;
- O descanso não ocorreu dentro do intervalo de tempo permitido;
- Foi atribuída hora extra ou um tempo regular de trabalho para um motorista que não descansou;
- A rendição ocorreu em um PC não permitido.

A restrição de horas extras de um motorista acima do limite permitido pode ocorrer, pois mesmo a enumeração prevendo esta restrição, após a inserção na solução das tarefas de garagem ou rendição, a jornada do motorista pode se estender e violar a restrição. Se isso acontecer, é aplicada uma penalidade ao valor de f.

Otimizações: Para otimizar recursos, foram aplicadas penalidades específicas para a utilização de cada um. Assim, para cada veículo aplicamos uma penalidade $custoVeiculo$ e para cada motorista uma penalidade $custoTripulacao$. Isto guia os procedimentos no sentido de minimizar a utilização destes recursos. Uma observação relevante é que tais valores devem ser relativamente baixos, evitando que se reduzam veículos em detrimento do atendimento da demanda.

Para otimizar jornadas de motoristas, serão contabilizados os custos $horaRegular$, $horaOciosa$ e $horaExtra$, para cada hora em questão. Tais valores devem ser baixos e definidos de forma que $horaRegular < horaOciosa < horaExtra$, reduzindo o número total de horas extras e ociosas e impedindo que as jornadas se tornem mais relevantes do que outras restrições operacionais.

Detalhes da implementação: A primeira função objetivo implementada percorria toda a solução diversas vezes, para determinar a violação de restrições e utilização de recursos. Essa implementação era muito ineficiente, e se tornou rapidamente o gargalo de execução do algoritmo. Para otimizar a função objetivo, foram criadas estruturas internas que guardam valores associados às restrições e à utilização de recursos relativos a um ônibus, incluindo o atendimento da demanda. Isso permite que saibamos como cada ônibus influi na função objetivo, de forma que cada ônibus i contribui com uma parcela f_i para o valor da função objetivo f. Desta forma, essas estruturas são atualizadas a cada modificação realizada em um ônibus. O cálculo da função objetivo pode ser visto como $f_{novo} = f_{velho} - f_{i\_velho} + f_{i\_novo}$, onde i é o ônibus em consideração.

8 A Heurística de Trocas Aleatórias

Tendo a enumeração, modelagem, solução inicial e função objetivo em mãos, o primeiro passo foi criar um algoritmo visando testes, para o aperfeiçoamento das estruturas de dados. Este algoritmo serviu plenamente para seu propósito e já conseguiu soluções com características interessantes. A heurística foi apelidada de Busca Cega, pois realiza uma busca através de modificações puramente aleatórias.

Algoritmo 3 Heurística de Trocas aleatórias

```
 1: procedimento BuscaCega(parametros)
 2:     Crie solução S pelo método de GeraçãoHeurística
 3:     Avalie a solução S
 4:     enquanto nro_iteracao < parametros.totalIteracoes faça
 5:         Escolha um ônibus i e um motorista j qualquer na solução
 6:         Busque NovaJornada na enumeração ou com uma probabilidade baixa, deixe-a vazia
 7:         se o ônibus i não estiver alocado, ou o motorista j não estiver alocado então
 8:             aloque
 9:         fim se
10:         Atribua a NovaJornada para i, j
11:         Reavalie a solução S
12:         se melhorar a solução então
13:             mantenha a jornada
14:         senão
15:             desfaça as alterações
16:         fim se
17:     fim enquanto
18:     retornar S
19: fim procedimento
```

Como descrito no algoritmo 3, a heurística tenta, a cada iteração, melhorar a solução. Caso não melhore, ela desfaz as alterações. Deste modo, a heurística implementa um procedimento de descida que não mantém qualquer memória sobre o processo. Na linha 6, o procedimento tenta, ao deixar a $NovaJornada$ vazia, diminuir o número de motoristas e ônibus da solução, já que para a implementação a atribuição de jornadas vazias é o mesmo que a retirada do motorista. Já na Linha 8 o procedimento tenta alocar novos motoristas podendo, deste modo, atender à demanda restante. A troca por uma jornada qualquer na enumeração tenta melhorar a estrutura interna das soluções.

Em geral, esse procedimento consegue retirar as restrições mais fortes mas é muito incipiente quando se deseja otimizar recursos. Para fins de testes, este algoritmo foi forçado a resolver o problema com limitação de recursos, conseguindo gerar soluções viáveis, muitas vezes melhores que outras criadas pelo procedimento de Busca Local descrito na próxima seção.

A grande eficiência em satisfazer as restrições é atribuída ao alto grau de diversificação atingido pelo procedimento, característica que será útil para a Busca Tabu.

9 A Busca Local

Um passo natural antes da implementação da Busca Tabu (BT), é a implementação da Busca Local (BL), que muitas vezes é reutilizada na própria BT. Neste momento, muitos dos componentes estão definidos, como por exemplo, os movimentos, as vizinhanças e o critério de escolha das soluções. É possível também observar as dificuldades da BL, e tentar corrigir esses defeitos com as melhorias específicas da BT. O algoritmo da BL foi implementado em separado da BT. Porém sua implementação é exatamente a mesmo da BT desconsiderando a lista tabu, a oscilação estratégica e os critérios de aspiração. Assim, pode-se depreender o pseudo-código da BL ao se analisar o pseudo-código da BT, apresentado adiante.

9.1 As vizinhanças

A vizinhança de uma solução é formada por todas as soluções que são alcançáveis a partir dela por um movimento qualquer. Este movimento deve ser definido de forma que realize alguma modificação na solução, alterando propriedades que influem na função objetivo. Deste modo, podemos, a partir do movimento, determinar a vizinhança de uma solução, avaliar essas novas soluções e assim escolher o melhor movimento, ou seja, aquele que leva à melhor solução da vizinhança.

A partir desta definição é fácil notar a importância do movimento na obtenção de soluções. Vemos também que a capacidade do movimento em gerar novas soluções será determinante para percorrer o espaço de busca de forma eficaz. Para o PPV, é desejável que os movimentos consigam, de algum modo, não violar as restrições e otimizar os recursos utilizados. Para estes fins foram escolhidos 4 tipos de movimentos, cada qual gerando sua própria vizinhança de soluções.

O movimento mais simples é o de troca de motoristas entre quaisquer dois ônibus da solução. Nesta vizinhança não são considerados movimentos que gerem inviabilidades. A vizinhança para este movimento tem um tamanho máximo que é proporcional ao quadrado do número de motoristas na solução. O motivo para a utilização desta vizinhança é tentar otimizar o número de ônibus e motoristas utilizados.

Os movimentos de inserção e remoção de jornadas são subcasos do último tipo de movimento, mas recebem atenção especial pois influenciam em muito a qualidade das soluções. Para o movimento de inserção é gerada uma vizinhança proporcional ao número de ônibus. Enquanto que, para o movimento de remoção, são consideradas as remoções de cada um dos motoristas da solução. Estes dois movimentos conseguem melhorar o número de motoristas e ônibus usados, e também influenciam na demanda atendida.

Com os três movimentos citados, conseguimos alterar demanda e otimizar recursos, como motoristas e ônibus. Para otimizar horas extras, horas ociosas, e reduzir o sobre-atendimento da demanda, precisamos de um movimento que altera as jornadas já colocadas na solução, encontrando jornadas substitutas que melhorem as existentes. Para este fim, existe o movimento que troca jornadas por outras jornadas da enumeração. Esta troca não é realizada aleatoriamente, como no primeiro algoritmo, e sim de forma mais apurada, tentando manter certas propriedades interessantes das jornadas. É usado o recurso de buscar jornadas na enumeração e que satisfaçam a certas propriedades. Assim, considerando as propriedades de uma jornada de forma ordenada,

crescente pela dimensão como visto em 4.2, fixamos as primeiras m propriedades, onde m é uma variável pré-definida. Então, para cada um dos possíveis valores da propriedade $m+1$, buscamos uma jornada na enumeração que satisfaça as primeiras m propriedades fixadas combinadas com os valores possíveis da propriedade $m+1$, para ser trocada pela jornada analisada.

Para exemplificar, vamos considerar que desejamos trocar uma jornada que inicie na hora 7 e tenha duração de 8 horas. Se $m = 1$, então devemos buscar jornadas com ínicio na hora 7, já que a 1ªdimensão é a de horário inicial e está fixada. A dimensão $m+1$ equivale a 2ªdimensão, de duração da jornada. Assim, devemos buscar jornadas para cada possível tempo de duração, ou seja, jornadas que iniciem na hora 7 e tenham 6,7,8,9 ou 10 horas de duração, por exemplo.

Desta forma teremos, para uma jornada, um máximo de trocas, que é exatamente o número de valores que a propriedade $m+1$ pode assumir. Mesmo realizando estas trocas para cada jornada da enumeração, o número de valores da propriedade $m+1$ é reduzido e, portanto, temos uma vizinhança pequena. Essa vizinhança possui um tamanho diferenciado para cada m adotado, de forma que, se variarmos m durante o processo, teremos uma vizinhança dinâmica e mais adequada para cada fase do algoritmo.

Todos os movimentos considerados possuem uma vizinhança controlada, ou seja, não crescem exponencialmente, nem as vizinhanças são grandes demais para deixarem de ser totalmente avaliadas.

9.2 Escolha do movimento

Uma estratégia de escolha de movimento deve levar em consideração o tamanho das vizinhanças. No caso de vizinhanças muito grandes, é comum não realizar uma avaliação completa, e sim escolher o primeiro movimento que apresente melhoria ou o melhor dentro de um máximo de movimentos. No caso das vizinhanças modeladas, uma avaliação completa mostrou-se sempre ser a mais apropriada.

10 A Busca Tabu

O objetivo final da primeira fase do projeto foi a implementação do algoritmo de Busca Tabu monoprocessado. Para tal, foi necessário definir a lista tabu e os critérios de aspiração, componentes estes que servem para guiar a Busca Tabu para fora dos mínimos locais. Foi usada uma Lista Tabu que se aproveitou das características da enumeração de jornadas, parametrizando a forma como ela restringe ou libera a aceitação de movimentos. A escolha de um critério de aspiração clássico foi suficiente e efetivo.

Ao contrário da Busca Local, o procedimento de Busca Tabu não é reiniciado depois de um certo número de iterações sem ganho no valor da melhor solução. Ao invés, optou-se por usar uma oscilação estratégica, fazendo a busca passar por regiões que antes não seriam atingidas, ou por causa da atratividade criada pela função objetivo ou da inviabilidade da região.

A seguir serão detalhados cada um dos componentes, e como sua implementação afetou a Busca Tabu.

10.1 A Listas Tabus

O objetivo da lista tabu é impedir a reincidência de soluções para que o processo não entre em um ciclo e acabe preso em um mínimo local. Como o principal componente da modelagem são as jornadas, independentes do ônibus dos quais elas se encontram, uma solução se repetirá quando seu conjunto de jornadas for o mesmo. Para impedir este acontecimento, as jornadas serão os atributos aos quais os tabus serão aplicados. Porém, é visível que as jornadas possuem inúmeras diferenças entre si, e ainda, existem muitas jornadas que se assemelham às outras e podem resultar em pouca diferença na função objetivo. Se forem consideradas as jornadas com todas suas propriedades como tabu, podemos ficar presos em mínimos locais que reaproveitem as jornadas semelhantes em muitas formas. Deste modo, devemos considerar propriedades comuns às jornadas, de modo que uma tupla destas propriedades deixe um número maior de jornadas como tabu. Por exemplo, podemos considerar que ao utilizarmos uma jornada que se inicia às 7 horas, com 10 tarefas e 8 horas de duração é tabu; assim, qualquer jornada que se enquadre nessas condições, também será tabu. É fácil notar que a restringência da lista tabu, ou seja, a forma com que a lista restringe ou libera a busca, é inversamente proporcional ao número de propriedades que consideramos.

A implementação realizada utilizou-se das propriedades que a enumeração gerencia, e a restringência da Lista Tabu é configurável, variando de 0 a 8, ou seja, considerando nenhuma ou todas as dimensões da enumeração. Se não for considerada nenhuma dimensão, todas as jornadas serão consideradas tabu. Para os movimentos usados na Busca Tabu, qualquer jornada que se enquadre de alguma forma nas condições impostas será tabu ativa.

A representação interna da lista tabu se dá por uma matriz n-dimensional semelhante àquela usada na enumeração de jornadas. Essa estrutura é construída até a dimensão igual ao nível de restringência escolhido, copiando exatamente as propriedades da enumeração até aquele ponto. Neste modelo, se escolhermos a restringência como 8, a estrutura conterá um elemento para cada uma das jornadas da enumeração e, neste elemento, é guardada a última iteração que a jornada foi usada em algum movimento. Assim, podemos verificar, utilizando o tamanho da Lista Tabu, se na iteração atual, uma jornada implica no movimento ser tabu ou não. A figura 2 ilustra como seria a estrutura da lista tabu, no caso de se considerar uma ou duas dimensões.

Horário de Partida										
...	1	2	3	4	6	7	8	9	10	...
	10	0	4	0	5	0	1	2	0	

Tamanho da Jornada	Horário de Partida										
	...	1	2	3	4	6	7	8	9	10	...
...											
5		1	0	0	1	0	0	0	5	0	
6		0	0	5	0	0	1	0	1	0	
7		2	0	0	0	0	0	0	6	0	
8		0	0	1	11	0	2	0	0	1	
9		0	0	10	1	0	1	0	4	0	
10		3	0	0	0	0	0	1	0	0	
...											

Figura 2: Representação da lista tabu

10.2 Critério de Aspiração

Durante o processo de busca, a lista tabu serve como um diversificador efetivo para encontrar novas regiões. No entanto, em alguns momentos dessa diversificação podemos encontrar um movimento tabu que pode ser útil à busca. Para permitir a aceitação desses movimentos, são definidos critérios de aspiração que, ao serem satisfeitos, permitem que um movimento tabu seja aceito. Um critério de aspiração clássico, que foi usado no algoritmo implementado, aceita o movimento se este levar a uma solução melhor do que qualquer outra já encontrada.

10.3 Oscilação estratégica

A função objetivo possui uma influência muito grande na forma com a busca é guiada, e a atratividade a certas regiões pode dificultar a exploração do espaço de busca. É possível verificar que as restrições mais fortes acabam sendo descartadas rapidamente pelo processo de busca, sendo sua forte atração resultado de suas grandes penalidades na função objetivo. A implementação básica da Busca Tabu conseguiu bons resultados, como será visto na seção de testes. Porém, mesmo assim, investigando a fundo a função objetivo, foi identificada uma possível atração que poderia estar prejudicando as soluções encontradas. A restrição de atendimento da demanda é forte e para retirá-la são inseridas novas jornadas na solução até ela ser removida. Isso ocorre com grande eficiência, e após isso, a retirada de jornadas da solução se torna muito complicada, pois a retirada de qualquer outra jornada quase sempre leva ao não-atendimento da demanda, o que proíbe a remoção. Existe ainda uma outra condição indesejável, que ocorre quando falta pouca demanda a ser atendida e nenhuma jornada consegue atendê-la completamente, pois a demanda retirada com essa jornada tem peso menor que a adição de um motorista.

Para resolver os dois problemas citados foi definida uma oscilação estratégica, alterando os pesos das restrições, alcançando soluções que não seriam consideradas, dadas suas atratividades. Esta oscilação dividiu o algoritmo de busca em um ciclo de duas fases, onde a segunda se inicia após um número de iterações sem melhorias na melhor solução encontrada. As duas fases são: busca normal, e busca com restrições modificadas. A última possui dois tipos de modificações que se alternam durante as várias fases. A primeira modificação se dá ao desconsiderar a penalidade pela violação da restrição do não atendimento da demanda e diminui-se em muito a penalidade por cada demanda não atendida tentando, desta forma, remover motoristas da solução. A segunda modificação, é como a primeira, porém aumenta-se em muito a penalidade por cada demanda não atendida, fazendo com que a remoção de qualquer demanda pequena seja muito mais importante que o número de motoristas.

Essas oscilações fazem com que a busca passe por regiões não atrativas e, ao final da fase de oscilação, a solução é reavaliada com a função objetivo padrão e a busca caminha novamente do modo usual. É desejável que neste caminho o procedimento consiga fugir do mínimo local e ainda aproveitar-se das boas características obtidas na fase de oscilação.

A Busca Tabu melhorou em muitos seus resultados com a adoção destas oscilações, demonstrando que, mesmo obtendo soluções aparentemente boas, o algoritmo pode não estar explorando todo o espaço de busca.

10.4 O Algoritmo

No algoritmo 4 está apresentada a versão final do procedimento de Busca Tabu.

Algoritmo 4 Busca Tabu para o PPV

1: **procedimento** BUSCATABU(*parametros*)
2: Crie solução S pelo método de GeraçãoHeurística
3: Avalie a solução S
4: $S_{melhor} \leftarrow S$
5: $iteracoesSemMelhora \leftarrow 0$
6: **enquanto** $nro\_iteracao < parametros.totalIteracoes$ **faça**
7: Busque o melhor movimento m_1 na vizinhança de Inserção e Remoção, verificando os Tabus e Critérios de Aspiração
8: Busque o melhor movimento m_2 na vizinhança de Troca de Motoristas, verificando os Tabus e Critérios de Aspiração
9: ▷ A dimensão buscada na vizinhança de Trocas de Jornadas é $dimDeBusca$
10: $dimDeBusca \leftarrow nro\_iteracao \pmod 8 + 1$
11: Busque o melhor movimento m_3 na vizinhança de Trocas de Jornadas, usando $dimDeBusca$ e verificando os Tabus e Critérios de Aspiração
12: Escolha o melhor dos movimentos m_1, m_2 e m_3
13: Aplique o movimento em S obtendo S_m
14: Avalie a solução S_m
15: **se** S_m melhor que S_{melhor} e não esta oscilando **então**
16: $S_{melhor} \leftarrow S_m$
17: $iteracoesSemMelhora \leftarrow 0$
18: **senão se** não esta oscilando **então**
19: $iteracoesSemMelhora \leftarrow iteracoesSemMelhora + 1$
20: **fim se**
21: **se** $iteracoesSemMelhora = numeroMaximoIteracoesSemMelhora$ **então**
22: Mantenha os pesos da função objetivo alterados durante $NumeroIteracoesOscilacao$
23: **fim se**
24: $S \leftarrow S_m$
25: **fim enquanto**
26: **retornar** S
27: **fim procedimento**

10.5 Detalhes de Implementação

Como parte do estudo para o projeto, foram vistos FrameWorks [GS01] e técnicas orientadas a objetos para modelagem de algoritmos de busca local e metaheurísticas [ACR98]. Esse estudo permitiu que houvesse reaproveitamento dos códigos entre as implementações e, desse modo, foi possível implementar algoritmos diferentes para resolver o problema. Houve grande uso de classes e interfaces, e somente optou-se por uma modelagem mais estruturada quando se desejava maior desempenho. Este ponto não se constituiu em problema em nenhum dos algoritmos, como poderá ser visto na seção de testes. A modelagem da solução recebeu maior atenção pois houve um esforço cooperativo com outro aluno de iniciação cientifica [Cir04] para que as estruturas fossem únicas, viabilizando futuras hibridizações.

11 A Busca Tabu Paralelizada

11.1 Introdução

A Busca Tabu seqüencial, ou monoprocessada, obteve resultados bastante bons para o PPV. No entanto, por se tratar de um problema difícil, e com um domínio muito extenso, ainda existem algumas deficiências, as quais estão intimamente ligadas à qualidade da soluções e a agressividade da busca.

Um dos pontos já notados foi a dificuldade em se realizar diversificação, evidenciada pelas melhorias obtidas após a implementação da oscilação estratégica. Isso demonstra que a lista tabu sozinha não foi suficiente para levar à busca a novas regiões promissoras do espaço de busca.

Pretende-se, com um modelo paralelizado, aumentar a capacidade de exploração da Busca Tabu, e, conseqüentemente, melhorar a qualidade das soluções geradas.

Na subseção seguinte, será apresentada uma taxonomia para as possíveis estratégias de paralelização, e a partir disso, poderá ser melhor definida uma modelgem para a Busca Tabu Paralelizada aplicada ao PPV.

11.2 Estratégias

As arquiteturas paralelas oferecem a possibilidade de aumento da eficiência dos algoritmos. Esse aumento é visto normalmente como a aceleração de algum trecho do algoritmo, ou também como uma nova forma de modelagem do problema. Em metaheurísticas, essas duas alternativas são aplicáveis, e cada uma delas possui sua própria representação.

Buscando classificar e demonstrar as possibilidades de paralelização possíveis para a Busca Tabu, uma taxonomia foi proposta [CTG97]. Nela, buscou-se detalhar as possibilidades de modelagem tanto em *baixo nível* quanto em *alto nível*. Entende-se como de *baixo nível* as modelagens que buscam o aumento da eficiência da Busca Tabu, não considerando a interação entre suas partes, e então não se diferenciando do modelo seqüencial, sendo somente mais rápidas. Como exemplo clássico dessa idéia, pode-se fazer o particionamento da vizinhança, realizando a avaliação do melhor movimento em paralelo. Essa é uma abordagem efetiva pois a avaliação da vizinhança é em geral a parte mais custosa da Busca Tabu.

Consideram-se entre os aspectos de uma modelagem de *alto nível*: o tipo de comunicação, o controle sobre a busca e a oferta do conhecimento às *threads*. Nesses aspectos enquadram-se as questões relativas à troca e ao tratamento da informação entre os processos, centrais na modelagem de qualquer procedimento paralelizado, as quais também tomam uma posição de maior relevância nas estratégias da Busca Tabu Paralelizada.

11.2.1 Taxonomia

Ao se definir a taxonomia, não desejou-se classificar as Buscas Tabu Paralelizadas pelo modo através do qual elas fazem a exploração básica e adquirem o conhecimento. De outro lado, como esses pontos são cruciais, eles servirão como critérios nesta taxonomia. Além do mais, qualquer

estratégia de paralelização implica em uma decomposição do domínio de atuação, dos passos ou tarefas dos algoritmos.

Para cobrir todos esses fatores foram propostas três dimensões. As duas primeiras representam os esquemas de controle da trajetória de busca e a forma de troca e tratamento da informação, enquanto a última especifica as formas de partição do domínio e parametrização de cada busca. As três dimensões estão ilustradas na tabela 3, e serão melhor descritas nas seções seguintes.

Cardinalidade do Controle	1-controle p-controle
Controle e Tipo de Comunicação	Sincronização Rígida Sincronização do Conhecimento Coletivo Conhecimento Coletivo
Diferenciação da Busca	UPUE UPME MPUE MPME

Figura 3: Tabela - Taxonomia proposta por [CTG97]

11.2.2 Cardinalidade de Controle

O controle sobre a busca deve ficar ou com um processador, usualmente chamado de *master*, ou distribuído entre diversos processadores. Assim, duas categorias são definidas:

- **1-controle:** Neste caso encontra-se o paralelismo *low level* citado anteriormente. Nele, o *master* delega aos outros processadores tarefas custosas, como a avaliação da vizinhança.
- **p-controle:** No segundo caso, quando o controle é divido em p, $p > 1$, processadores, existem *multithreads*, cada qual com sua própria busca. Assim, a coordenação das buscas e a garantia de disponibilidade das informações são as principais questões neste contexto.

11.2.3 Controle e Tipo de Comunicação

Esta segunda dimensão da taxonomia se baseia no tipo e flexibilidade do controle. Ela leva em conta a organização da comunicação, a sincronização e a hierarquia, e também a forma na qual a informação é processada e divida pelos processos. Os quatro graus dessa dimensão, combinados com os dois de controle, definem as estratégias de manipulação e processamento da informação.

- **Sincronização Rígida:** A operação de sincronização ocorre quando todos processos, após um determinado número de iterações ou tempo, devem parar e entrar numa fase de troca e atualização de informações. A qualidade rígida vem da pouca ou nenhuma quantidade de informações trocadas. Para o *1-controle*, um exemplo é a paralelização de alguns passos da

busca seqüencial, existindo sempre uma sincronização no fim do trecho paralelizado, sendo que não há praticamente nenhuma informação trocada na execução do trecho. No caso do *p-controle*, a sincronização rígida é demonstrada pela implementação trivial, onde diversas *threads* executam a busca e sincronizam no final para definir a melhor solução encontrada.

- **Sincronização do Conhecimento:** Ainda sendo uma operação sincronizada, aqui é permitido um nível maior de troca de informações. No contexto de *1-controle*, as tarefas delegadas pelo *master* são maiores, como alguns passos da busca sobre um subconjunto da vizinhança. No entanto, os *slaves* continuam não se comunicando entre si. Para o *p-controle*, essa estratégia se dá quando todas as buscas independentes param em um determinado número de iterações ou tempo, e iniciam uma fase de intensa comunicação, depois voltando à execução da busca.

- **Coletivo:** Os dois últimos graus são caracterizados pela operação assíncrona. O grau *coletivo* se dá quando a busca, ao encontrar uma solução com melhoria, envia esta solução para todos seus vizinhos ou para uma memória central. Eventualmente, pode-se optar por enviar a solução somente quando esta melhorar a melhor solução já encontrada. A comunicação é dita simples, pois a mensagem enviada é a mesma recebida pelas outras buscas, sem agregar informações inferidas sobre ela.

- **Conhecimento Coletivo:** A diferença deste grau com o anterior, está unicamente na inferência de informações sobre as mensagens enviadas. Essas conclusões são tiradas examinando-se a trajetória de todas as buscas, através de memórias de freqüência sobre atributos das soluções obtidas. Deste modo, a solução recebida contém mais informações do que quando foi enviada, pois a memória global é acrescentada.

11.2.4 Estratégia de Diferenciação da Busca

A última dimensão considerada permite classificar as Buscas Tabus Paralelizadas de acordo com o número de soluções iniciais e o número de diferentes estratégias de busca que utiliza. Esse termo, "estratégia de busca", será usado de maneira global e incluirá as diferentes definições de vizinhança, configurações de parâmetros, regras de gerência de memória, esquemas de diversificação, etc. A seguir são citados os quatro graus de diferenciação sobre a multiplicidade de solução iniciais e estratégias de busca, os quais são auto-explicativos.

- **UPUE:** ou Único Ponto (de Partida) e Única Estratégia (de Busca).

- **UPME:** ou Único Ponto e Multiplas Estratégias.

- **MPUE:** ou Múltiplos Pontos e Única Estratégia

- **MPME:** ou Múltiplos Pontos e Multiplas Estratégias

11.3 Estratégia Utilizada

Sobre a modelagem utilizada para a busca seqüencial existem dois fatos importantes e relevantes que influem na estratégia a ser proposta: a vizinhança modelada é completamente avaliada e não se apresentou como um gargalo para o procedimento, e o cálculo da função objetivo foi feito de forma otimizada. Assim sendo, não existem motivações explícitas para se construir uma modelo *low level*, já que não há grandes gargalos no procedimento. Isso ainda pode ser notado pelo baixo tempo de execução necessário, em torno de 3 minutos, para que o algoritmo alcance uma boa solução.

Deste modo, a intensificação do algoritmo foi feita de forma eficiente, conseguindo superar rapidamente as restrições fortes e otimizar a maioria das restrições fracas. Nesta visão, a performance foi satisfatória.

Por outro lado, um grande problema enfrentado pela versão monoprocessada, evidenciado na subseção 10.3, era a atratividade à certas regiões, criada pela função objetivo. Numa tentativa de resolver este problema, foi desenvolvida uma oscilação estratégica, que obteve bons resultados, quando comparado aos outros métodos e à versão sem oscilação. Sendo este componente um diversificador, ficou claro que a busca seqüencial é deficiente na exploração do espaço de busca.

Esses fatos motivaram a modelagem de uma estratégia paralelizada que explorasse uma maior extensão do espaço de busca. Para tal, um modelo *Multithreads*, **1-controle**, é proposto, onde um processador *master* mantém um "Elite Pool" com as melhores soluções já encontradas pelos *slaves*, os quais possuem uma busca própria. Assim, cada *thread* terá seus próprios parâmetros e comportamento, além de iniciarem em pontos diferentes, caracterizando um **MPME**.

A implementação da oscilação estratégica foi modificada. Agora ela não mais será evidenciada como fases no algoritmo, mas estará presente no contexto global de todas as buscas, através da utilização de diferentes funções objetivo para cada *thread*. Para manter o conceito de oscilação, será usado o componente de "Path Relinking", que fará a recombinação das soluções nas *threads*.

Como cada *thread* receberá e enviará soluções ao *master* de forma diferenciada, a comunicação será assíncrona e a informações trocadas serão soluções, enquadrando o procedimento como **Coletivo**.

Nas seções seguintes serão apresentados o protocolo de comunicação utilizado, o componente de "Path Relinking", uma descrição do "Elite Pool", da nova oscilação estratégica e, finalmente, o pseudo-código da Busca Tabu Paralelizada.

11.4 Protocolo de Comunicações para o PPV

Para implementar a estratégia definida, deve-se escolher entre dois paradigmas de programação paralela: troca de mensagens e memória compartilhada. Como foi utilizado um paralelismo de alta granularidade, onde a comunicação é feita principalmente pela troca de soluções, a troca de mensagens mostra-se mais apropriada para a estratégia descrita.

Além disso, o paradigma de programação paralela necessita de um ambiente tecnológico específico, envolvendo um conjunto de hardware e software que interagem de forma ampla e complexa. Para abstrair grande parte desta complexidade, que envolve, dentre muitos itens, o gerenciamento de rede, a comunicação de baixo nível e a implementação dos conceitos básicos do paradigma, no

caso escolhido o de troca de mensagens, já existem tecnologias desenvolvidas. As principais tecnologias utilizadas em metaheurísticas paralelizadas são o *MPI* (*Message Passing Interface*) e *PVM* (*Parallel Virtual Machine*). Cada qual traz consigo uma série de conceitos e definições próprias, e acoplá-los à lógica de funcionamento da metaheurística tornaria a aplicação não-portável e de difícil manutenção.

Desta forma, surge a necessidade de criarmos um conjunto de funções básicas capazes de abstrair da aplicação toda a lógica de troca de mensagens, que é dependente da tecnologia usada. Ao mesmo tempo, ela deve ser capaz de prover um tratamento centralizado e uniforme para a comunicação, simplificando ao máximo a implementação dos formatos desejados de paralelismo.

Foi desenvolvido, então, um *protocolo de comunicações* para assumir estas responsabilidades, em conjunto com [Cir04], bastante simples e baseado no MPI, denominado *PC-PPV* (*Protocolo de Comunicações para o PPV*). Além de prover todo o conjunto de funções necessárias para as mais diversas estratégias, ele é completamente independente da metaheurística.

Isto implica que o protocolo também pode ser utilizado para paralelizar facilmente outras técnicas, além de servir como ponte de comunicação para o desenvolvimento de modelos híbridos. Exemplos de outras metaheurísticas que também usaram o PC-PPV podem ser vistas em [Cir04].

11.4.1 Entidades

Todas as operações do protocolo estão associadas à quatro entidades: o *Comunicador*, a *MensagemPPV*, a *Requisição* e o *Recibo de Requisição*.

O *Comunicador* consiste na entidade central do protocolo e é a partir dele que toda a comunicação se realiza. Inicialmente, todos os processadores devem se registrar a um Comunicador válido que irá compor uma subrede fechada, ou seja, os processadores de um Comunicador não podem trocar mensagens com os de outros Comunicadores[1]. Na prática, usa-se apenas um Comunicador e a topologia da rede fica a cargo da camada de aplicação.

Já a *MensagemPPV* representa qualquer tipo de mensagem que pode trafegar na rede. Ela é composta por quatro campos: o *nó de origem*, o *nó de destino*, um *rótulo* e uma *mensagem empacotada*. Além disso, uma MensagemPPV também deve possuir duas operações associadas: uma contendo o procedimento necessário para empacotar a mensagem como uma string de *bits*, e outra para desempacotá-la. Tais operações são necessárias para tornar o protocolo independente da tecnologia de rede usada, mesmo se a mensagem consistir de um simples número inteiro.

Assim, para criar novas mensagens, basta definir um *rótulo* único e implementar suas duas funções de empacotamento e desempacotamento. A primeira mensagem criada foi a *Mensagem-Solução*, contendo as informações necessárias de toda uma solução do PPV. Para isso, bastou especificar em sua função de empacotamento que os dados consistiam apenas dos índices das jornadas de motoristas referentes à Enumeração, e o inverso no desempacotamento.

A *Requisição* representa uma solicitação de envio ou recebimento de entidades do tipo MensagemPPV, e está associada a uma fila assíncrona de mensagens. Uma vez executada a operação de envio de uma mensagem, a MensagemPPV entra nesta fila e o processador recebe um *Recibo*

[1]Definido por padrão, mas este tipo de privilégio pode ser fornecido a certos processadores, caso necessário

de Requisição, para checar se a mensagem foi ou não enviada e para que possa liberar seu *buffer* de memória associado. No caso do recebimento, o *Recibo de Requisição* serve para verificar se a MensagemPPV já foi recebida, para então utilizar seu *buffer* para extrair as informações desejadas.

O uso de um *Recibo de Requisição* permite que o protocolo só realize a troca de mensagens quando tanto o nó enviador como o receptor estão prontos, utilizando uma mensagem especial bastante pequena para efetuar este controle. Esta técnica reduz bastante o tráfego na rede e não é explícita para o usuário, geralmente já estando associada também à tecnologia de troca de mensagens.

11.4.2 Operações

Há seis tipos de operações fundamentais associadas ao Comunicador e, portanto, ao protocolo, descritas a seguir. A partir delas, todas as demais funcionalidades podem ser implementadas.

1. **Inicialização:** Esta operação deve ser realizada logo no início da aplicação. Ela registra o processador no Comunicador e atribui a ele um identificador único de nó, utilizado para enviar e receber mensagens.

2. **Envio de Mensagens:** Antes do envio, deve-se instanciar uma MensagemPPV e associá-la ao objeto no qual ela se refere (por exemplo, uma solução no caso da MensagemSolução). Além disso, seus outros campos devem ser preenchidos, como nó de origem (que é o identificador obtido no final da inicialização) e rótulo da mensagem.

   Em seguida, quando o Comunicador for requisitado a enviá-la, ele irá utilizar a função de empacotamento da mensagem para alocá-la em um *buffer* de bits. Depois, associará este *buffer* a uma Requisição, que irá inserí-la em uma fila de envio de mensagens. Por fim, será retornado um Recibo de Requisição para a aplicação.

   O objeto que representa a MensagemPPV só pode ser desalocado da memória a partir do momento que o Recibo de Requisição indicar que o envio foi bem sucedido. Isto porque várias mensagens podem estar associadas à fila da mesma Requisição, o que não garante que o envio será imediato[2].

3. **Recebimento de Mensagens:** No recebimento, pode-se especificar o rótulo e o nó de origem da mensagem que se deseja receber, porém estes campos são opcionais e é possível receber qualquer tipo de mensagem e de qualquer processador, caso sejam omitidos. Neste caso, é de responsabilidade do Comunicador identificar seu tipo e instanciar os objetos adequados para a aplicação.

   De forma semelhante ao envio, um *buffer* vazio é associado à uma Requisição que, por sua vez, está vinculada a uma fila de recebimento de mensagens, além de retornar também o Recibo de Requisição. No momento em que a mensagem for recebida, ela será desempacotada do *buffer* pelo Comunicador (com a função especificada pelo tipo de mensagem) e o Recibo de Requisição será modificado para alertar o recebimento.

[2]Existe, também, um limite de 200 mensagens na fila de envio/recebimento de uma Requisição

4. **Aguardo de Alerta de Recibos:** Todas as operações descritas até agora são assíncronas; a partir do momento em que a aplicação adquiriu o Recibo de Requisição de envio ou recebimento de mensagens, ela está livre para executar outros processos. Cabe a ela, portanto, verificar seus Recibos nos momentos em que julgar adequado.

   Porém, em determinadas modelagens, é importante receber ou enviar mensagens de forma síncrona, principalmente quando as informações a receber são essenciais para a continuidade do processamento. Assim, foi criada uma função especial na qual a aplicação entra em estado de espera até que todos os Recibos indiquem a efetivação da Requisição, abstraindo tal lógica da aplicação.

   Há três tipos possíveis de espera: aguardar a efetivação de apenas um Recibo, aguardar a efetivação de todos os Recibos de um conjunto ou, por fim, aguardar a efetivação de pelo menos um ou mais Recibos de um conjunto.

5. **Broadcast:** Esta operação é análoga ao envio, porém as mensagens são enviadas para todos os nós pertencentes ao Comunicador. A vantagem de criar uma operação exclusiva para isso é que, na maioria das vezes, a tecnologia de troca de mensagens possui funções otimizadas de *broadcast* de mensagens.

6. **Finalização:** Na finalização, é removido o registro do processador no Comunicador e o identificador é desalocado.

Todos os detalhes das operações acima podem ser abstraídos durante a implementação das metaheurísticas paralelas, e foram descritos apenas para fundamentar seu funcionamento básico. Na prática, seu uso se resume em invocar funções como *enviar*, *receber* ou *aguardar*, com parâmetros muito simples e sem se preocupar com detalhes de configuração do *cluster* de processadores.

Com este protocolo implementado e testado para ser utilizado com o *MPI*, foi possível desenvolver muito mais rapidamente o modelo proposto, já que ele foi capaz de prover um nível de abstração bastante alto na aplicação.

A seguir serão apresentados outros dois componentes básicos para o modelo paralelo, que, em conjunto com o protocolo apresentado, já serão suficientes para implementar a estratégia paralelizada.

11.5 O "Elite Pool"

Um "Elite Pool" deve manter armazenadas as melhores soluções obtidas pela busca. Para a estratégia proposta, necessitamos, além de manter as melhores soluções de um procedimento, saber qual foi a melhor solução encontrada. Para tal, foi utilizada uma estrutura com um "Global Elite Pool" e um "Local Elite Pool" para cada *thread*. Em termos de implementação, foram utilizadas listas ordenadas pela valor da função objetivo da *thread* e uma lista ordenada pelo valor de uma função objetivo global com as soluções de todas as outras listas. Vale lembrar que esta memória estará localizada no *master*.

11.6 O "Path Relinking"

Proposto inicialmente por Glover[GL97], o componente de "Path Relinking", doravante chamado de PR, fornece a possibilidade de se explorar trajetórias entre duas soluções no espaço de busca. Em linhas gerais, realiza-se a exploração partindo de uma solução inicial e, iterativamente, aplicam-se transformações até que ela se torne uma outra solução, chamada de guia. Espera-se que durante esse processo seja encontrada uma solução ainda melhor do que a inicial e a guia. Naturalmente, ao se implementar um PR devem ser definidas quais as transformações que serão aplicadas de forma que os atributos da solução guia sejam incorporados gradativamente à solução inicial.

Havendo a necessidade de reaproveitamento das soluções trocadas entre as *threads* para a estratégia de paralelismo proposta, o PR será utilizado como um operador de recombinação entre soluções, assim como o *crossing over* usado em algoritmos genéticos [Cir04].

Já tido como efetivo em outras aplicações semelhantes [HG04], o "Path Relinking" será aplicado sobre a solução corrente de uma *thread* e a solução recebida do "Elite Pool", sendo a última a melhor encontrada até o momento por uma outra *thread*. Assim, poderão ser agregados atributos à solução corrente que uma função objetivo diferente levou outra busca a obter.

Algoritmo 5 "Path Relinking" entre duas soluções do PPV

```
1: procedimento PathRelinkingPPV(solucaoInicial,solucaoGuia)
2:     valorInicial ← avalia(solucaoInicialAlterada)
3:     Para i faça0MIN(totalOnibus(solucaoInicial),totalOnibus(solucaoGuia))
4:         (onibusDaInicial,onibusDaGuia) ← buscaParMaisSemelhante(solucaoInicialAlterada,solucaoGuia)
5:         trocaOnibus(onibusDaInicial,onibusDaGuia,solucaoInicialAlterada)
6:         se valorInicial > avalia(solucaoInicialAlterada então
7:             retornar solucaoInicialAlterada
8:         fim se
9:     fim Para
10:    Para todos onibusDaGuia a mais que totalOnibus(solucaoInicialAlterada) e nao presente na
   solucaoInicialAlterada faça
11:        insereOnibus(onibusDaGuia,solucaoInicialAlterada)
12:        se valorInicial > avalia(solucaoInicialAlterada) então
13:            retornar solucaoInicialAlterada
14:        fim se
15:    fim Para
16:    Para todos onibusDaInicial a mais que totalOnibus(solucaoGuia) e nao presente na solucaoGuia
   faça
17:        removeOnibus(onibusDaInicial,solucaoInicialAlterada)
18:        se valorInicial > avalia(solucaoInicialAlterada) então
19:            retornar solucaoInicialAlterada
20:        fim se
21:    fim Para
22:    se avalia(solucaoInicial) < avalia(solucaoGuia) então
23:        retornar solucaoInicial
24:    senão
25:        retornar solucaoGuia
26:    fim se
27: fim procedimento
```

O PR implementado realiza três tipos de transformações: troca, inserção e remoção, onde o elemento a ser transformado na solução é um ônibus. Quando se diz que o elemento é o ônibus, entende-se que todas as informações do ônibus estão aí compreendidas (Ver estrutura da solução na figura 1), tais como os motoristas e as viagens. Assim, o PR iterativamente transforma os ônibus da solução inicial naqueles da solução guia.

O pseudo-código do PR é mostrado no algoritmo 5. Nele, quando as duas soluções fornecidas para o PR possuem o mesmo número de ônibus, só será utilizada a operação de troca. Caso existam mais ônibus na solução guia do que na inicial, haverá mais um passo onde são inseridos os ônibus da solução guia que não foram alvo de trocas. No caso contrário, onde há menos ônibus na solução guia, aqueles que não foram alvo de trocas na solução inicial, serão retirados. O PR, em seu todo, finaliza quando consegue melhorar a solução inicial.

Por fim, o PR será utilizado em conjunto com o "Elite Pool" e a diferenciação de funções objetivo para implementar o conceito da oscilação estratégica, que é o pilar central da estratégia proposta.

11.7 A Oscilação Distribuída

O conceito de uma oscilação estratégica está em permitir que a busca explore espaços inviáveis de uma forma controlada. Espera-se que, desta forma, a busca possa, após a fase de oscilação, percorrer uma trajetória que a levará à uma região que era de difícil acesso. Pode-se também visualizar esta fase de oscilação como uma transformação do espaço de busca, criando uma relevo novo. Neste novo relevo a busca realizará um nova otimização, que não necessariamente levará a soluções inviáveis.

Para uma série de problemas, onde o PPV está incluso, a função objetivo depende da definição dos pesos sobre certas restrições. Essas são divididas em dois tipos: restrições fortes, que implicam em inviabilidade, e restrições fracas. Sem contar algumas restrições fortes, que podem ser diretamente eliminadas pela modelagem, a distinção entre restrições fortes e fracas está na atribuição de pesos bem maiores às restrições fortes.

Quando existem muitas restrições, a atribuição de pesos se torna uma tarefa complexa e, em geral, exige a realização de experimentos para a definição dos pesos que serão utilizados em uma versão final. Nota-se que os pesos têm grande influência sobre os procedimentos, já que eles, no fundo, definem o relevo do espaço de busca.

A versão monoprocessada do algoritmo demonstrou que a implementação proposta obteve melhorias quando utilizou a oscilação estratégica. Sendo assim, seria interessante manter este conceito na versão paralela. Entretanto, como houve uma mudança de paradigma, a implementação da oscilação foi reavaliada, e um novo modelo, específico para a versão paralelizada, foi identificado.

O novo modelo de oscilação estratégica, apelidado de *oscilação estratégica distribuída* ou *oscilação distribuída*, foi implementado de maneira simples, sem a divisão da busca em duas fases. Para tal, as *threads* receberam, em conjunto com seus parâmetros de busca diferenciados, diferentes pesos para a função objetivo. Além disso, e alternativamente ao início da fase de oscilação, as *threads* recebem uma solução do "Elite Pool", localizado no *master*, e fazem a recombinação da solução corrente com a recebida (em nosso caso, será utilizado o "Path Relinking" para a recombinação).

Caso a recombinação não obter uma solução melhor, a busca continuará, usando a solução recebida. Isso é equivalente à fase de oscilação, já que, dadas as diferentes funções objetivo utilizadas, as soluções recebidas são semelhantes às obtidas no final da fase de oscilação. A *oscilação distribuída*, ainda tem a vantagem de ser mais eficiente, não necessitando de nenhuma iteração da busca em si para obter tal solução.

Uma grande vantagem da *oscilação distribuída* é que a presença de diferentes pesos para a função objetivo, antes um problema, agora se torna uma aliado na exploração do espaço de busca. Neste contexto, algumas buscas terão a característica tradicional de oscilação estratégica, onde suas soluções estarão em regiões inviáveis, enquanto, outras buscas terão os pesos ligeiramente diferentes porém sempre de maneira coerente com as restrições. Serão essas últimas buscas as que efetivamente encontrarão uma boa solução para o problema.

11.8 Pseudo-Códigos

Após a definição da estratégia de forma sucinta e o detalhamento de cada um de seus componentes, podemos apresentar os pseudo-códigos finais nos algoritmos 6 e 7, da *thread master* e das *slaves*, respectivamente.

Algoritmo 6 *Thread master* da Busca Tabu Paralelizada

```
 1: procedimento BuscaTabuMaster(parametros)
 2:     total_threads_executando ← parametros.total_nodos − 1
 3:     enquanto total_threads_executando > 0 faça
 4:         Para todos thread faça
 5:             se RecebeuMensagemComSolução(thread) então
 6:                 mensagem ← ReceberMensagem(thread)
 7:                 adicionarSoluçaoAoPool(thread,mensagem.solucaoEnviada)
 8:                 solucaoPedida ← ElitePool.PegarMelhor(mensagem.threadPedida)
 9:                 enviarSolucao(thread,solucaoPedida)
10:             fim se
11:             se RecebeuMensagemDeFinalizacao(thread) então
12:                 total_threads_executando ← total_threads_executando − 1
13:             fim se
14:         fim Para
15:     fim enquanto
16:     retornar S
17: fim procedimento
```

Algoritmo 7 *Thread slave* da Busca Tabu Paralelizada

1: **procedimento** BUSCATABUSLAVE(*parametros*)
2: Crie solução S pelo método de GeraçãoHeurística
3: Avalie a solução S
4: $S_{melhor} \leftarrow S$
5: $iteracoesSemMelhora \leftarrow 0$
6: **enquanto** $nro\_iteracao < parametros.totalIteracoes$ **faça**
7: Busque o melhor movimento m_1 na vizinhança de Inserção e Remoção, verificando os Tabus e Critérios de Aspiração
8: Busque o melhor movimento m_2 na vizinhança de Troca de Motoristas, verificando os Tabus e Critérios de Aspiração
9: A dimensão buscada na vizinhança de Trocas de Jornadas é $(nro\_iteracao mod 8) + 1$
10: Busque o melhor movimento m_3 na vizinhança de Trocas de Jornadas verificando os Tabus e Critérios de Aspiração
11: Escolha o melhor dos movimentos m_1, m_2 e m_3
12: Aplique o movimento em S obtendo S_m
13: Avalie a solução S_m
14: **se** S_m melhor que S_{melhor} **então**
15: $S_{melhor} \leftarrow S_m$
16: $iteracoesSemMelhora \leftarrow 0$
17: **senão**
18: $iteracoesSemMelhora \leftarrow iteracoesSemMelhora + 1$
19: **fim se**
20: **se** $iteracoesSemMelhora = numeroMaximoIteracoesSemMelhora$ **então**
21: $thread$ = proximaThread()
22: $S_{recebida} \leftarrow$ EnviarReceberSolução(S_{melhor},$thread$) ▷ pede ao *master* uma solução de uma outra *thread*
23: $S_m \leftarrow$ PathRelinking(S_m,$S_{recebida}$)
24: $iteracoesSemMelhora \leftarrow 0$
25: **fim se**
26: $S \leftarrow S_m$
27: **fim enquanto**
28: **retornar** S
29: **fim procedimento**

12 A Distribuição dos Horários

12.1 Motivação

Na primeira fase do projeto focou-se muita na redução de custos que é refletida pelos ótimos resultados obtidos para número de ônibus, motoristas e de horas extras. No entanto, para o público usufruidor dos serviços de transporte coletivo, existe outro ponto de alta criticidade, a qualidade do serviço, que vai além dos custos de operação que têm sua parcela correspondente no bilhete pago. A qualidade do serviço é de extrema importância e engloba dois fatores: o pleno atendimento da demanda e a distribuição uniforme dos horários. Tanto o atendimento da demanda como a distribuição de horários influem significativamente na superlotação dos ônibus, causadora de grandes desconfortos para os usuários. Enquanto que o mal atendimento da demanda é causa direta deste problema, a distribuição de horários leva algumas viagens à superlotação, já que a má distribuição das viagens pode levar a má distribuição da demanda. Além disso, minimizar o tempo de espera entre viagens e reduzir o acúmulo de veículos em um PC também são objetivos da distribuição uniforme dos horários.

O atendimento da demanda já foi tratado com sucesso na primeira fase do projeto e somente a distribuição de horários não havia recebido um tratamento adequado. Nesta fase, e tal como no trabalho de [Vaz03], foi criado um módulo adicional para modificar escalonamentos já válidos de ônibus e motoristas, ganhando-se grande liberdade na modelagem dos demais subproblemas do PPV. Neste módulo são utilizadas uma Busca Local que opera sobre uma função objetivo própria para a distribuição de horários e uma heurística para complementar o procedimento.

Os resultados, mostrados na seção 13, demonstram que o módulo foi efetivo em melhorar a distribuição de horários. No entanto, em algumas faixas horárias, a distribuição não atingiu a qualidade desejada. Isto se deve à grande redução no número de veículos e motoristas, e também às restrições que a heurística respeita durante seu procedimento. Sobre estas constatações foram planejadas melhorias futuras.

12.2 A Busca Local para distribuição dos horários

O primeiro passo do módulo é, a partir de uma solução final obtida pela Busca Tabu monoprocessada ou paralela, aplicar uma Busca Local para melhorar a distribuição de horários sem deteriorar as otimizações já atingidas.

No entanto, os movimentos utilizados pela Busca Tabu não servem para o fim desta Busca Local, pois alteram todas as propriedades da solução, como demanda atendida, número de ônibus, motoristas e horas extras. Assim sendo, devemos definir outro tipo de movimento que restrinja as modificações à distribuição dos horários de partidas.

Como já explicado em tópicos anteriores, a enumeração de jornadas permite, através de seu esquema de dimensões, trocar jornadas contidas em uma solução por outras jornadas que mantenham um conjunto de propriedades intactas. Como no total temos 8 dimensões na Enumeração, poderíamos fixar as sete primeiras, o que acarretaria apenas em mudanças relativas ao fator de aleatoriedade das soluções.

Modificar o fator de aleatoriedade, a última dimensão, implica em alterarmos apenas os horários de partida dos motoristas sem que eles deixem de atender todas as suas faixas horárias originais, evitando a violação de qualquer outra restrição forte do problema.

Portanto, nesta abordagem, basta modelarmos uma nova função objetivo que considere apenas a distribuição de horários, já que as demais características da solução não sofrerão alterações e não há necessidade de reverificá-las.

12.3 Nova Função Objetivo

Um primeiro modelo trivial para avaliarmos a uniformidade da distribuição seria estabelecermos pontos fixos ideais em cada faixa horária e, assim, o objetivo seria minimizar os desvios de cada viagem em relação à esses pontos. Porém, como demonstrado em [Vaz03] com o exemplo da figura 4, as viagens não influenciam as adjacentes, e soluções menos uniformes poderiam ser priorizadas.

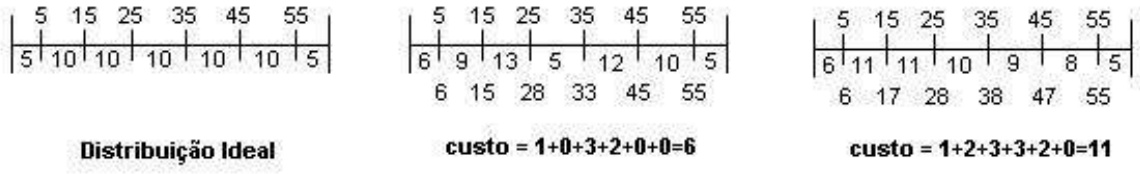

Figura 4: Figura de [Vaz03] mostrando custos equivocados por não considerar adjacência de horários de partida.

Portanto, a função adotada baseia-se no modelo linear[3] criado por [Vaz03] e consiste em minimizar as diferenças entre os espaçamentos consecutivos das viagens, os quais devem assumir valores absolutos para que o custo da solução seja adequadamente calculado.

Para tanto, definimos as seguintes variáveis inferidas a partir de uma solução válida do PPV:

- $v_{i,p}$: i-ésima viagem partindo do PC p;
- nv_p: número de viagens do PC p;
- $f_{i,p}$: faixa horária em que está a i-ésima viagem do PC p;

Temos, então, que calcular os espaçamentos consecutivos entre as viagens, obtido simplesmente pela diferença entre eles. Sendo $e_{i,p}$ o i-ésimo espaçamento do PC p, então:

$$e_{i,p} = (v_{i+1,p} - v_{i,p}), \forall i \in \{1, ..., nv_p - 1\}, \forall p \in \{1, 2\} \quad (1)$$

[3]Apesar de ser baseado neste modelo, a função objetivo resultante não será linear.

Porém, o espaçamento ideal não é o mesmo para todos os pares de viagens consecutivas, já que cada faixa horária possui um número diferente de viagens.

Seja $k_{f,p}$ este fator na faixa horária f e no PC p. Se $esp_{f,p}$ é o espaçamento ideal na faixa horária f, PC p, e c é uma constante positiva qualquer, então podemos definir o fator k como:

$$k_{f,p} = \frac{c}{esp_{f,p}}, \forall f \in \{0, ..., 23hs\}, \forall p \in \{1, 2\} \tag{2}$$

O valor do espaçamento ideal da faixa horária é calculado dividindo-se o tamanho em minutos da faixa horária pelo total de viagens naquela faixa e PC. Podemos, portanto, definir a constante c como o tamanho em minutos da faixa horária, fazendo com que $k_{f,p}$ seja simplesmente o número total de viagens na faixa f e PC p pela equação 2, eliminando a divisão.

Adicionando então o fator k na equação 1 e sendo ini_f o minuto inicial da faixa horária f, temos:

$$e_{i,p} = (v_{i+1,p} - ini_{f_{i+1},p}).k_{f_{i+1},p} - (ini_{f_i,p} - v_{i,p}).k_{f_{i,p}}, \forall i \in \{1, ..., nv_p - 1\}, \forall p \in \{1, 2\} \tag{3}$$

Note que, se $f_{i+1,p} = f_{i,p}$, então $ini_{f_{i+1},p} = ini_{f_i,p}$ e $k_{f_{i+1},p} = k_{f_{i,p}}$, e a equação 3 torna-se igual à equação 1 com o lado direito multiplicado pelo fator k.

Por fim, a função objetivo será minimizar tais espaçamentos, ou seja, minimizar o valor absoluto das diferenças entre eles. Isto é:

$$F = min \sum_{i=1}^{nv_p-2} \sum_{p=1}^{2} |e_{i+1,p} - e_{i,p}| \tag{4}$$

A função F é então utilizada para calcular a uniformidade d a distribuição de horários na Busca Local, realizando uma pós-otimização na solução final da Busca Tabu.

12.4 Heurística para distribuição dos horários

Após a implementação da Busca Local, foram realizados testes e os resultados demonstraram que a Busca Local, sozinha, não estava sendo suficiente para realizar uma distribuição uniforme dos horários. Essa limitação do procedimento é atribuída à incapacidade da enumeração em fornecer jornadas suficientemente diferenciadas em seus horários de partidas para realizar a distribuição. Sendo as partidas das jornadas já determinadas previamente, durante a geração da enumeração, se não for realizado um ajuste fino de cada horário de partida, o procedimento deverá encontrar uma combinação de partidas que dê uma boa distribuição. Esta tarefa mostrou-se difícil e levou a necessidade de se alterar os horários de partidas diretamente.

Como a Busca Local, a heurística também deverá respeitar as otimizações já obtidas, ou seja, manter o atendimento da demanda, o número de ônibus, motoristas e horas extras utilizadas. A heurística só realizará alterações nos tempos de partida das tarefas, não realizando inclusões ou remoções de tarefas ou jornadas. Desta forma, só será necessário se preocupar com o atendimento da demanda, pois se uma tarefa mudar de faixa horária, a demanda de sua faixa inicial, poderá

ficar subatendida. Para garantir que isso não ocorrerá, é necessário conhecer o quanto se pode deslocar cada tarefa para frente e para traz no tempo, de forma que nenhuma tarefa mude de faixa horária. As fórmulas abaixo demonstram o cálculo do minuto de partida mínimo de uma viagem i em uma jornada, o $P_{min}(i)$, e do minuto máximo de partida de uma viagem i em uma jornada, o $P_{max}(i)$. Para o P_{min}, o cálculo é feito da primeira tarefa da jornada até a última, enquanto que para o P_{max} de forma contrária. Nessas fórmulas $DFH(i)$ indica se a faixa horária i já teve seus horários de partida distribuídos e $FH(i)$ devolve a faixa horária da viagem i.

$$P_{min}(i) = \begin{cases} MinutoFinalDe(i) - Duracao(i) & \text{Se } i = 0; \\ P_{min}(i-1) + Duracao(i) & \text{Se } \neg DFH(i) \text{ e } FH(i-1) = FH(i); \\ MAX(FH(i) * 60, P_{min}(i-1) + Duracao(i)) & \text{Se } \neg DFH(i) \text{ e } FH(i-1) \neq FH(i); \\ MinutoFinalDe(i-1) & \text{Se } DFH(i) \text{ e } FH(i-1) = FH(i); \\ MAX(FH(i) * 60, MinutoFinal(i-1)) & \text{Se } DFH(i) \text{ e } FH(i-1) \neq FH(i). \end{cases}$$

$$P_{max}(i) = \begin{cases} (FH(i)+1) * 60 - 1 & \text{Se } i = n; \\ P_{max}(i+1) - Duracao(i) & \text{Se } \neg DFH(i) \text{ e } FH(i+1) = FH(i); \\ MAX(FH(i) * 60, P_{max}(i+1) - Duracao(i)) & \text{Se } \neg DFH(i) \text{ e } FH(i+1) \neq FH(i); \\ MinutoInicialDe(i+1) & \text{Se } DFH(i) \text{ e } FH(i+1) = FH(i); \\ MAX(FH(i) * 60, MinutoInicialDe(i-1)) & \text{Se } DFH(i) \text{ e } FH(i+1) \neq FH(i). \end{cases}$$

Nas equações passadas, usa-se a função $DFH(i)$ por quê a heurística realiza um processo iterativo onde as faixas horárias têm seus horários de partida distribuídos uma a uma. Assim, uma faixa horária que já tenha seus horários distribuídos não pode ter suas partidas afetadas pela distribuição realizada em outra faixa horária. Levando em isso em conta no cálculo do mínimo e máximo deslocamento, o procedimento respeitará a restrição.

Como já descrito, a heurística aplica um procedimento de distribuição de horários sobre cada faixa horária do dia. Este procedimento (veja o algoritmo 8) realiza uma otimização gulosa, tentando remover um minuto entre duas partidas e inserindo-o entre duas outras, enquanto isso melhorar a distribuição. A avaliação da melhoria é feita pela mesma função objetivo usada na Busca Local. Um ponto muito importante é a ordenação primária das partidas pelos seus P_{min} e secundária pelos P_{max}, gerando uma ordem entre as partidas, que deverá ser respeitada até o final do procedimento. A vantagem em se fazer esta ordenação é que as partidas com menor P_{min} conseguem se aproximar mais do ínicio da faixa, enquanto que as partidas com maior P_{max} conseguem se aproximar mais do fim da faixa. É por isso que essa ordenação parece ser mais promissora em distribuir as partidas.

13 Resultados Computacionais

Nesta seção serão apresentados e analisados os resultados computacionais coletados durante todo o projeto, que envolvem o Algoritmo de Busca Tabu monoprocessado, a versão paralelizada e também a distribuição de horários. Os resultados obtidos pelas duas versões da Busca Tabu foram melhores

Algoritmo 8 Distribuindo horários de partida das viagens

```
1: procedimento DISTRIBUIRPARTIDAS(horario)
2:     otimizou ← = verdadeiro
3:     ordenarPartidasPorMinMax(horario)
4:     melhorValor ← = avalia(horario)
5:     enquanto otimizou faça
6:         otimizou ← falso
7:         Para todos intervalo j entre partidas do horario partindo de i faça
8:             Remove um minuto no intervalo j
9:             Para todos intervalo k, onde k é diferente de j do horario partindo de i faça
10:                Insere um minuto no intervalo k
11:                novoValor ← avalia(horario)
12:                se novoValor < melhorValor então
13:                    melhorValor ← novoValor
14:                    otimizou ← verdadeiro
15:                    vai para 7
16:                senão
17:                    Remove um minuto no intervalo k
18:                fim se
19:            fim Para
20:            Insere um minuto no intervalo j
21:        fim Para
22:    fim enquanto
23: fim procedimento
```

em muitas instâncias se comparados aos outros procedimentos e o maior ganho observado foi a otimização da utilização de motoristas e veículos.

As implementações dos programas foram desenvolvidos em C++ e compilados usando o GNU-GCC-3.2.2. Todos os testes da versão monoprocessada foram realizados utilizando-se computadores com processadores *AMD Atlhon* de 1.67 GHZ e 512 MB de memória RAM. A versão paralelizada usou um cluster de seis workstations com dois processadores *Intel Xeon 2.4Ghz* e 1,0 GB de *RAM* cada.

13.1 Parametrizações

Refinar os parâmetros do algoritmo é uma tarefa de caráter empírico, que deve ser feita buscando o ajustamento do procedimento às diversas instâncias, mantendo um comportamento homogêneo e otimizado sobre os resultados. Manter essa uniformidade é importante para que o procedimento não fique adaptado apenas à algumas instâncias e tenha resultados ruins para outras. Para o algoritmo de Busca Tabu Monoprocessado, foram encontrados os parâmetros, apresentados na tabela 2, e que permitiram a obtenção de boas soluções para todas as instâncias disponíveis.

É importante notar que a tabela 2 fornece parâmetros que são compartilhados com os algoritmos de Busca Local e Busca Cega, e que também terão seus resultados analisados mais a frente.

Para a versão paralelizada, outros parâmetros devem ser definidos. Desta forma, e seguindo o modelo proposto, as *threads* realizarão a *oscilação distribuída* através do uso de diferentes funções objetivo. Para manter-se de acordo com a versão monoprocessada, que obteve bons resultados com

Parâmetro	Valor	Observação
Número de iterações	15000	Utilizado como critério de parada
Número máximo de iterações sem melhora	500	Após, inicia-se a oscilação
Número de iterações de oscilação	200	
Penalidade por restrição de demanda	15.000	
Penalidade por restrição de máximo de horas extras	15.000	
Custo por Veículo	500	
Custo por Motorista	250	
Custo por Passageiro Sub-alocado	10	
Custo por Passageiro Super-alocado	0,5	
Custo por Hora Regular Trabalhada	0,00001	
Custo por Hora Ociosa	0,00050	
Custo por Hora Extra	0,001	

Tabela 2: Parâmetros utilizados para os testes

os parâmetros de oscilação descritos na subseção 10.3, definiu-se três tipos de funções objetivo. A primeira tem os custos da tabela 2, sendo chamada de função objetivo "normal", enquanto as duas outras, chamadas de funções objetivo "oscilatórias 1 e 2", têm custo zero para a restrição de demanda e custo por passageiro sub-alocado 50 e 5, respectivamente.

Uma tarefa mais complexa é determinar qual a proporção entre threads com função objetivo "normal" e "oscilatórias" que serão usadas. Para tal, foram realizados testes com diferentes proporções dessas funções objetivo. A figura 5 demonstra o resultado obtido para essas proporções. Foram utilizados 10 processadores e as configurações foram numeradas de 1 a 11, sendo que a número 1 possui todas suas *threads* com funções objetivos "oscilatórias", metade de cada tipo. Assim, a número 2 possui uma função objetivo "normal", a 3 possui duas, e assim vai até a configuração 11 onde todas as funções objetivo são "normais".

Analisando a figura 5 nota-se que nos dois extremos, onde ou todas funções objetivos são, ou "normais" ou "oscilatórias, os resultados foram piores. Verifica-se claramente que a configuração 6, onde existem cinco *threads* "normais", três "oscilatórias" do tipo 1 e duas "oscilatórias" do tipo 2, foi a melhor obtida. Além disso, nas configurações 5, 7 e 9, onde temos um número igual de "oscilatórias" do tipo 1 e 2, a eficiência foi menor do que quando temos mais oscilatórias do tipo 1 (configurações 4,6 e 8).

As *threads* também terão um número máximo de iterações de execução e um número iterações sem melhoria, que será usado para indicar quando deve-se fazer uma comunicação com o *master*.

13.2 Convergência dos Algoritmos

Uma primeira análise realizada sobre os algoritmos foi a da convergência. Essa análise é útil para verificar se o algoritmo está conseguindo evoluir e pode ajudar a tomar decisões sobre a necessidade de maior diversificação ou intensificação no procedimento. A Busca Cega tem seu comportamento mostrado na figura 6. Foi verificado que a BC rapidamente retira todas as restrições fortes e, após isso, é ineficiente para otimizar os recursos, como hora extras, horas ociosas, motoristas e veículos. O gráfico da figura 6 mostra essa característica através da queda brusca no início e da

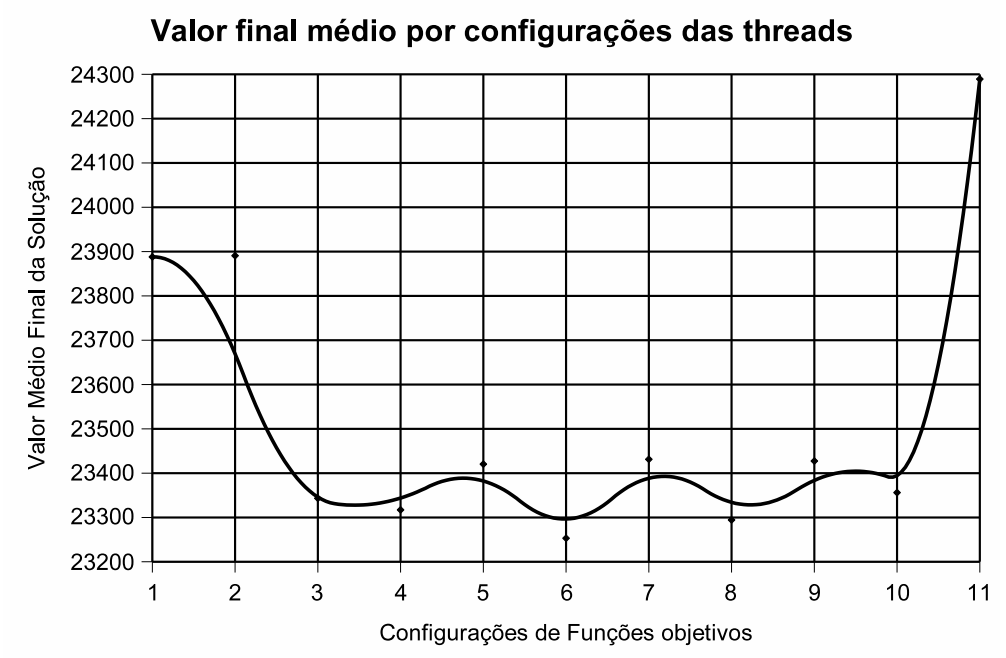

Figura 5: Valor médio final da função objetivo para diferentes configurações das *threads*

monotonicidade restante.

A primeira diferença a ser notada entre a convergência da Busca Cega e da Busca Local, é que a Busca Local reinicia a busca quando não consegue melhorar a solução por um número determinado de iterações. Podemos notar na figura 7 que, a cada reinício, a Busca Local volta a uma solução com pouca qualidade e realiza uma intensificação que rapidamente retira as restrições e consegue otimizar os recursos de forma mais satisfatória. No entanto, esses reinícios não conseguem obter uma solução muito melhor do que as encontradas nas primeiras 5000 iterações. Isso ocorre na maior parte das instâncias e demonstra que a BL não consegue diversificar o bastante, convergindo para regiões muito semelhantes, que rapidamente esgotam seus mínimos locais.

Para conseguir explorar novas regiões, e corrigir o problema encontrado na BL, a Busca Tabu utilizou-se da oscilação estratégica em conjunto com a Lista Tabu para implementar a diversificação. Na figura 8 com a convergência da BT, podemos ver os ínicios e fins das oscilações nos vários platôs que aparecem no gráfico. Esse formato é dado pela alteração dos parâmetros da função objetivo, que gera soluções com qualidade baixa se analisada com a função objetivo original. O gráfico ainda demonstra que a BT consegue, mesmo na iteração 10000, encontrar uma solução melhor e, em diversos momentos, ela fica próxima da melhor já encontrada.

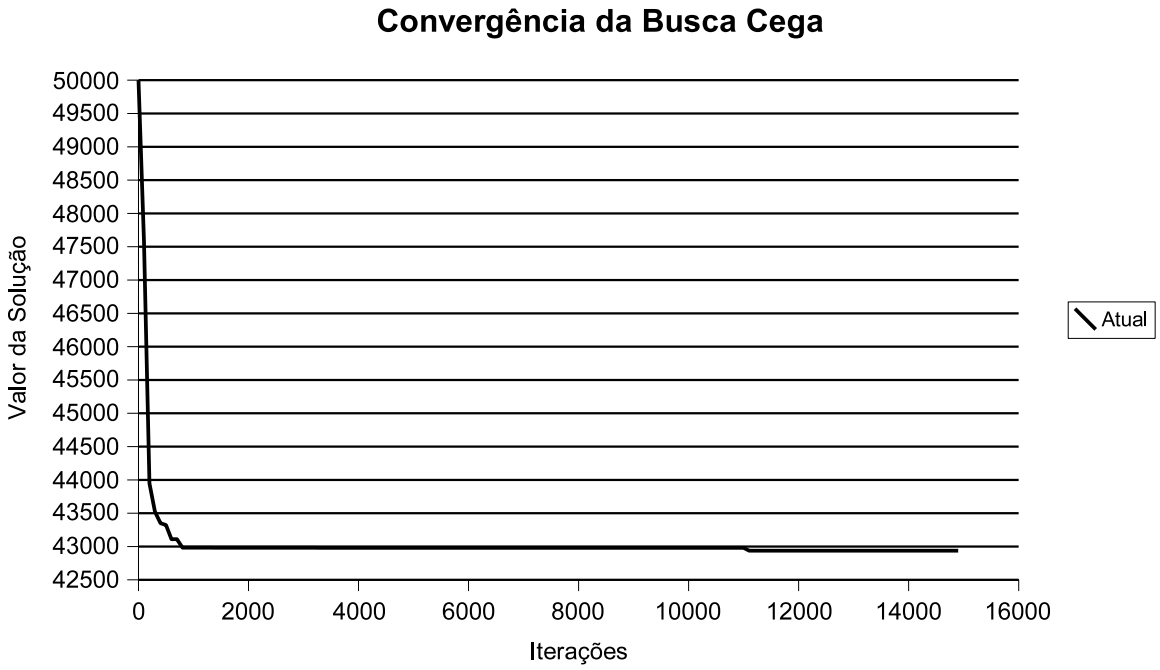

Figura 6: Convergência da Busca Cega

13.3 Aceleração do Paralelismo

Um modo comum para verificar a aceleração de um procedimento, após a passagem para o modelo paralelizado, é o "speedup". No caso da Busca Tabu Paralelizada, foi usada a versão onde as *threads* não se comunicam. Assim, para vários números de processadores são realizados testes, e são retirados os tempos médios para que se alcance uma determinada qualidade de solução. É esperado que com um número maior de *threads* a solução seja obtida com um tempo médio menor e ainda que seja inversamente proporcional ao número de processadores.

O resultado do "speedup" pode ser verificado na figura 9. Dela chegamos que o tempo necessário para chegar à uma certa qualidade de solução diminui pela metade cada vez que dobramos o número de processadores.

13.4 Metodologia de Testes

Os procedimentos analisados, baseiam-se, em alguns pontos da implementação, em variáveis aleatórias. Por este motivo, foram realizados 10 testes para cada instância, com os parâmetros fixados, somente variando a semente aleatória. O resultado de cada teste foi retornado em um formato padronizado, aceito pelo software *Sistema de Apoio ao Planejamento de Viagens* ou *SAPV*, desenvolvido por outros alunos que trataram o mesmo problema com outros procedimentos. Neste software pode-se retirar algumas informações dos resultados, como o atendimento da demanda, utilização de veículos

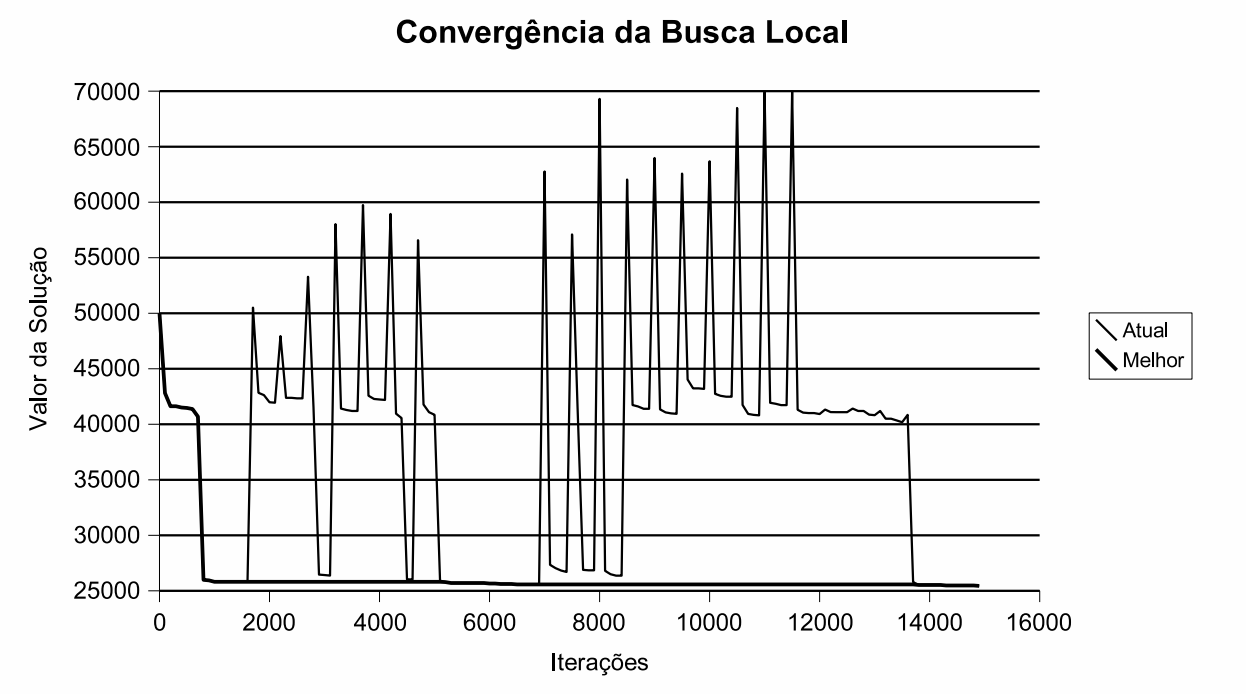

Figura 7: Convergência da Busca Local

e motoristas, número de horas extras e ociosas, dentre outras mais.

Para realizar as comparações, com os outros procedimentos, escolheu-se a melhor solução dentre as 10 disponíveis e focalizou-se principalmente o atendimento às restrições e a otimização dos recursos.

13.5 Descrição das Instâncias utilizadas

Estavam disponíveis para a realização de testes seis instâncias reais da região Metropolitada de São Paulo. Em geral, são linhas do tipo bairro-centro, onde existem grandes demandas nos bairros no ínicio do dia, e grandes demandas no centro no fim do dia. As linhas comumente recebem uma numeração, sendo as linhas tratadas as de identificadores 2280, 376, 702, 476, 5758 e 2210.

Nenhuma das linhas consideradas são de caráter circular, possuindo sempre dois PCs. Somente poderemos comparar os resultados obtidos com soluções feitas manualmente nas instâncias 376 e 702. As outras características das instâncias estão demonstradas na tabela 3. A legenda da tabela está descrita a seguir:

- **JL**: duração da jornada regular de um motorista;
- **HE**: número máximo de horas extras de um motorista;
- **JM**: duração da jornada mínima de um motorista;

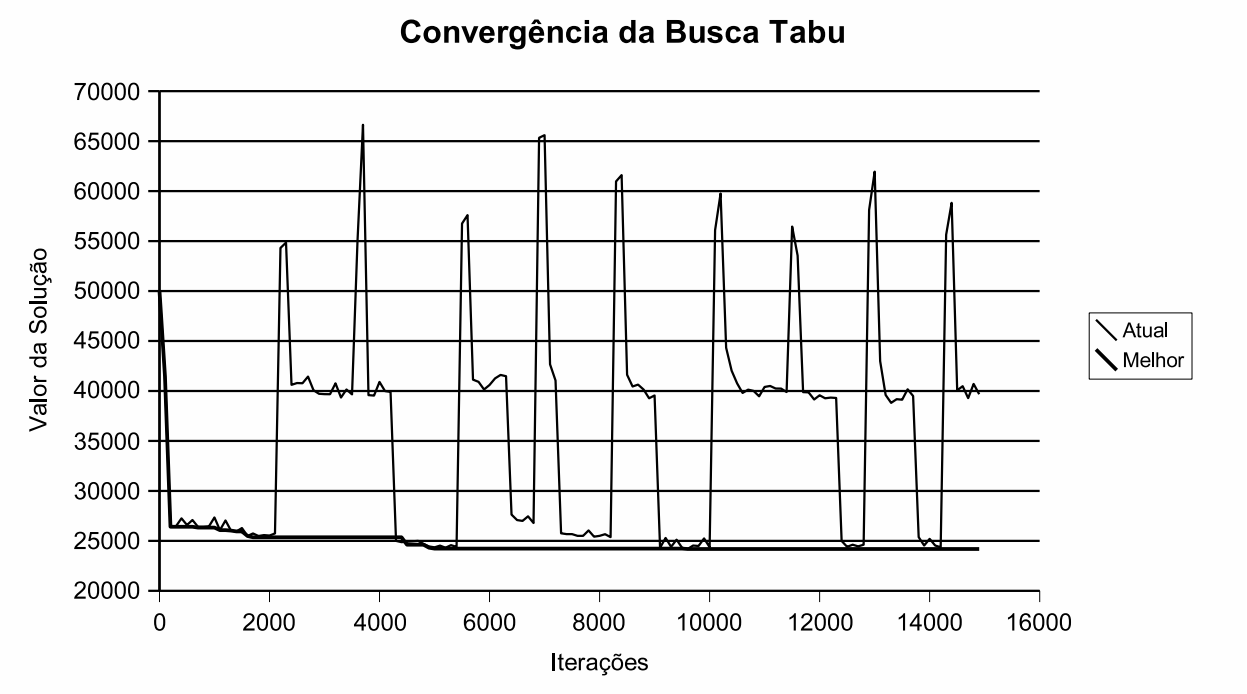

Figura 8: Convergência da Busca Tabu

- **NID**: menor instante em relação ao início da jornada em que se pode iniciar o descanso de um motorista;
- **MID**: maior instante em relação ao início da jornada em que se pode iniciar o descanso de um motorista;
- **DD**: duração do descanso;
- **SD**: S, se a empresa aceita que não haja descanso, e N caso contrário;
- **DR**: duração da rendição do motorista;
- **NV**: número de veículos disponíveis para o escalonamento.

Mesmo sendo restrição, o número de veículos disponíveis nem sempre é suficiente para o atendimento completo da demanda. No caso dos algoritmos implementados, e dos trabalhos realizados em [Vaz03] e [Cir04], a demanda é considerada uma restrição inviolável, sendo, deste modo, prioritária sobre o número de veículos disponíveis. Assim, algumas instâncias usaram mais veículos do que o permitido.

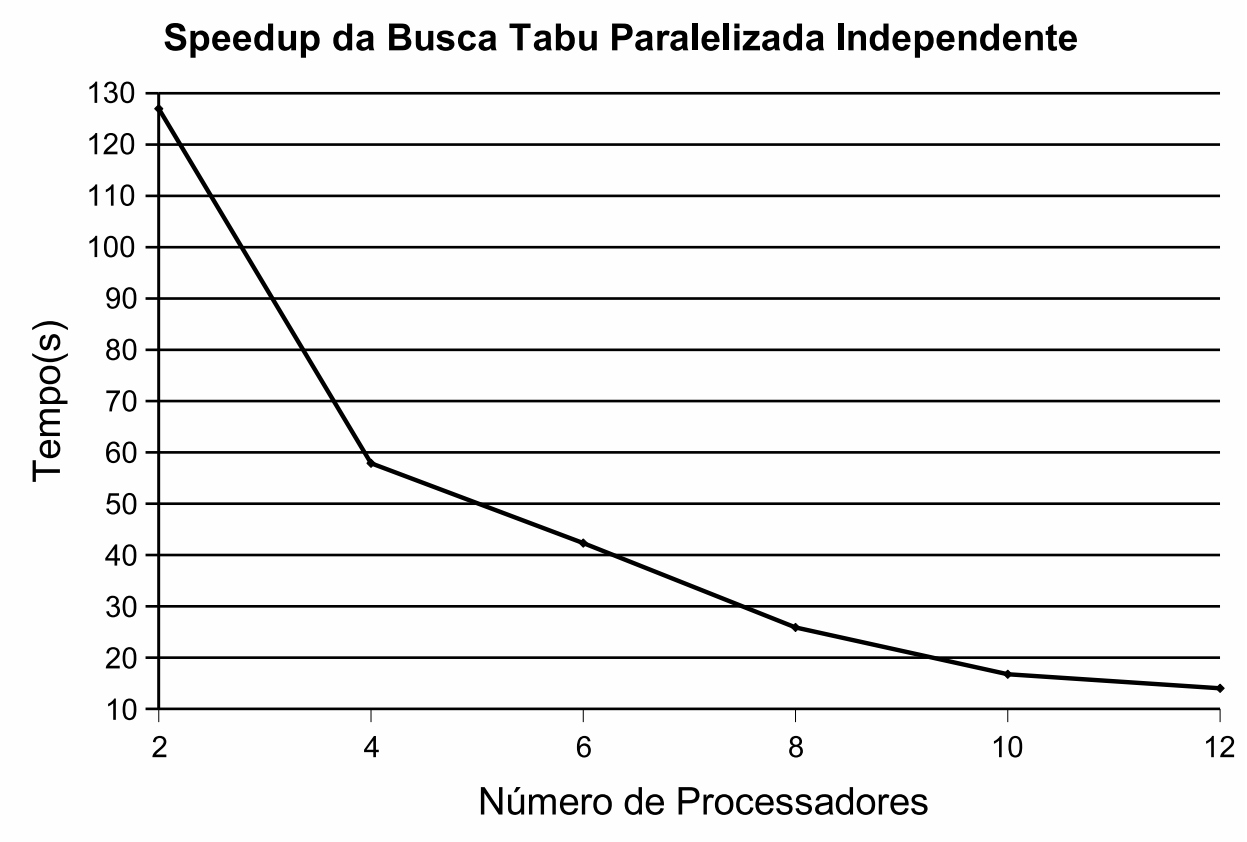

Figura 9: Speedup da Busca Tabu Paralelizada Independente

13.6 Resultados

Será apresentada a seguir uma seqüencia de tabelas com os melhores resultados obtidos para cada instância sobre diferentes aspectos. Em cada coluna temos os resultados de cada trabalho, sendo eles: [Rod01], [Vaz03], [Cir04], Busca Cega(BC), Busca Local(BL), Busca Tabu(BT) e Busca Tabu Paralelizada (BTP). Será sinalizado com um * quando um dos procedimentos conseguir o melhor resultado sobre o aspecto analisado em uma linha.

Número de Veículos								
Inst.	**[Rod01]**[3]	**[Vaz03]**	**Manual**	**[Cir04]**	**BC**	**BL**	**BT**	**BTP**
2280	31	19	-	*17	23	18	*17	*17
376[4]	20	13	19	12	8	7	11	*10
702[4]	35	23	32	*20	24	21	21	*20
476	12	11	-	10	10	10	10	*9
5758	10	8	-	8	8	8	8	*7
2210	33	26	-	26	29	23	*21	*21

Instância	JL	HE	JM	NID	MID	DD	SD	DR	NV
2280	07:20	02:00	05:00	03:00	05:00	00:30	S	00:20	25
376	07:45	01:50	05:00	03:00	05:00	00:35	S	00:20	15
702	07:10	00:50	05:00	03:00	05:00	00:30	S	00:20	26
476	07:20	01:00	05:00	03:00	05:00	00:30	S	00:20	7
5758	07:20	01:40	05:00	03:00	05:00	00:30	S	00:20	7
2210	07:20	02:00	05:00	03:00	05:00	00:30	S	00:20	40

Tabela 3: Principais diferenças operacionais entre instâncias

Número de Motoristas								
Inst.	**[Rod01]**[3]	**[Vaz03]**	**Manual**	**[Cir04]**	**BC**	**BL**	**BT**	**BTP**
2280	42	33	-	*31	38	35	32	*31
376[4]	28	24	30	21	16	14	20	*18
702[4]	51	40	52	36	44	42	40	*39
476	22	21	-	19	19	19	*18	*18
5758	*14	*14	-	*14	16	*14	*14	*14
2210	47	42	-	42	49	43	40	*36

Horas Regulares de Motoristas								
Inst.	**[Rod01]**[3]	**[Vaz03]**	**Manual**	**[Cir04]**	**BC**	**BL**	**BT**	**BTP**
2280	277:05	220:52	-	221:17	241:37	224:41	209:34	*205:57
376[4]	188:21	153:59	226:36	160:58	101:24	97:53	123:30	*118:40
702[4]	320:49	265:28	358:14	246:16	263:21	246:10	*239:29	239:37
476	145:07	142:14	-	137:15	111:55	117:23	*109:25	115:26
5758	98:33	101:51	-	109:41	93:29	94:27	93:23	*89:25
2210	301:47	281:18	-	293:31	297:18	277:21	258:36	*251:25

Horas Extras de Motoristas								
Inst.	**[Rod01]**[3]	**[Vaz03]**	**Manual**	**[Cir04]**	**BC**	**BL**	**BT**	**BTP**
2280	15:48	38:34	-	*09:21	21:30	15:18	19:08	15:55
376[4]	18:01	*02:58	11:44	07:59	06:45	07:57	06:10	05:36
702[4]	02:15	03:26	34:25	06:44	03:09	02:17	*00:57	03:36
476	08:05	03:49	-	02:05	00:54	00:22	*00:30	01:23
5758	13:13	12:10	-	11:25	01:35	01:42	*00:40	03:27
2210	21:42	35:34	-	*14:33	15:00	15:17	18:50	19:38

[3]A marcação com * para a melhor solução, não se aplica aos resultados de [Rod01] quando não há o atendimento completo da demanda.

[4]Nessas instâncias a BL e BC não atenderam a demanda por completo. Por isto, elas não receberam o *, mesmo quando possuem menor número de recursos na tabela.

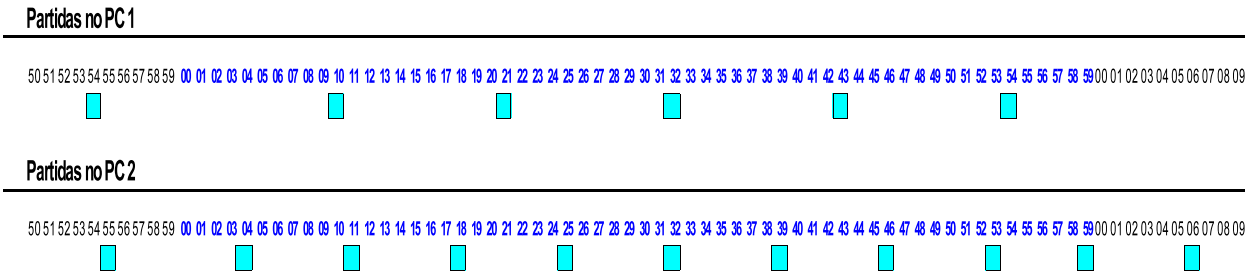

Figura 10: Exemplo de uma distribuição boa de horários de partida em uma faixa horária

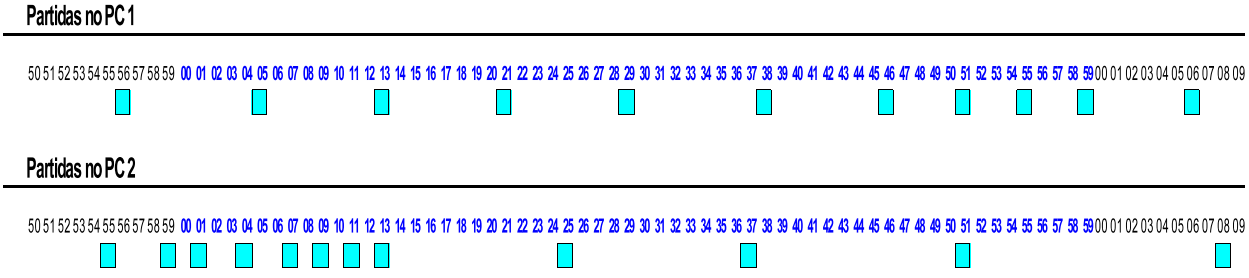

Figura 11: Exemplo de uma distribuição regular de horários de partida em uma faixa horária

Horas Ociosas de Motoristas								
Inst.	**[Rod01]**[3]	**[Vaz03]**	**Manual**	**[Cir04]**	**BC**	**BL**	**BT**	**BTP**
2280	26:18	21:49	-	*16:15	55:02	35:26	30:59	26:29
376[4]	22:57	27:52	36:20	13:52	10:09	10:35	16:30	*12:36
702[4]	13:54	23:19	23:54	*10:30	43:05	26:57	20:16	25:23
476	35:20	32:30	-	16:57	11:56	17:52	*09:16	15:02
5758	17:35	25:16	-	19:06	14:40	15:24	14:32	*14:01
2210	26:30	50:49	-	*17:43	54:29	42:52	29:20	25:15

Para obter esses resultados, os procedimentos executaram em média durante 300 segundos cada um, sendo um tempo pequeno e completamente viável para o problema em questão.

13.6.1 Distribuição de Horários de Partida

Como apresentado na seção 12, foram implementadas uma Busca Local e uma heurística para a distribuição dos horários de partida. Para verificar a eficácia desses procedimentos, realizou-se testes em todas as instâncias fornecidas, e sobre os resultados foi feita uma classificação para a distribuição dos horários.

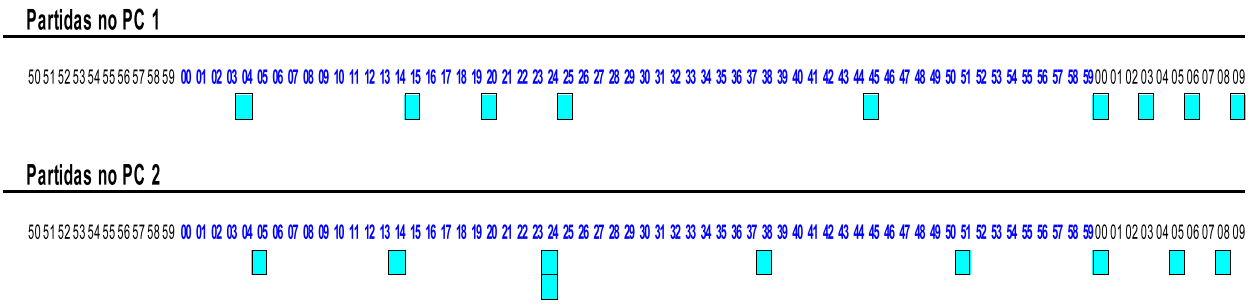

Figura 12: Exemplo de uma distribuição ruim de horários de partida em uma faixa horária

Dada uma faixa horária de uma solução, três níveis de qualidade foram definidos para a distribuição:

- "Boa": Neste caso, os horários estão bem distribuídos, sem concentrações de viagens em um pequeno período ou sobreposições de viagens num mesmo horário. Veja a figura 10.
- "Regular": Na faixa horária em questão, as partidas de somente um dos PCs estão concentradas ou sobrepostas, como na figura 11.
- "Ruim"': As partidas da faixa horária, saindo dos dois PCs estão concentradas ou sobrepostas. Veja a figura 12.

Assim sendo, foi contabilizado para cada instância o número de faixas horárias com cada um desses níveis. Esses dados são mostrados na tabela abaixo. Não foram contadas as faixas horárias com nenhuma partida em ambos os PCs.

Qualidade na Distribuição dos Horários de Partida						
	Busca Local			**Heurística**		
Inst.	***Ruins***	***Regulares***	***Boas***	***Ruins***	***Regulares***	***Boas***
2280	15	6	1	2	6	14
376	12	4	2	0	12	6
702	12	8	1	1	7	13
476	16	5	0	2	7	12
5758	10	9	1	2	9	9
2210	11	9	1	1	5	15
$\%_{\text{média}}$	61,8%	33,4%	4,9%	6,5%	37,4%	56,1%

Olhando a tabela, fica claro que a Busca Local não obteve bons resultados, sendo este o motivo da implementação da heurística que complementou os resultados da Busca Local. O fator de limitação da Busca Local é que ela se restringe a trocar jornadas por outras da enumeração, estando

dessa forma, a mercê da diversidade de jornadas e também, da manutenção da demanda atendida. Sabendo que as combinações de horários de partidas das viagens para cada jornada são muitas, não se pode esperar que a enumeração contenha todas essas possibilidades. Assim, a heurística, ao realizar alterações nos horários das viagens, possui mais liberdade e conseguiu melhores resultados.

Não foram mostrados os resultados obtidos sem o uso da Busca Local e da heurística pois, em geral, as distribuições ficaram ruins. Além disso, as distribuições de [Vaz03], que também foram omitidas, foram todas boas, sendo este o provável motivo da diferença de otimizações quando se comparam os custos relativos a veículos, motoristas e horas extras.

13.6.2 Análise dos Resultados

A Busca Tabu Monoprocessada e Paralelizada conseguiram obter resultados muito bons para as instâncias disponíveis. A principal característica encontrada nas soluções obtidas foi a otimização do número de veículos e motoristas necessários para a linha. Este é um resultado muito importante, dados os custos envolvidos na alocação desses recursos.

A linha que ilustra bem este cenário é a 2210. Nesta instância, foi encontrada uma solução com 21 veículos, uma grande otimização se comparada aos 26 veículos de [Cir04] e de [Vaz03]. Ainda assim, nesta instância, foi possível reduzir o número de motoristas para 36 ao custo de algumas horas extras a mais que os outros procedimentos.

O paralelismo se mostrou efetivo na obtenção de soluções de qualidade, atingindo patamares que a versão monoprocessada não conseguiu atingir, mesmo com um alto número de iterações. Dos componentes do modelo, a interação entre o componente de "Path Relinking" com a *oscilação distribuída* foi determinante para esses resultados, ao fornecer uma maior diversificação à busca.

Sobre a distribuição de horários, a heurística desenvolvida conseguiu evoluir muito os resultados, sendo que em todas instâncias deixou a maioria das faixas horárias com uma boa distribuição.

14 Conclusões e Perspectivas

Na primeira fase do projeto, foi obtido sucesso na implementação da Busca Tabu. O Algoritmo conseguiu otimizar em muito a utilização de recursos e também tratou e retirou a maioria das restrições. Mesmo obtendo resultados interessantes com um algoritmo básico, não foram negligenciados os diversos componentes citados na literatura. Deste modo, foi implementada uma oscilação estratégica que conseguiu melhorar em muito as soluções obtidas pela Busca Tabu.

Ainda buscando melhorar mais a qualidade das soluções, a segunda fase focou-se no desenvolvimento da paralelização do algoritmo. O produto final disso foi uma Busca Tabu Paralelizada com múltiplas *threads*, cooperando assíncronamente através de um "Elite Pool" localizado num processador *master* e recombinando as soluções dele provindas, com um componente de "Path Relinking". Com essa versão, conseguiu-se melhorar praticamente todas as soluções obtidas, e toma-se como um dos grandes responsáveis desses resultados, o conceito de *oscilação estratégica distribuída*, também utilizado no modelo proposto.

Como trabalhos futuros, existem diversas possibilidades. Sendo um problema prático e com múltiplos pontos de otimização, uma abordagem multiobjetivo seria natural para o problema,

seguindo os mesmos passos de [LPP01]. Outro ponto que ainda pode receber mais atenção é a distribuição de horários, que se demonstrou muito conflitante com a redução de custos. Dessa forma, novas abordagens poderiam ser propostas, tentando tratar essa distribuição mais intensamente no procedimento. Por outro lado, o uso de uma enumeração de jornadas levantou a possibilidade de sua construção utilizando outros tipos de modelagens [PGSH96], em alternativa ao modelo heurístico proposto. Isso tudo, ainda, deveria aliado à uma busca por novas instâncias e realidades, para que se possa verificar e evoluir o algoritmo, e torná-lo uma ferramenta acessível às empresas interessadas.

Referências

[ACR98] A. Andreatta, S. Carvalho, and C. Ribeiro. An object-oriented framework for local search heuristics, 1998.

[BHX02] Maria J. Blesa, Lluis Hernandez, and Fatos Xhafa. Parallel skeletons for tabu search method based on search strategies and neighborhood partition. In *PPAM '01: Proceedings of the th International Conference on Parallel Processing and Applied Mathematics-Revised Papers*, pages 185–193, London, UK, 2002. Springer-Verlag.

[Cir04] André Augusto Ciré. Algoritmo genético paralelo para o problema de programação de viagens. Technical report, Universidade Estadual de Campinas, Iniciação Científica FAPESP, 2004.

[CTG97] T. G. Crainic, Toulouse, and M. Gendreau. Toward a taxonomy of parallel tabu search heuristics. *INFORMS Journal on Computing*, 9(1):61–72, 1997.

[GH02] Hermann Gehring and Jörg Homberger. Parallelization of a two-phase metaheuristic for routing problems with time windows. *J. Heuristics*, 8(3):251–276, 2002.

[GL97] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Boston, MA, 1997.

[GP02] Michel Gendrau and Jean-Yves Potvin. Metaheuristics in combinatorial optimization. Technical report, Centre de recherche sur les transports and Département d'informatique et de recherche opérationnelle, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Québec, 2002.

[GS01] L. Di Gaspero and A. Schaerf. Easylocal++: an object-oriented framework for the flexible design of local search algorithms and metaheuristics. *in Proceedings of the 4th Metaheuristics International Conference (MIC'2001)*, July 2001.

[HG04] S. C. Ho and M. Gendreau. Path relinking for the vehicle routing problem. Technical report, Department of Informatics, University of Bergen, 2004.

[HTW92] Alain HERTZ, Eric TAILARD, and Dominique de WERRA. A tutorial on tabu search. Technical report, EPFL, Département de Mathématiques, MA-Ecublens, CH-1015 Lausanne and Université de Montréal, Centre de Reserche sur les Transports, Montréal, Canada H3C 3J7, 1992.

[LAG94] Manuel LAGUNA. Guide to implementing tabu search. *Investigación Operativa*, 4(1):4–25, April 1994.

[LPP01] H.R. LOURENÇO, J.P. PAIXÃO, and R. PORTUGAL. Multiobjective metaheuristics for the bus-driver scheduling problem. *Transportation Science*, 35(3):331–343, 2001.

[PGSH96] C. Pavlopoulou, A. Gionis, P. Stamatopoulos, and C. Halatsis. Crew pairing optimization based on clp, 1996.

[PR95] S. Porto and C. Ribeiro. Parallel tabu search message-passing synchronous strategies for task scheduling under precedence constraints, 1995.

[Rod01] M.M. Rodrigues. Problema de planejamento de viagens no transporte coletivo. Master's thesis, Instituto de Computação, UNICAMP, 2001.

[THG97] E. Talbi, Z. Hafidi, and J. Geib. Parallel adaptive tabu search for large optimization problems. *MIC'97-2nd Metaheuristics International Conference*, Juillet 1997.

[Vaz03] G.J. Vaz. Uma abordagem alternativa para os escalonamentos de Ônibus e motoristas. Master's thesis, Instituto de Computação, UNICAMP, 2003.

[WG97] Anthony Wren and Nicolau D Fares Gualda. Integrated scheduling of buses and drivers. Technical report, Departamento de Engenharia de Transportes, Escola Politécnica, Universidade de São Paulo, Brasil, 1997.