PAULO HENRIQUE SIQUEIRA

APLICAÇÃO DO ALGORITMO DO *MATCHING* NO PROBLEMA DA CONSTRUÇÃO DE ESCALAS DE MOTORISTAS E COBRADORES DE ÔNIBUS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciências. Curso de Pós-Graduação em Métodos Numéricos em Engenharia – Programação Matemática, Setores de Tecnologia e de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Prof. Dr. Celso Carnieri
Co-orientadora: Prof.ª Dr.ª Maria Teresinha Arns Steiner

CURITIBA

1999

PAULO HENRIQUE SIQUEIRA

APLICAÇÃO DO ALGORITMO DO *MATCHING* NO PROBLEMA DA CONSTRUÇÃO DE ESCALAS DE MOTORISTAS E COBRADORES DE ÔNIBUS

Dissertação aprovada como requisito parcial para obtenção do grau de Mestre em Ciências, no Curso de Pós-Graduação em Métodos Numéricos em Engenharia – Programação Matemática da Universidade Federal do Paraná, pela Comissão formada pelos professores:

Orientador:

_____
Prof. Dr. Celso Carnieri
Departamento de Matemática, UFPR

_____
Prof.ª Dr.ª Maria Teresinha Arns Steiner
Departamento de Matemática, UFPR

_____
Prof. Dr.ª Mirian Buss Gonçalves
Departamento de Matemática, UFSC

Curitiba, 15 de dezembro de 1999.

À minha família

AGRADECIMENTOS

À minha família, em especial à minha mãe Claudete Maria Rossa Siqueira, pelo apoio e incentivo durante a realização deste curso.

Ao professor Celso Carnieri, pela orientação para a realização deste trabalho, e principalmente pelo companheirismo e amizade.

À professora Maria Teresinha Arns Steiner, pela co-orientação e pelas valiosas sugestões, principalmente na programação dos algoritmos.

Aos professores: Anselmo Chaves Neto, Beatriz Pierin Barros Silva, Celso Carnieri, Jin Jiahong, Maria Teresinha Arns Steiner, Neida Maria Patias Volpi e Rubens Robles Ortega Jr., pelos ensinamentos transmitidos.

Aos colegas, pela amizade formada durante o curso, em especial aos amigos: Ângela Olandoski Barboza, Elizabeth Cristina Adamowicz, Maria Eugênia de C. e S. Sampaio e Marlene Tambosi, que contribuíram para a realização deste trabalho.

Ao coordenador do curso, professor Waldir de Lima e Silva Jr., e aos funcionários do CESEC, pela disposição em sempre ajudar, quando necessário.

Às empresas RADsystem Desenvolvimento de Sistemas Ltda. e GPD Informática, pelo fornecimento dos dados que foram utilizados neste trabalho.

A todos os demais, que de alguma forma contribuíram para a realização deste trabalho.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

O objetivo deste trabalho é mostrar a aplicação do algoritmo do *matching* de peso máximo na elaboração de jornadas de trabalho para motoristas e cobradores de ônibus. Este problema deve ser resolvido levando-se em consideração o maior aproveitamento possível das tabelas de horários, com o objetivo de minimizar o número de funcionários, de horas extras e de horas ociosas. Desta forma, os custos das empresas são minimizados. O serviço de transporte urbano difere dos outros serviços, pois tem características próprias. Existe uma grande demanda em alguns horários, como no início da manhã e no final da tarde. Nos demais horários, existe demanda, embora seja menor. Além disso, as tabelas de horários são feitas de acordo com a demanda de usuários, o que implica em horários de início e término desencontrados, o que dificulta a elaboração das jornadas de trabalho. Outra dificuldade deste problema é a necessidade do cumprimento da legislação trabalhista e dos acordos com o Sindicato dos Trabalhadores, pois este conjunto de regras impede, muitas vezes, um melhor aproveitamento dos horários. As tabelas das escalas de horários dos ônibus são compostas de escalas com diversas durações, onde as maiores são divididas em escalas menores, de tal modo que o custo seja mínimo. Com estas tabelas divididas, as escalas de curta duração podem ser combinadas para a formação da jornada diária de trabalho de um funcionário. Esta combinação é feita com o algoritmo do *matching* de peso máximo, onde as escalas são representadas por vértices de um grafo, e o peso máximo é atribuído às combinações de escalas que não formam horas extras e nem horas ociosas. As demais combinações têm pesos proporcionais às suas durações. Deste modo, as jornadas para os dias úteis, sábados e domingos são construídas. De acordo com as regras mencionadas, uma jornada de final de semana pode ser designada para cada jornada de dia útil. O algoritmo do *matching* de peso máximo é utilizado novamente, atribuíndo-se um peso máximo às combinações de jornadas que não formam horas extras e nem horas ociosas. Assim, a jornada semanal de trabalho dos motoristas e dos cobradores de ônibus pode ser construída, representando um custo mínimo. A última fase deste problema consiste na designação das jornadas semanais de trabalho para cada motorista e cobrador. O algoritmo do *matching* é também utilizado nesta fase, onde o peso máximo é atribuído às jornadas que mais aproximam-se da escala anterior de cada funcionário. Este trabalho foi aplicado em três empresas de transporte coletivo da cidade de Curitiba, onde os algoritmos usados são todos heurísticos. Os resultados do algoritmo exato do *matching* são melhores do que os resultados dos algoritmos heurísticos, e seu tempo computacional não é tão elevado.

ABSTRACT

The purpose of this work is show the aplication of maximum weight matching algorithm in elaboration of workdays for bus drivers and conductors. This problem must be solved take into consideration the utmost utilization of schedules, in order to obtain the minimal number of officers, of overtimes and of idles hours. With this aplication, the costs of companies of public transport are minimized. The work of public transport differ of other works, because possess own characteristic. Exist a greath demand in a few timetable, like in dawn and in the end of afternoon. In other timetables, exist a demand, although be less. Moreover, the schedule are made in line with the demand of users, what involve in start and end of timetables with a failure of meeting, what difficult the construction of workdays. Other dificulty in this problem is the necessity of accomplishment of labourite legislation and agreements with the Workers Syndicate, because this set of rules impede, many times, a better utilization of timetables. The tables of scales of buses are composed of scales with many durations, where the biggers are divided in smallers scales, so that the cost be minimal. With these tables divided, the scales of small duration can be combinated to the formation of workday of an officer. This combination is made with the maximum weight matching algorithm, where the scales are representaded like nodes of an graph, and the maximum weight is assigned to the combinations of scales without overtime and idles hours. The others combinations have weights proportionals to his durations. From this form, the workdays of weekdays, saturdays and sundays are constructed. In accordance with the rules mentioned, one workday of weekend can be assigned to each workday of weekday. The maximum weight matching algorithm is used, where the maximum weight is assigned to the combinations of workdays without overtime and idles hours. With this aplication, the weekly scale of work for the bus drivers and conductors can be constructed with minimal cost. The final phase of this problem consist in assignment of weekly scales for each bus driver and conductor. The maximum weight matching algorithm is used in this phase too, where the maximum weight is assigned to the workdays more approach of previous scale of each officer. This work was applied in three companies of public transport of city of Curitiba, where the algorithms used are all heuristics. The results of exact *matching* algorithm are better of that the results of heuristcs algorithms, and your computacional time isn't much high.

CAPÍTULO I

1. INTRODUÇÃO

1.1. OBJETIVOS DO TRABALHO

Neste trabalho são propostas algumas aplicações da Pesquisa Operacional para solucionar o problema de programação de jornadas de trabalho semanais para motoristas e cobradores de ônibus. Além disso, são mostradas as relações que permitem a utilização do algoritmo do *matching* de peso máximo em todas as fases deste problema.

Os objetivos do planejamento das escalas de horários dos motoristas e cobradores de ônibus são:

a) minimizar o número de horas extras e de horas ociosas nas jornadas diárias de seus funcionários;

b) atender a demanda dos usuários de transporte coletivo com um número mínimo de funcionários.

1.2 IMPORTÂNCIA DO TRABALHO

Na cidade de Curitiba, o órgão responsável pelo planejamento do transporte coletivo é a URBS (Urbanização de Curitiba S.A.), e cada empresa filiada a este órgão faz o atendimento das linhas de ônibus em uma região da cidade.

Através de pesquisas da URBS com cada empresa de transporte coletivo, as tabelas de escalas dos ônibus são construídas de acordo com a demanda dos usuários. Estas tabelas são separadas, de acordo com as categorias dos ônibus, pois alguns motoristas não trabalham em alguns tipos de veículos. As categorias dos ônibus são divididas do seguinte modo:

- ônibus comuns, nos quais os motoristas e cobradores cumprem as jornadas de trabalho no próprio veículo;
- ônibus linha direta, onde os cobradores trabalham nas estações tubo e suas escalas são diferentes das escalas dos motoristas e
- ônibus bi-articulados, onde os cobradores também cumprem suas jornadas de trabalho nas estações tubo.

Cada tabela separada por categoria de ônibus possui sub-tabelas para dias úteis, sábados e domingos, pois possuem demandas distintas.

As empresas de transporte coletivo recebem as tabelas das jornadas de ônibus para cada linha, e são responsáveis pela construção das jornadas de trabalho para os motoristas e cobradores. Esta construção deve minimizar as quantidades de funcionários, de horas extras e de horas ociosas.

As tabelas de jornadas de trabalho para motoristas e cobradores de ônibus são modificadas constantemente pela URBS, de acordo com a demanda dos usuários. Estas modificações implicam em novas construções das jornadas de trabalho para os funcionários.

Algumas empresas de transporte coletivo da cidade de Curitiba fazem a construção das jornadas de trabalho para todos os funcionários manualmente. Esta maneira de construção é demorada e está sujeita a erros de cálculo. Outras empresas utilizam algoritmos heurísticos, baseados na construção manual das mesmas. Nos dois casos, a solução encontrada nem sempre é a ótima.

1.3 ESTRUTURA DO TRABALHO

O presente trabalho está dividido em cinco capítulos, incluindo esta introdução.

No capítulo II são apresentadas algumas referências bibliográficas de problemas sobre a construção de jornadas de trabalho para funcionários e de problemas que fazem a designação de funcionários para suas respectivas jornadas. Além disso, são mostradas as técnicas de Pesquisa Operacional utilizadas para cada tipo de problema.

No capítulo III é feita uma apresentação do algoritmo do *matching* de peso máximo através de sua formulação matemática. São mostradas também as relações que existem entre o algoritmo do *matching* com pesos e o método Húngaro, utilizado para resolver os problemas de designação.

O capítulo IV mostra a descrição do problema das escalas de motoristas e cobradores, as particularidades de cada fase deste problema, bem como a construção dos pesos para o algoritmo do *matching* em todas as fases. Os resultados obtidos em cada fase do problema são comparados com os resultados das empresas que utilizam um algoritmo heurístico, baseado nas técnicas utilizadas pelos funcionários que elaboravam estas escalas manualmente.

No capítulo V, é feita uma avaliação dos resultados obtidos, além de citar algumas sugestões para futuros trabalhos.

CAPÍTULO II

2. REVISÃO BIBLIOGRÁFICA

2.1. INTRODUÇÃO

O objetivo principal deste capítulo é mostrar algumas técnicas da Pesquisa Operacional que têm sido utilizadas recentemente, na construção de escalas de trabalho para funcionários.

Além de motoristas e cobradores de ônibus, algumas publicações mostram problemas semelhantes como a construção de escalas de trabalho para tripulação de trens e de aviões.

Alguns trabalhos mostram métodos heurísticos para resolver estes problemas, tais como relaxações lineares ou algoritmos heurísticos baseados em algoritmos exatos. Estas aplicações heurísticas são feitas em problemas que envolvem muitas restrições, pois os métodos exatos não demonstram eficiência computacional nestes casos.

Existe uma ampla bibliografia sobre construções de jornadas de trabalho para funcionários, com diversas técnicas distintas. Neste trabalho, são citadas somente algumas publicações sobre este assunto.

2.2. ESCALAS DE TRABALHO PARA MOTORISTAS E COBRADORES DE ÔNIBUS

2.2.1. O método *Haustus*

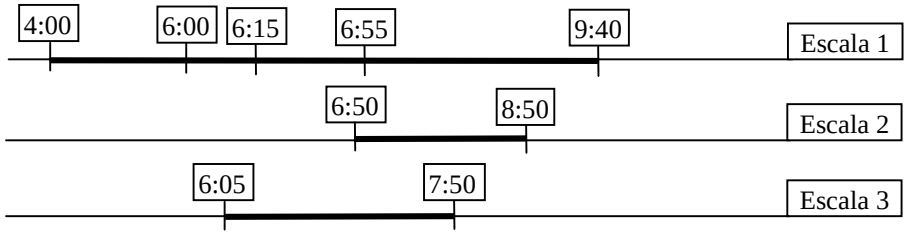
Existe um método de elaboração de escalas de trabalho de motoristas e cobradores de ônibus, conhecido na literatura como o método *Hastus*. Neste método, as escalas são construídas em três fases.

Na primeira fase, uma escala aproximada é construída usando-se um problema de Programação Linear, relaxando-se as restrições de integralidade e de factibilidade com o

objetivo de reduzir o tempo computacional. Estas restrições referem-se às possibilidades de formação e de combinação de escalas, máxima duração de uma parte de escala e sobre os locais de possíveis trocas de funcionários.

Por exemplo, na figura 2.1, a escala 1 deve ser dividida em duas escalas menores. Esta divisão deve ser feita, observando-se a possibilidade de combinar uma destas escalas resultantes com outra escala. Se a divisão for feita às 6:40 horas, a escala 2 não pode ser combinada com a primeira escala resultante desta divisão, pois não existe um intervalo entre as duas escalas. Além disso, estas divisões só devem ser feitas em pontos onde a troca de funcionários é possível, como em terminais de ônibus e pontos finais das linhas.

FIGURA 2.1 – COMBINAÇÕES DE ESCALAS.



Deste modo, obtém-se as primeiras escalas, que são denominadas “pedaços” de trabalho. O conjunto formado por estas escalas é denominado escala aproximada. Em uma etapa seguinte, as jornadas são divididas novamente, em partes semelhantes às primeiras escalas geradas, com o objetivo de melhorar a escala aproximada. Estas melhorias são feitas, reduzindo-se o número motoristas e a quantidade de horas extras que são formadas com as primeiras divisões.

A segunda fase consiste na combinação destas escalas divididas com a utilização do algoritmo do *matching* com atribuição de pesos às combinações factíveis de escalas de trabalho. A solução ótima da combinação é o *matching* de peso máximo. Esta é a idéia usada no presente trabalho para combinar as escalas, e sua descrição completa está nos capítulos seguintes.

Na última fase, as divisões das tabelas são reconsideradas e algumas melhorias são feitas modificando-se estas divisões visando, novamente, a redução do número de funcionários e a quantidade de horas extras. A descrição completa deste método, os diferentes modos de aplicação e de elaboração podem ser encontrados em publicações, tais como BLAIS et al. [2], LESSARD et al. [20] e ROUSSEAU et al. [24].

2.2.2. Os métodos heurísticos de divisões sucessivas

As heurísticas de divisões sucessivas constróem as escalas de trabalho em duas fases.

Na primeira fase, as divisões das escalas são feitas e combinadas, formando-se uma jornada diária de trabalho completa para o funcionário. Estas escalas combinadas devem formar dias de trabalho factíveis, com a mesma idéia utilizada no primeiro estágio do método *Haustus*.

Na segunda fase, as divisões das escalas são modificadas várias vezes, e o algoritmo heurístico procura formar jornadas de trabalho factíveis, melhorando as escalas de trabalho obtidas na primeira fase, com relação às quantidades de funcionários e de horas extras formadas. Algumas publicações que mostram estes métodos heurísticos e suas aplicações com mais detalhes são: MANINGTON et al.[21], WILHELM [27] e WREN et al. [29].

Na cidade de Curitiba, onde o problema que será descrito no capítulo IV foi implementado, estas divisões sucessivas são feitas somente para se obter as escalas diárias de trabalho (ou pedaços de trabalho). As combinações são feitas com um algoritmo heurístico, baseado nas técnicas utilizadas pelos funcionários que elaboravam as jornadas de trabalho manualmente. Com as jornadas diárias formadas, as divisões de escalas são feitas novamente, melhorando-se as escalas obtidas na primeira etapa do algoritmo.

2.2.3. O método de cobertura de conjuntos

Um método bastante utilizado para construir escalas de trabalho é o método de geração de colunas, usado para resolver problemas de Programação Linear que envolvem muitas variáveis.

No artigo de DESROCHERS et al. [10], o problema de cobertura de conjuntos é utilizado para elaborar as escalas de trabalho de motoristas e cobradores de ônibus. O objetivo deste método é cobrir todas as escalas com um custo mínimo. Em sua formulação, a função objetivo minimiza os custos dos dias de trabalho, e cada variável representa um dia de trabalho factível.

Neste artigo, o método de geração de colunas foi utilizado para resolver o problema de cobertura de conjuntos. Em uma primeira fase, os dias de trabalho factíveis são gerados e formam os nós de um grafo. Os dias de trabalho factíveis devem satisfazer as mesmas restrições do primeiro estágio do método *Haustus*. Esta geração de nós utiliza um algoritmo semelhante ao método *branch and bound* desenvolvido por RYAN e FOSER [25].

Um arco do grafo gerado representa um “pedaço” de trabalho, um intervalo, o início ou o final de um dia de trabalho. Os custos dos arcos deste grafo referem-se ao número de horas ociosas e ao custo das horas de trabalho. A descrição completa deste grafo pode ser encontrada em DESROCHERS [12].

Na segunda fase, o problema de encontrar o caminho com custo mínimo foi resolvido utilizando-se o algoritmo da Programação Dinâmica *backward* (DESROCHERS [11]). Estes caminhos formam as possíveis jornadas diárias de trabalho de motoristas e cobradores de ônibus, utilizando-se os nós gerados pelo método de geração de colunas.

Este trabalho foi implementado em uma cidade americana e outra britânica.

Estas cidades utilizavam programas similares aos métodos heurísticos apresentados nesta seção para construir as escalas de trabalho dos funcionários.

Neste artigo, são feitas comparações com o método *Haustus*, o método heurístico e o método de cobertura de conjuntos. Nas duas cidades, o método de cobertura de conjuntos

apresentou melhor desempenho computacional e obteve melhores resultados do que os dois anteriores.

2.3. ESCALAS DE TRABALHO PARA TRIPULAÇÃO FERROVIÁRIA

Semelhante ao problema das escalas de motoristas de ônibus, o problema das escalas de trabalho para a tripulação ferroviária consiste em construir as escalas de trabalho e designá-las aos funcionários de tal maneira que a tabela de demandas seja cumprida. Este problema está descrito em detalhes em CAPRARA et al. [3].

Inicialmente, toma-se uma tabela de serviços de trem, a ser cumprida diariamente em certos períodos de tempo. Alguns segmentos de jornadas são definidos para cada funcionário. Estes segmentos definem os tempos inicial e final de uma jornada e as estações de entrada e de saída para um funcionário.

Quando um funcionário termina sua jornada, o mesmo pode retornar à estação base (início da jornada), ou deslocar-se a uma outra estação para iniciar outra jornada. Este deslocamento não é considerado como uma jornada de trabalho, e o funcionário tem um intervalo de pelo menos um dia para fazer este deslocamento e iniciar sua próxima jornada de trabalho em uma estação.

O problema consiste em encontrar um conjunto de escalas, que cubra toda a tripulação e satisfaça a demanda, com um custo mínimo.

Para resolver este problema, considera-se um grafo direcionado, onde o arco (i, j) existe se e somente se o a sequência de trabalho (i, j) existir, ou seja, o funcionário j pode substituir o funcionário i em uma determinada estação.

Utiliza-se o modelo clássico de cobertura de vértices, com a relaxação Lagrangeana para modelos de Programação Linear Inteira (FISHER[15]).

Este trabalho foi implementado em uma companhia férrea italiana, e os resultados desta aproximação Lagrangeana foram melhores do que os resultados heurísticos, utilizados pela companhia.

2.4. ESCALAS DE TRABALHO PARA TRIPULAÇÃO AÉREA

Este problema é semelhante ao problema de construção de escalas para tripulação ferroviária, e está apresentado com mais detalhes em GAMACHE et al. [16].

A resolução deste problema é feita em duas fases. Na primeira fase, formam-se pares de segmentos de vôo em dois dias consecutivos, onde a tripulação viaja e retorna à cidade base. Alguns dias isolados, que não formam pares, são separados dos demais períodos.

O problema de tripulação em pares consiste em encontrar o conjunto de pares que cobre todos os segmentos de vôo, com custo mínimo. O custo de um par de vôo é dado pela duração total deste par.

Na segunda fase, toma-se a listagem da tripulação, construindo-se a escala mensal personalizada para cada funcionário, de acordo com sua atividade.

Neste artigo, o método de geração de colunas é utilizado para resolver este problema, gerando vários subproblemas *NP-Hard*.

Algumas relaxações lineares e aproximações na solução final foram feitas no método de geração de colunas, através do programa de otimização *CPLEX 2.0* (CPLEX [8]), com o objetivo de reduzir o tempo computacional.

Com as modificações mencionadas, o tempo computacional foi reduzido na ordem de mais de 1000.

Este procedimento de solução foi testado na companhia *Air France*, onde o problema pode ser dimensionado do seguinte modo: milhares de restrições, centenas de subproblemas e centenas de milhares de arcos.

Alguns testes com o programa *CADET*, mostram que a aplicação do método de geração de colunas é mais eficiente computacionalmente para o problema da companhia *Air France*, e os resultados são melhores do que os métodos heurísticos utilizados pela companhia.

2.5. DESIGNAÇÃO DE ESCALAS DE TRABALHO PARA MOTORISTAS DE ÔNIBUS

Uma das últimas fases da construção de jornadas de trabalho de motoristas e cobradores de ônibus é a designação de cada jornada semanal de trabalho para um funcionário.

Esta designação envolve um grande número de restrições, impostas com o objetivo de satisfazer os critérios estabelecidos pelas empresas de transporte coletivo. Por exemplo, alguns funcionários não podem ser escalados para alguns horários, como no início da manhã ou no final da noite por problemas de deslocamento da sua residência até o local de trabalho ou vice-versa.

Uma aproximação utilizada para a designação de motoristas de ônibus para as jornadas é denominada *Multi-level bottleneck* (ou "engarrafamento" em vários níveis). A descrição completa deste método pode ser encontrada em CARRARESI et al.[6].

Nesta aproximação, considera-se um grafo bipartido, onde um conjunto de vértices representa as jornadas, e o outro conjunto de vértices representa os funcionários.

Um peso $p(i, j)$ é atribuído para cada designação de uma escala i para um funcionário j, onde este peso representa o custo total desta designação. A duração e o horário de início e de término de cada escala devem ser considerados, pois os custos da empresa com o funcionário dependem destes fatores.

O objetivo é encontrar um peso total mínimo das designações de escalas para motoristas.

O algoritmo inicia com uma designação inicial. A partir desta designação, escolhe-se um vértice sem designação, criando-se uma cadeia alternante, cujos vértices são designados e não designados. Quando uma cadeia possui o peso total dos arcos sem designação menor do que o peso total dos arcos já designados, trocam-se as funções da designação, ou seja, os arcos que não tinham designação passam a ter, e os demais deixam de ter designação.

Este algoritmo é uma simplificação do algoritmo do *matching* de peso mínimo, usado somente para grafos bipartidos (CHRISTOFIDES[7]).

Os resultados encontrados demonstram que este algoritmo é mais eficiente do que o algoritmo desenvolvido por DERIGS e ZIMMERMANN [9], mais conhecido como o método DZ. O método DZ começa com uma solução inicial, em geral não factível, e a cada iteração, o algoritmo procura reduzir as infactibilidades.

CAPÍTULO III

3. O PROBLEMA DO *MATCHING* COM PESOS

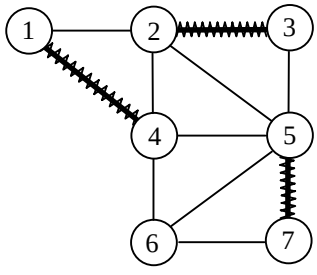
3.1. DESCRIÇÃO DO PROBLEMA

Sejam X um conjunto de vértices e A um conjunto de arcos associados aos elementos de X. Seja $G = (X, A)$ um grafo não direcionado, com pesos (ou custos) p_i atribuídos aos arcos $a_i \in A$.

Um *matching* (que significa emparelhar ou formar pares) é um conjunto de pares de vértices de um grafo com pesos não nulos, e tais que cada vértice deve formar, no máximo, um par.

Deste modo, um *matching* de um grafo $G = (X, A)$ é um subconjunto M do conjunto A de arcos de G, escolhidos de tal maneira que cada vértice é a extremidade de, no máximo, um arco de M, ou seja, não existem arcos adjacentes em M (CHRISTOFIDES [7]). A figura 3.1 mostra um exemplo de um conjunto *matching*, onde os arcos destacados formam tal conjunto.

FIGURA 3.1 – EXEMPLO DE UM CONJUNTO *MATCHING*.



O problema de encontrar o *matching* M com peso máximo consiste em maximizar a soma dos pesos dos arcos pertencentes a M. Deste modo, o peso total é dado por:

$$P_M = \sum_{a_j \in M} p_j \ .$$

Neste capítulo, são apresentadas as considerações para o problema do *matching* de peso máximo, pois a abordagem para o problema do *matching* de peso mínimo é análoga.

A formulação da programação matemática para o problema do *matching* de peso máximo utiliza variáveis binárias x_{ij}, ou seja, se um arco (i, j) pertence ao *matching*, $x_{ij} = 1$ e, caso contrário, $x_{ij} = 0$.

A formulação matemática não se mostra eficiente computacionalmente, pois envolve um grande número de restrições. A quantidade de restrições depende do número n de vértices do grafo, e pode ser estimada por $n + k$, onde k é da ordem n^2 (THACKER [26]). Por exemplo, para um grafo completo de 50 vértices, o número de restrições é aproximadamente igual a $5{,}6 \times 10^{14}$.

O problema do *matching* pode ser resolvido com o uso de um algoritmo próprio, que está descrito nas próximas seções deste capítulo. Este algoritmo tem uma convergência mais rápida do que o uso da formulação matemática do *matching*, embora faça aproximadamente n^3 cálculos (CHRISTOFIDES [6]).

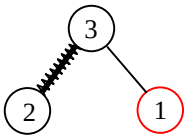
Como o problema das escalas de motoristas e cobradores de ônibus necessita de um algoritmo cuja convergência seja rápida, a formulação matemática não é a mais eficaz. Por isso, optou-se pelo uso do algoritmo do *matching* de peso máximo para resolver o problema em questão.

3.2. O USO DAS ÁRVORES ALTERNANTES NO ALGORITMO DO *MATCHING* COM PESOS

Esta é a parte principal do algoritmo do *matching* de peso máximo, pois permite que o peso final do *matching* seja maximizado através das construções das principais cadeias do grafo.

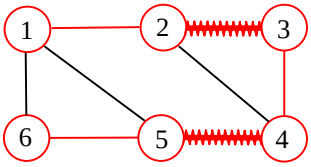
Tomando-se um *matching* M de um grafo $G = (X, A)$, um vértice $i \in X$ que não é a extremidade de um dos arcos do conjunto M é denominado vértice exposto com relação ao *matching* M. Na figura 3.2, o vértice 1 é exposto.

FIGURA 3.2 – EXEMPLO DE UM GRAFO COM VÉRTICE EXPOSTO.



Uma cadeia alternante, relativa a um conjunto *matching M*, é uma cadeia elementar cujos arcos são, alternadamente, pertencentes e não pertencentes (ou não pertencentes e pertencentes) ao conjunto *M*. Na figura 3.3, a cadeia $C = \{1, 2, 3, 4, 5, 6\}$, nesta ordem, é um exemplo de cadeia alternante, onde os arcos representados em vermelho formam a cadeia *C*.

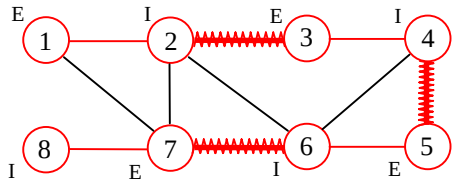
FIGURA 3.3 – EXEMPLO DE UMA CADEIA ALTERNANTE.



Uma árvore alternante relativa a um conjunto *matching M* é uma árvore *T* na qual um vértice de *T* exposto é denominado raiz de *T*, e todas as cadeias partindo do vértice raiz são cadeias alternantes.

Na construção de uma cadeia ou árvore alternante, inicia-se do vértice raiz da árvore e rotula-se o mesmo como externo (E). Os demais vértices são rotulados alternadamente como internos (I) e externos (E) com relação ao vértice raiz. Esta rotulação serve para atualizar os valores das variáveis duais de cada vértice do grafo a partir do vértice raiz. A figura 3.4 mostra um exemplo de uma cadeia alternante com as rotulações mencionadas. Neste exemplo, os vértices expostos são 1 e 8.

FIGURA 3.4 – EXEMPLO DE UMA ÁRVORE ALTERNANTE.



Uma cadeia aumentante é uma cadeia alternante na qual os vértices inicial e final são expostos. Se a soma dos pesos dos arcos do *matching* for menor do que a soma dos pesos dos arcos da cadeia que não pertencem ao *matching*, tem-se uma cadeia aumentante forte.

Na figura 3.5a, C_M = {1, 2, 3, 4, 5, 6} representa um exemplo de uma cadeia aumentante forte. Invertendo-se as funções do *matching* dos arcos nesta cadeia, ter-se-ia um *matching* resultante com peso maior do que o *matching* original. Na figura 3.5b, as funções do *matching* da cadeia aumentante da figura 3.5a são trocadas, obtendo-se um peso resultante maior.

FIGURA 3.5 – EXEMPLO DE UMA CADEIA AUMENTANTE FORTE. FIGURA 3.5a – CADEIA AUMENTANTE FORTE. FIGURA 3.5b – CADEIA AUMENTANTE DA FIGURA 3.5a, COM AS FUNÇÕES DO *MATCHING* INVERTIDAS.

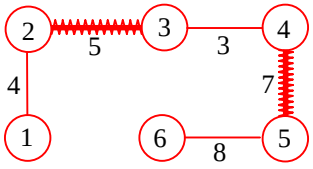


FIGURA 3.5a

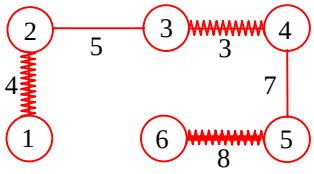


FIGURA 3.5b

Se a soma dos pesos dos arcos do *matching* for igual à soma dos pesos dos arcos que não pertencem ao *matching*, tem-se a cadeia aumentante neutra. Na figura 3.6, C_N = {1, 2, 3, 4} é um exemplo deste tipo de cadeia aumentante. Neste caso, a troca é feita se e somente se o objetivo for de maximizar o peso total dos arcos e o número de vértices que pertencem ao conjunto *matching*.

FIGURA 3.6 – EXEMPLO DE UMA CADEIA AUMENTANTE NEUTRA.

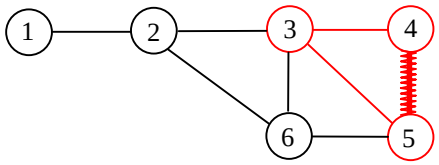
No caso em que a soma dos pesos dos arcos do *matching* for maior do que a soma dos pesos dos arcos que não pertencem ao *matching*, tem-se a cadeia aumentante fraca. Neste caso, a troca dos arcos não deve ser feita, pois o valor do peso total dos arcos do conjunto *matching* fica menor. Na figura 3.7, C_F = {1, 2, 3, 4 } é um exemplo de cadeia aumentante fraca.

FIGURA 3.7 – EXEMPLO DE UMA CADEIA AUMENTANTE FRACA.

Um ciclo ímpar, referente a um *matching M*, é uma cadeia alternante na qual os vértices inicial e final são idênticos e sua cardinalidade é ímpar. Quando a cardinalidade for par, o ciclo é denominado de ciclo regular.

A figura 3.8 mostra um exemplo de um ciclo ímpar, no caso C_I = {3, 4, 5}, nesta ordem. Nesta mesma figura, C = {1, 2} é uma cadeia aumentante, pois 1 e 2 são vértices expostos.

FIGURA 3.8 – EXEMPLO DE UM CICLO ÍMPAR.



No algoritmo do *matching* de peso máximo, os ciclos ímpares são retraídos em vértices artificiais, pois a escolha dos arcos do conjunto *matching* de um ciclo ímpar depende dos pesos dos arcos com extremidades nos vértices deste ciclo. Considerando-se o grafo representado na figura 3.9a, tem-se o ciclo ímpar C_I = {3, 4, 5} onde o arco (3, 4) é o que tem menor peso e pertence ao conjunto *matching*. Na figura 3.9b, o grafo representado contém o ciclo ímpar C_I = {3, 4, 5} que não possui arcos no conjunto *matching*. Com estes dois

exemplos, pode notar-se que não existe um critério para determinar quais arcos de um ciclo ímpar devem pertencer ao conjunto *matching*.

FIGURA 3.9 – EXEMPLOS DE CICLOS ÍMPARES.

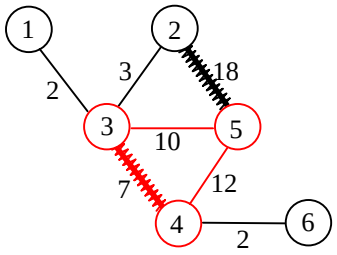


FIGURA 3.9a

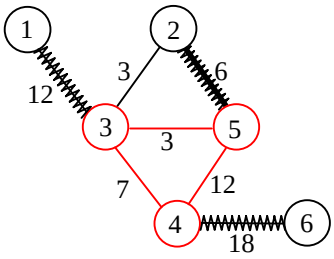
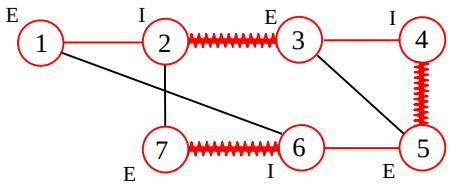


FIGURA 3.9b

Uma árvore húngara de um grafo, relativa a um *matching M*, é uma árvore alternante na qual o vértice final não é exposto, mas rotulado como interno ou externo. Deste modo, pode-se dizer que uma árvore húngara é uma cadeia alternante com um único vértice exposto, excluindo-se os casos de ciclos.

Um vértice isolado é uma árvore húngara, pois o vértice final é exposto, mas coincidente com o vértice inicial. Quando encontra-se uma árvore húngara, o algoritmo do *matching* procura algum vértice exposto para "completar" uma cadeia aumentante. Na figura 3.10 encontra-se um exemplo de árvore húngara, que é formada pelos vértices 1, 2, 3, 4, 5, 6 e 7, nesta ordem.

FIGURA 3.10 – EXEMPLO DE UMA ÁRVORE HÚNGARA.



Quando uma cadeia ou árvore alternante for gerada pelo algoritmo do *matching*, a mesma inicia-se de um vértice raiz, exposto no grafo *G*, e deve ser uma cadeia aumentante ou um ciclo ímpar ou uma árvore húngara.

TEOREMA 3.1 – Um subconjunto M de um conjunto de arcos do grafo G é um *matching* de peso máximo se, e somente se, M não possuir cadeias aumentantes fortes.

DEMONSTRAÇÃO:

a) NECESSIDADE: Por redução ao absurdo, admite-se que M possui uma cadeia aumentante forte. Logo, M não tem peso máximo, pois o *matching* M', obtido pela inversão das funções dos arcos da cadeia aumentante C, tem peso maior do que o *matching* M.

b) SUFICIÊNCIA: Pela contra-positiva, supõe-se que exista um conjunto *matching* M' com peso maior do que M. Seja G' o conjunto de todos os arcos que pertencem a um destes dois conjuntos M e M', ou seja, $G' = M \cup M'$. Tem-se que cada vértice de G' é a extremidade de, no máximo, dois arcos de G'. Caso contrário, ter-se-ia dois arcos de um mesmo conjunto *matching* (M ou M') com um vértice em comum, o que é impossível. Logo, o conjunto G' contém somente cadeias e ciclos regulares. Como o conjunto M' tem peso maior do que M, em uma das cadeias de G' os arcos de M' devem ter um peso maior do que os arcos de M. Isto implica na existência de uma cadeia aumentante forte em M.

C.Q.D.

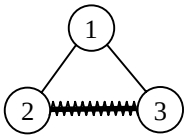
A seguir, o problema do *matching* de peso máximo será formulado como um problema de programação inteira, e, a partir desta formulação seu algoritmo será descrito.

3.3. FORMULAÇÃO DA PROGRAMAÇÃO MATEMÁTICA PARA O PROBLEMA DO *MATCHING*

Sejam $V = \{V_1, V_2, ..., V_z\}$ o conjunto de todos os subconjuntos de vértices com cardinalidade ímpar do grafo $G = (X, A)$, T_m o conjunto de todos os arcos com as extremidades no conjunto de vértices V_m e $T = \{T_1, T_2, ..., T_z\}$.

O número de vértices de V_m é dado por $2n_m + 1$, onde $n_m \in$?. Um conjunto *matching* de um conjunto T_m não pode conter mais do que n_m membros de T_m. Para exemplificar, seja V_1 um subconjunto de G com 3 vértices, ilustrado na figura 3.11. Neste caso, $n_m = 1$, e o número máximo de arcos de um conjunto *matching* é igual a 1. Esta é uma das restrições do problema do *matching* de peso máximo. Com esta restrição abrangendo todos os subconjuntos com cardinalidade ímpar de um grafo, não permite-se a formação de arcos adjacentes pertencentes ao conjunto *matching*.

FIGURA 3.11 – EXEMPLO DE UM CONJUNTO COM CARDINALIDADE ÍMPAR.



Deste modo, o problema do *matching* de peso máximo pode ser formulado como um problema de programação inteira, onde as variáveis estruturais são x_{ij}. Assim, se o arco (i, j) não pertencer a M, $x_{ij} = 0$ e, caso contrário, $x_{ij} = 1$ (MINIEKA [22]).

A formulação do problema primal (PP) para o *matching* de peso máximo é dada por:

(PP)

Maximizar $\sum_{(i,j)} p_{ij} x_{ij}$ (3.3.1)

Sujeito a $\sum_{j \in X} (x_{ij} + x_{ji}) \leq 1, \ \forall i \in X$ (3.3.2)

$\sum_{(i,j) \in T_m} x_{ij} \leq n_m$, onde $m = 1, 2, ..., z$ (3.3.3)

$x_{ij} = 0$ ou 1 , $\forall (i, j)$. (3.3.4)

O conjunto de restrições (3.3.2) garante que cada arco (i, j) será considerado pertencente ao conjunto *matching*, no máximo, uma vez. Além disso, como o grafo é considerado não direcionado, se x_{ij} for igual a 1, então x_{ji} deve ser 1.

O conjunto (3.3.3) refere-se aos subconjuntos de cardinalidade ímpar de *G*, e para cada conjunto T_m de vértices em V_m, o número de arcos em *M* deve ser menor que n_m. Estas restrições não permitem a formação de arcos adjacentes pertencentes ao conjunto *matching*. O número *z* representa a quantidade total de subconjuntos com cardinalidade ímpar do grafo *G*.

O algoritmo do *matching* de peso máximo foi construído baseando-se nas condições complementares de folga dos problemas primal e dual. Segue abaixo a formulação do problema dual correspondente ao primal (PP) apresentado.

(PD)

Minimizar $\sum_{i \in X} y_i + \sum_{m=1}^{z} n_m z_m$ (3.3.5)

Sujeito a $y_i + y_j + \sum_{m:(i,j) \in T_m} z_m \geq p_{ij}, \quad \forall (i, j)$ (3.3.6)

$y_i \geq 0, \quad \forall i \in X$ (3.3.7)

$z_m \geq 0, \quad$ onde $m = 1, 2, ..., z.$ (3.3.8)

Usando-se as condições complementares de folga para o par primal-dual (ZIONTS [30]), obtém-se os seguintes resultados:

a) se a variável primal $x_{ij} > 0$, ou seja, se o arco (i, j) pertencer ao conjunto *M* então

$$y_i + y_j + \sum_{m:(i,j) \in T_m} z_m = p_{ij}, \quad \forall (i, j); \tag{3.3.9}$$

b) se a variável dual $y_i > 0$, então

$$\sum_{j \in X} (x_{ij} + x_{ji}) = 1, \ \forall i \in X, \tag{3.3.10}$$

ou seja, x_{ij} ou x_{ji} será igual a 1, o que garante que o arco (i, j) pertence ao conjunto dos arcos do *matching*;

c) se a variável dual $z_m > 0$, então

$$\sum_{(i,j) \in T_m} x_{ij} = n_m, \text{ onde } m = 1, 2, ..., z, \tag{3.3.11}$$

ou seja, existem n_m arcos do conjunto V_m que pertencem ao conjunto M, sendo este o maior número possível de arcos em V_m que podem pertencer ao *matching*.

O problema do *matching* poderia ser resolvido através de sua formulação matemática, mas sempre tem-se que obter todos os subconjuntos de vértices de G com cardinalidade ímpar. Em grafos com um número razoável de vértices, esta obtenção torna-se difícil, pelo grande número de combinações de vértices, aumentando o número de restrições do conjunto (3.3.3) e, consequentemente, o tempo computacional. Uma outra maneira de solucionar o problema do *matching* é a busca exaustiva, onde todas as possibilidades de escolhas de soluções seriam testadas. Neste caso, o número de possibilidades é igual a n!.

Na próxima seção, o algoritmo do *matching* de peso máximo será apresentado e a sua aplicação resolve o problema do grande número de restrições que a formulação matemática exige. A combinação das igualdades e desigualdades de (3.3.1) a (3.3.11) permite a construção de tal algoritmo.

3.4. O ALGORITMO DO *MATCHING* DE PESO MÁXIMO

Nesta seção é apresentado o algoritmo para encontrar o *matching* de peso máximo, devido a EDMONDS e JOHNSON [13]. O passo principal deste algoritmo é a construção de árvores ou cadeias alternantes. No Anexo 1, encontra-se um exemplo numérico de aplicação deste algoritmo.

3.4.1 Descrição do algoritmo

PASSO 1 – *Inicialização:*

Sejam M_0 o conjunto *matching* inicial, que não contém arcos, e as variáveis duais $z_m = 0$, onde $m = 1, 2, ..., z$. Escolha qualquer valor inicial para as variáveis duais y_i, com $i \in X$, tais que $y_i + y_j \geq p_{ij}$, para todos os arcos (i, j) (condição 3.3.6).

Fazendo-se cada y_i igual à metade do peso máximo, tem-se $y_i + y_j \geq p_{ij}$ para todos os arcos (i, j). Seja $k = 0$. Denomine o grafo original por $G_k = (X_k, E_k)$ e faça $k = k + 1$.

PASSO 2 – *Verificação do Vértice Exposto:*

Selecione qualquer vértice v, não artificial, exposto no grafo G_k, com $y_v > 0$. Se não existir tal vértice, vá para o passo 6.

Caso contrário, seja E^* o conjunto de todos os arcos (i, j) em G_k tais que

$$y_i + y_j + \sum_{m:(i,j)\in T_m} z_m = p_{ij}, \quad \forall (i, j)$$

Neste momento, o algoritmo procura os arcos (i, j) que satisfazem a igualdade (3.3.9) da formulação matemática do problema dual do *matching* (PD). Se $x_{ij} > 0$, então a condição (3.3.9) é verdadeira, isto é, se o arco (i, j) pertence ao *matching*, então satisfaz a igualdade (3.3.9).

A partir do vértice v (rotulado como externo, E), crie uma árvore alternante gerada por v usando somente arcos em E^*, rotulando-se os demais vértices alternadamente como internos (I) e externos (E). Esta rotulação serve para atualizar os valores das variáveis duais y_i quando encontra-se uma árvore húngara.

Se uma cadeia aumentante for encontrada, vá para o passo 3.

Se um ciclo ímpar for encontrado, vá para o passo 4.

Se uma árvore húngara for encontrada, vá para o passo 5.

PASSO 3 – *Cadeia Aumentante:*

Se a cadeia encontrada for uma cadeia aumentante forte, então inverta as funções do *matching* M_k dos arcos desta cadeia. O vértice v não é mais exposto. Selecione outro vértice exposto e volte ao passo 2.

Caso contrário, selecione outro vértice exposto e volte ao passo 2.

PASSO 4 – *Ciclo Ímpar:*

Denote este ciclo ímpar por C_k. Faça a retração do ciclo ímpar C_k em um vértice artificial a_k. Denote o novo grafo por $G_k = (X_k, E_k)$. Considere o *matching* M_k em G_k, que consiste de todos os arcos em M_{k-1}(o conjunto *matching* anterior) que estão em G_k.

Nas próximas rotulações, todos os vértices retraídos no vértice artificial a_k terão o mesmo rótulo de a_k. Retorne ao passo 2 e continue o crescimento da árvore gerada pelo vértice v como em G_k, mesmo que este vértice seja artificial.

Neste passo, o ciclo ímpar encontrado é retraído em um vértice artificial, para que possam ser analisadas todas as cadeias aumentantes formadas por arcos com extremidades nos vértices do ciclo, o que possibilita um acréscimo no valor do peso final do *matching*.

PASSO 5 – *Árvore húngara:*

Seja

$$d_1 = \text{mín}\{y_i + y_j - p_{ij}\}$$

para todo arco (i, j), onde $i \in X_0$ é um vértice externo (E) e $j \in X_0$ é não rotulado. Se não existir um vértice $j \in X_0$ não rotulado, $d_1 = \infty$.

Considere

$$d_2 = \frac{1}{2}\,\text{mín}\{y_i + y_j - p_{ij}\}$$

para todo arco (i, j), onde $i, j \in X_0$ são vértices externos (E) que estão em vértices artificiais distintos. Se não existirem vértices rotulados como externos retraídos em vértices artificiais distintos, $d_2 = \infty$.

Seja

$$d_3 = \frac{1}{2}\,\text{mín}\{z_m\}$$

onde a minimização é tomada em todos os conjuntos de vértices V_m com cardinalidade ímpar que estão retraídos em um vértice artificial a_k que está rotulado como interno (I). Se não existem vértices artificiais rotulados como internos, $d_3 = \infty$.

Sejam

$$d_4 = \text{mín}\{y_i\}$$

para todos os vértices $i \in X_0$ que estão rotulados como externos (E) e

$$d = \text{mín}\{d_1, d_2, d_3, d_4\}.$$

Calcule e ajuste as variáveis duais como segue:

a) os valores das variáveis y_i de vértices externos (E) são subtraídos por d, ou seja, $y_i = y_i - d$.

b) os valores das variáveis y_i de vértices internos (I) são acrescidos por d, ou seja, $y_i = y_i + d$.

c) para cada vértice artificial externo (E), em G_k, acrescente a esta variável dual z_m o valor $2d$, ou seja, $z_m = z_m + 2d$.

d) para cada vértice artificial interno (I), em G_k, subtraia $2d$ do valor desta variável dual z_m, ou seja, $z_m = z_m - 2d$.

Se $d = d_1$, então o arco (i, j) que determina d_1 entra em E^* (pois, com o ajuste dos valores das variáveis duais, este arco passa a satisfazer a condição (3.3.9) e, portanto, é um novo elemento do conjunto E^*). Este arco pode ser inserido na árvore alternante gerada pelo vértice raiz v. Retorne ao passo 2 e continue o crescimento da árvore alternante gerada por v.

Se $d = d_2$, então o arco (i, j) que determina d_2 entra em E^* (pois o ajuste dos valores das variáveis duais fez com que este arco satisfaça a condição (3.3.9) da formulação matemática do problema do *matching*). Este arco pode ser inserido na árvore alternante, criando um ciclo ímpar. Retorne ao passo 2 e continue o crescimento da árvore alternante gerada por v.

Se $d = d_3$, então algumas variáveis duais z_i tornam-se nulas. Faça a expansão do vértice artificial correspondente a esta variável dual, voltando ao ciclo ímpar original. Seja $k = k + 1$. Denomine o grafo resultante por $G_k = (X_k, E_k)$. Seja o *matching* M_k contendo todos os arcos em M_{k-1} juntamente com os n_i arcos que formam o conjunto *matching* com os $2n_i$ vértices expostos de V_i. O vértice remanescente de V_i é colocado no conjunto *matching* M_k (tal vértice

sempre existe, pois V_i tem cardinalidade ímpar, igual a $2n_i + 1$, e o número máximo de *matching*s possível no ciclo é n_i) desde que todos os vértices artificiais internos (I) em G_{k-1} estejam no *matching* M_{k-1}.

Retorne ao passo 2 e continue o crescimento da árvore alternante gerada por v.

Se $d = d_4$, então as variáveis duais y_i de alguns vértices externos i tornam-se nulas. A cadeia encontrada na árvore alternante do vértice raiz v até o vértice i é uma cadeia aumentante neutra. O arco que contém v torna-se *matching* e o vértice i torna-se exposto, logo, $y_i = 0$. Retorne ao passo 2, escolhendo outro vértice exposto qualquer.

PASSO 6 – *Expansão dos Vértices Artificiais:*

Este passo é realizado somente depois que todas as condições de violação (3.3.10) forem examinadas pelo passo 2.

Considere todos os vértices artificiais contidos no grafo final. Faça a expansão de cada vértice artificial na ordem inversa (o último a ser gerado é expandido primeiro, e assim por diante) induzindo um *matching* de peso máximo no ciclo ímpar resultante de cada vértice artificial.

O *matching* final tem peso máximo para o grafo original G_0. Pare.

O algoritmo encontra a solução ótima quando não existirem mais vértices expostos com variáveis duais positivas, o que implica na inexistência de cadeias aumentantes fortes. Logo, pelo teorema 3.1, o conjunto M encontrado é o *matching* de peso máximo.

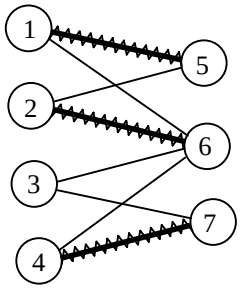
3.5. O PROBLEMA DA DESIGNAÇÃO

Nesta seção, um caso particular do problema do *matching* com pesos é apresentado. Considerando-se um grafo bipartido $G = (X^a \cup X^b, A)$, onde X^a e X^b são conjuntos independentes de vértices, os arcos $a_j = (w_i, w_k) \in A$ com $w_i \in X^a$ e $w_k \in X^b$ com pesos p_j, o objetivo deste problema é encontrar o *matching* de peso máximo (ou mínimo) com arcos do

grafo *G*. A versão deste problema com peso mínimo (ou custo mínimo) é mais conhecida na literatura como o problema da designação (CHRISTOFIDES [7]). A figura 3.12 mostra um exemplo de problema de designação.

Para resolver o problema da designação, o algoritmo do *matching* com pesos pode ser utilizado. A principal complicação encontrada no algoritmo do *matching* é a formação e a retração de ciclos ímpares, mas no problema da designação não existe possibilidade de formação de tais ciclos, pois o grafo deste problema é sempre bipartido, e o número de vértices que formam um ciclo é sempre par. Assim, o algoritmo da designação pode ser formulado de uma maneira mais simples, desconsiderando-se os ciclos ímpares.

FIGURA 3.12 - EXEMPLO DE UM PROBLEMA DE DESIGNAÇÃO.



3.5.1 Descrição do algoritmo da designação

O algoritmo descrito nesta seção, foi desenvolvido por KONIG [17], EVERGARY [14] e KUHN [18, 19] e é uma adaptação do algoritmo do *matching* com peso mínimo. A versão deste algoritmo para peso mínimo é conhecida na literatura como Método Húngaro, e as operações são feitas em uma matriz, chamada de matriz reduzida de custos.

PASSO 1 – *Inicialização:*

Seja M_0 o conjunto *matching* inicial, que não contém arcos do grafo $G = (X, A)$, onde $X = X^a \cup X^b$. Escolha qualquer valor inicial para as variáveis duais y_i, com $i \in X$, tais que $y_i{}^a + y_j{}^b \leq p_{ij}$, para todos os arcos (i, j).

Seja $k = 0$. Denomine o grafo original por $G_k = (X_k, E_k)$. Faça $k = k + 1$.

PASSO 2 – *Verificação do Vértice Exposto:*

Selecione qualquer vértice v, não artificial, exposto no grafo G_k, com $y_v > 0$. Se não existir tal vértice, vá para o passo 5.

Caso contrário, seja E^* o conjunto de todos os arcos (i, j) em G_k tais que

$$y_i{}^a + y_j{}^b = p_{ij}, \quad \forall (i, j), \text{ onde } i \in X^a \text{ e } j \in X^b \tag{3.6.1}$$

A partir do vértice v (rotulado como externo, E), crie uma árvore alternante gerada por v usando somente arcos em E^*.

Se uma cadeia aumentante for encontrada, vá para o passo 3.

Se uma árvore húngara for encontrada, vá para o passo 4.

PASSO 3 – *Cadeia Aumentante:*

Se a cadeia encontrada for uma cadeia aumentante forte, então inverta as funções do *matching* M_k dos arcos desta cadeia. O vértice v não é mais exposto. Selecione outro vértice exposto e volte ao passo 2.

Caso contrário, selecione outro vértice exposto e volte ao passo 2.

PASSO 4 – *Árvore húngara*:

Sejam

$$d_1 = \text{mín}\{p_{ij} - y_i{}^a - y_j{}^b\}$$

para todo arco (i, j), onde $i \in X^a$ é um vértice externo (E) e $j \in X^b$ é não-rotulado;

$$d_2 = \text{mín}\{y_i\}$$

para todos os vértices $i \in X^a$ que estão rotulados como externos (E) e

$$d = \text{mín}\{d_1, d_2\}.$$

Calcule e ajuste as variáveis duais como segue:

a) os valores das variáveis $y_i{}^a$ de vértices rotulados como externos (E) são subtraídos por d, ou seja, $y_i{}^a = y_i{}^a - d$.
b) os valores das variáveis $y_i{}^b$ de vértices rotulados como internos (I) são acrescidos por d, ou seja, $y_i{}^b = y_i{}^b + d$.

Se $d = d_1$, então o arco (i, j) que determina d_1 entra em E^*. Este arco pode ser inserido na árvore alternante. Retorne ao passo 2 e continue o crescimento da árvore alternante gerada pelo vértice v.

Se $d = d_2$, então as variáveis duais $y_i{}^a$ de alguns vértices externos i tornam-se nulas. A cadeia encontrada na árvore alternada do vértice raiz v até o vértice i é uma cadeia aumentante neutra. O arco que contém v torna-se *matching* e o vértice i torna-se exposto, logo, $y_i = 0$. Retorne ao passo 2, escolhendo outro vértice exposto qualquer.

PASSO 5 - *Final*:

O *matching* final tem peso máximo para o grafo original G_0. Pare.

3.5.2 O método húngaro visto como o algoritmo do *matching* simplificado

Usualmente, as operações deste algoritmo são feitas em uma matriz de pesos (ou custos) $C_{n \times n}$, onde as linhas representam os n vértices de X^a e as colunas os n vértices de X^b.

Quando o número de elementos de cada conjunto de vértices difere, utilizam-se vértices artificiais com o objetivo de completar a matriz C. Os pesos das combinações dos vértices artificiais com os demais vértices são considerados iguais a ∞, pois o objetivo do problema é minimizar os pesos da designação final.

Inicia-se com um conjunto *matching* M_0 formado por arcos que satisfazem a condição (3.6.1). Para determinar o conjunto M_0, verifica-se qual é o menor elemento de cada linha *i* da matriz *C*. O menor elemento da linha *i* é subtraído em todas as entradas da linha *i* da matriz *C*. O mesmo procedimento feito nas linhas é feito nas colunas da matriz *C*. Este é o passo 1 do algoritmo do *matching* de peso mínimo, e quando $p_{ij} - y_i{}^a - y_j{}^b = 0$, o arco (i, j) entra no conjunto E^* (pois $y_i{}^a + y_j{}^b = p_{ij}$).

A matriz encontrada, C', contém vários zeros, os quais determinam o conjunto de arcos que podem formar um *matching* inicial M_0. Estes zeros são designados arbitrariamente, de tal modo que cada linha e cada coluna de C' pode ter apenas um elemento nulo designado (os demais elementos ficam eliminados), caso contrário não tem-se um *matching*. Esta matriz é denominada matriz reduzida de custos de *C*.

Na matriz C', as linhas que não tem zeros designados indicam vértices expostos no grafo *G*. Na figura 3.13 tem-se um exemplo da matriz reduzida C' com os respectivos custos c'_{ij} e os elementos das linhas e das colunas pertencentes, respectivamente, aos conjuntos X^a e X^b. Seja *j* uma linha sem designação. Esta linha é marcada e procura-se o zero que tenha sido eliminado nesta linha, no caso, o elemento c_{ji}, e marca-se a coluna *i*. Procura-se na coluna *i* um zero designado, no caso, c_{hi} e marca-se a linha *h*. Repete-se o mesmo procedimento até não ser mais possível encontrar zeros designados.

Este procedimento matricial é a determinação das árvores ou cadeias alternantes do algoritmo do *matching* com pesos. Nas marcações de linhas e colunas, determina-se qual é o caminho da cadeia ou árvore alternante. Neste exemplo, o caminho $j^a - i^b - h^a - k^b - n^a$ é uma árvore húngara.

Os custos de C' são dados por $c'_{ij} = c_{ij} - p_i{}^a - p_j{}^b$ e os cálculos para a determinação de *d* mínimo podem ser feitos com os elementos c'_{ij}, e por este motivo a matriz C' é chamada de matriz reduzida. As variáveis duais de vértices de X^a (vértices externos) são subtraídas por *d* e as variáveis de X^b (vértices internos) são acrescidas por *d*. Na matriz C', as entradas das linhas marcadas são subtraídas por *d* e as entradas das colunas marcadas são acrescidas por *d*.

FIGURA 3.13 – EXEMPLO DA RESOLUÇÃO DO PROBLEMA DA DESIGNAÇÃO NA FORMA MATRICIAL.

	1	2	…	i	…	j	…	k	…	n
1	c'_{11}	c'_{12}	…	c'_{1i}	…	C'_{1j}	…	c'_{1k}	…	c'_{1n}
2	0^*	c'_{22}	…	c'_{2i}	…	c'_{2j}	…	c'_{2k}	…	c'_{2n}
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
h	c'_{h1}	c'_{h2}	…	0*	…	c'_{hj}	…	0	…	c'_{hn}
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
j	c'_{j1}	c'_{j2}	…	0	…	c'_{jj}	…	c'_{jk}	…	c'_{jn}
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
k	c'_{k1}	c'_{k2}	…	c'_{ki}	…	c'_{kj}	…	c'_{kk}	…	c'_{kn}
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
n	c'_{n1}	c'_{n2}	…	c'_{ni}	…	c'_{nj}	…	0*	…	c'_{nn}

O procedimento repete-se enquanto existirem vértices expostos, ou seja, ainda existem linhas da matriz C' sem designação.

Os cálculos feitos para o problema da designação são os mesmos do problema do *matching* de peso mínimo, mas o problema da designação pode ser resolvido com o uso da matriz reduzida. O esforço computacional para os dois problemas é o mesmo e o algoritmo genérico do *matching* de peso máximo (ou mínimo) pode ser utilizado para resolver o problema da designação sem aumento de tempo computacional.

3.6 ALGUMAS ADAPTAÇÕES DO ALGORITMO DO *MATCHING*

Em qualquer aplicação prática de um algoritmo, o tempo computacional é um dos fatores mais importantes, pois um algoritmo programado de modo correto, mas que demora muito para apresentar resultados pode mostrar-se ineficiente para empresas, onde o fator tempo é muito importante. Por este motivo, os algoritmos devem ser programados de tal modo que seu tempo computacional seja o menor possível.

O número de operações envolvidas no algoritmo do *matching* de peso máximo é uma função polinomial do número de vértices do grafo, *n*. Quanto maior o número de

vértices, maior o número de comparações e construções de cadeias e árvores alternantes. O tempo computacional cresce, teoricamente, na ordem de n^4, embora na prática este crescimento seja um pouco menor, aproximadamente na ordem de n^3 (THACKER [26]). Além disso, o algoritmo sempre toma árvores alternantes e vértices expostos de maneira aleatória, aumentando o esforço computacional.

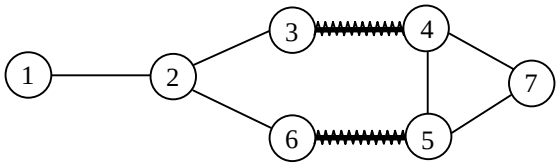
Com o objetivo de melhorar a performance computacional do algoritmo, algumas modificações apresentadas a seguir foram feitas neste trabalho, sem perder a generalidade do algoritmo e satisfazendo as condições impostas que garantem a solução ótima, e que foram mostradas nas seções anteriores deste trabalho.

Quanto à entrada dos dados, os vértices foram todos ordenados. No momento da escolha do primeiro vértice exposto, ao invés de ser escolhido um vértice, arbitrariamente, o algoritmo toma o primeiro vértice ordenado (uma outra abordagem equivalente pode ser feita tomando-se, por exemplo, o último da lista). Nos próximos passos, o algoritmo procura vértices expostos na ordem crescente da lista ordenada de vértices (ou decrescente, se a outra abordagem for utilizada). Desta forma, houve um ganho no tempo computacional, pois se a forma arbitrária fosse utilizada, o algoritmo poderia escolher arbitrariamente um mesmo vértice diversas vezes, mesmo que o vértice não fosse exposto.

Uma outra modificação no algoritmo foi feita com o objetivo de minimizar o número de ciclos ímpares, visto que a formação dos mesmos implica na retração e a expansão dos mesmos e o esforço computacional torna-se maior.

Para evitar ao máximo a formação de tais ciclos, o algoritmo pode encontrar um melhor vértice exposto para formar uma árvore alternante. Na figura 3.14, os vértices 1, 2 e 7 são expostos. Se o vértice 2 for escolhido, pode-se formar as cadeias aumentantes $C_1 = \{1, 2\}$, $C_2 = \{2, 3, 4, 7\}$ e $C_3 = \{2, 6, 5, 7\}$. Por outro lado, pode-se formar o ciclo ímpar $C_I = \{2, 3, 4, 5, 6\}$. O algoritmo evita a formação do ciclo ímpar pesquisando as possibilidades de escolhas de cadeias aumentantes que podem ser formadas. Deste modo, um ciclo ímpar só será formado se não existir outra escolha.

FIGURA 3.14 – EXEMPLO DE FORMAÇÃO DE UM CICLO ÍMPAR.



Este modo de evitar os ciclos ímpares não impede que o algoritmo encontre a solução ótima, pois os ciclos são sempre retraídos para que os arcos adjacentes aos vértices dos mesmos sejam pesquisados para encontrar o melhor *matching*, sendo a mesma idéia usada nos dois casos.

Deste modo, houve um ganho no tempo computacional, pois o número de operações e de comparações foi reduzido. Além disso, o número de grafos alternativos, com vértices artificiais também foi minimizado. Portanto, o algoritmo terá uma melhor performance, melhorando sua aplicação em problemas que necessitam de resultados mais imediatos, que é o caso do problema das escalas de motoristas de ônibus que será visto no próximo capítulo.

Uma das fases do problema das escalas de motoristas e cobradores de ônibus é o problema da designação dos finais de semana para cada escala formada em dias úteis, e a última fase é a designação de cada jornada semanal construída para um funcionário. Estes problemas foram resolvidos utilizando-se o algoritmo genérico do *matching* com pesos, sem perda de tempo computacional, pois como foi mostrado na seção anterior, os cálculos para a versão do problema da designação, feitos na matriz reduzida do grafo, são os mesmos do algoritmo genérico do *matching*.

No próximo capítulo, o problema das escalas de motoristas e cobradores de ônibus será mostrado, com suas restrições e as aplicações do algoritmo do *matching* de peso máximo em cada uma de suas fases.

CAPÍTULO IV

4. A APLICAÇÃO DO ALGORITMO DO *MATCHING* NO PROBLEMA DAS ESCALAS DE MOTORISTAS E COBRADORES DE ÔNIBUS

4.1. DESCRIÇÃO DO PROBLEMA

Neste capítulo, o problema da construção de jornadas semanais de trabalho para motoristas e cobradores de ônibus e suas formas de resolução são descritos. O objetivo principal das empresas de ônibus, na elaboração das jornadas, é construí-las de tal maneira que o custo total com motoristas e cobradores seja mínimo.

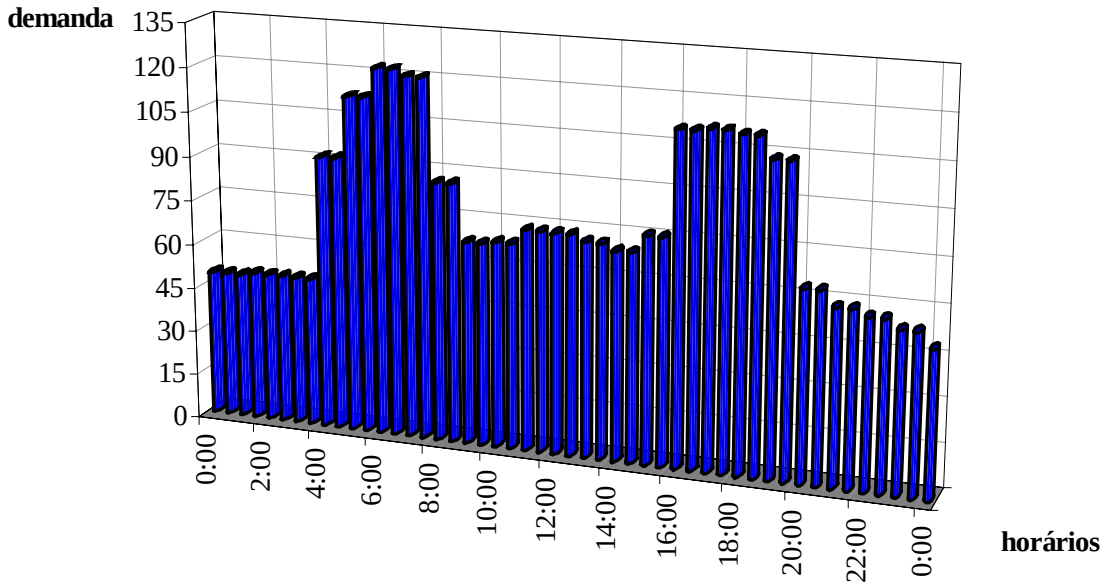
Para minimizar custos, um maior aproveitamento dos horários deve ser feito, reduzindo o total de horas ociosas e de horas extras, além da quantidade de motoristas e de cobradores.

As empresas de transporte coletivo da cidade de Curitiba utilizam as jornadas semanais de trabalho dos motoristas e cobradores com padrão ideal de duração igual a 36 horas. As jornadas semanais com duração maior do que o padrão ideal representam um custo maior para a empresa, pois o que excede as 36 horas é contabilizado como hora extra, e o valor pago pela hora extra é 50% maior. Por outro lado, outras jornadas semanais podem ter duração menor do que o padrão ideal, o que evidencia um número de horas ociosas, pois neste caso, a empresa paga pelas 36 horas.

O serviço de transporte urbano difere dos serviços com horários normais, pois tem características próprias. As escalas de horários dos ônibus são construídas de acordo com a demanda dos usuários, o que implica em horários de início e término desencontrados e escalas com durações variadas, o que impossibilita que todos os funcionários tenham 36 horas semanais de trabalho. Outros serviços dependem somente do horário de funcionamento da empresa, e as escalas de horários podem ter durações uniformes.

Existe uma grande demanda em alguns horários, como no início da manhã e no final da tarde. Muitos ônibus só circulam nestes horários e depois voltam à garagem, gerando algumas escalas de curta duração que são as denominadas jornadas curtas. Nos demais horários, existe demanda, embora seja menor, existindo alguns ônibus com escalas maiores, que são denominadas jornadas longas. Na figura 4.1, encontra-se um gráfico da demanda média de motoristas e cobradores em dias úteis, para os tipos de linhas de ônibus interbairros, convencionais, alimentadores e expressos de uma empresa da cidade de Curitiba.

FIGURA 4.1 – DEMANDA MÉDIA DE MOTORISTAS E COBRADORES EM DIAS ÚTEIS.



FONTE: Auto Viação Redentor Ltda.

Cada linha possui os chamados pontos de “rendição”, que são os locais onde são possíveis as trocas de motoristas e/ou cobradores. Estes locais são geralmente terminais de ônibus ou o ponto final da linha.

Para designar os horários dos ônibus aos respectivos motoristas e/ou cobradores, as escalas de longa duração devem ser divididas de tal forma que as escalas resultantes possam ser combinadas com as demais, formando as jornadas diárias de trabalho dos funcionários com custo mínimo. Estas divisões são feitas de acordo com os pontos de “rendição”

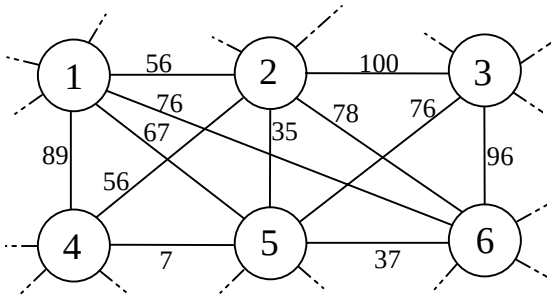
linha. Algumas empresas da cidade de Curitiba utilizam um programa que faz estas divisões de tal modo que pelo menos uma escala de curta duração possa ser gerada de cada escala de longa duração dividida. Este programa utiliza a Programação Dinâmica (CARNIERI [4, 5]). Esta é a primeira fase do problema, e não será abordada neste trabalho.

A necessidade do cumprimento da legislação trabalhista e de acordos com o Sindicato dos Trabalhadores impede, muitas vezes, que um melhor aproveitamento dos horários seja feito.

Existem três tipos de tabelas de escalas de horários, uma para dias úteis, outra para sábados e outra para os domingos.

As tabelas divididas são consideradas, e as escalas de curta duração resultantes devem ser combinadas para a formação da jornada diária de um motorista. Esta é a segunda fase do problema, e estas combinações são feitas através do algoritmo do *matching* de peso máximo. Cada escala representa um vértice v do conjunto de vértices V em um grafo $G = (V, A)$, e os arcos pertencentes ao conjunto A representam combinações factíveis de escalas. A figura 4.2 mostra um exemplo de representação de escalas em um grafo completo. Quando uma combinação de escalas não for possível, o arco entre essas duas escalas não existe, ou seja, o peso da combinação é nulo.

FIGURA 4.2 – EXEMPLO DE REPRESENTAÇÃO DE COMBINAÇÕES DE ESCALAS EM UM GRAFO.



Um peso máximo é atribuído aos arcos que representam combinações que têm durações iguais ao padrão ideal diário, de 6 horas. As demais combinações têm arcos com pesos proporcionais que variam de acordo com suas respectivas durações. Atualmente,

algumas empresas da cidade de Curitiba utilizam um algoritmo heurístico para fazer tais combinações, que é baseado na maneira com que as combinações eram feitas manualmente.

De acordo com as leis trabalhistas, um motorista deve ter, no máximo, duas escalas de curta duração em sua jornada diária, e estas devem ter um intervalo mínimo de 1 hora e máximo de 5 horas entre elas. As combinações que não estiverem neste intervalo, devem ter pesos nulos.

Outra lei trabalhista determina que um horário de final de semana pode ser designado para cada motorista e cobrador. Esta é a terceira fase do problema, e o modelo da designação foi utilizado como o algoritmo do *matching* de peso máximo simplificado, sendo atribuído um peso máximo às combinações que têm duração igual ao padrão ideal de 36 horas.

O tempo computacional não impediu a utilização destes algoritmos exatos, pois as empresas fazem as escalas separadamente, de acordo com as categorias de linhas de ônibus. As maiores categorias têm, aproximadamente, 400 escalas de trabalho.

Desta maneira, as jornadas semanais de trabalho de motoristas e cobradores de ônibus de cada empresa podem ser construídas, de forma a se obter um custo mínimo.

Após a construção das jornadas dos motoristas e dos cobradores, a última fase do problema consiste na designação de cada funcionário a uma jornada semanal de trabalho.

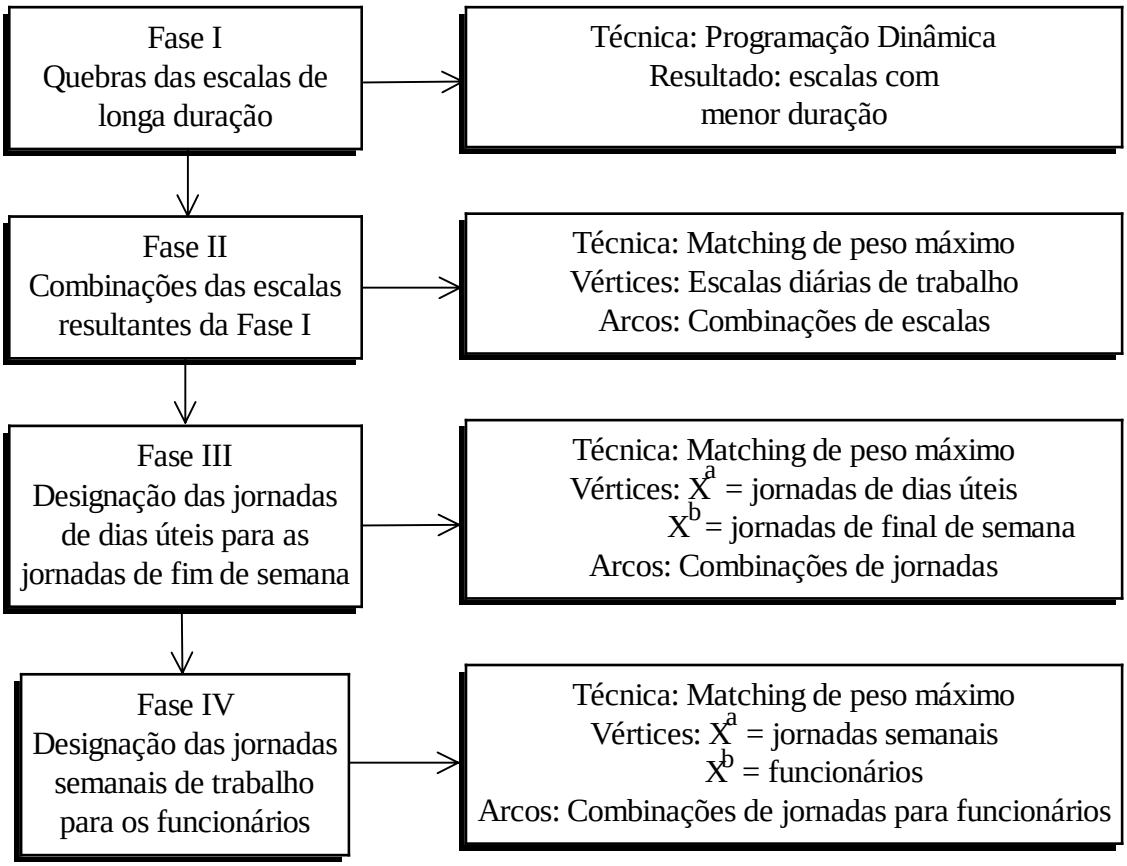
Novamente, o algoritmo do *matching* com pesos foi utilizado, atribuindo-se um peso máximo às designações de jornadas que mais se aproximam das jornadas que os funcionários faziam anteriormente. Objetiva-se, desta forma, minimizar as alterações nas jornadas de trabalho para os funcionários. A proximidade das jornadas anteriores com as atuais é verificada através dos horários de início e de término das jornadas, e comparando-se as linhas que os funcionários tinham anteriormente e as linhas atuais.

A cada intervalo de tempo que a nova jornada difere da jornada anterior de um funcionário, o peso da designação da nova jornada para o funcionário tem um desconto, penalizando-se, desta forma, o arco em questão. Por exemplo, a cada meia hora que a nova jornada inicia antes da jornada anterior, desconta-se 5 pontos do peso da designação do funcionário para a nova jornada.

Quando a linha da nova jornada for diferente da linha que o funcionário fazia, o peso da designação do funcionário para a nova jornada sofre um desconto.

Alguns funcionários não têm disponibilidade em todos os horários, pois os mesmos têm um outro emprego. Desta forma, a nova jornada de um funcionário deve ter o horário compatível com a disponibilidade do mesmo. Quando esta restrição for violada, o peso da designação do funcionário para a nova jornada é nulo.

Desta maneira, as jornadas construídas na primeira fase podem ser designadas aos funcionários, de tal modo que se aproximem ao máximo das jornadas que os funcionários faziam anteriormente. No fluxograma 4.1, encontram-se todas as fases do problema e as técnicas utilizadas para resolvê-las.



FLUXOGRAMA 4.1 – AS ETAPAS DA RESOLUÇÃO DO PROBLEMA.

Nas seções seguintes, cada uma destas fases para a resolução do problema e as técnicas utilizadas são detalhadas.

4.2. AS FORMAS DE ENTRADA E AS PARTICULARIDADES DOS DADOS

Quando um ônibus começa a operar em um determinado horário, o motorista deve dirigir-se à garagem da empresa para iniciar sua jornada de trabalho. Nestes casos, a empresa acrescenta meia hora na jornada de trabalho do motorista para que o mesmo possa deslocar-se até a garagem e conduzir o ônibus até o ponto de partida da linha. O mesmo ocorre quando a escala do ônibus termina, e o mesmo deve ser recolhido para a garagem.

Os cobradores devem prestar contas à empresa ao final de cada jornada de trabalho. Deste modo, o tempo de suas jornadas são incrementadas em meia hora a partir de seu término. A tabela 4.1 mostra um exemplo de tabelas de motoristas acrescidas ou não com meia hora. Nesta tabela, os pontos inicial e final de cada escala são numerados, e representam pontos finais das linhas ou terminais de ônibus. Somente nestes pontos são feitas as trocas de motoristas e cobradores.

TABELA 4.1 – EXEMPLO DE ESCALAS DE MOTORISTAS CORRIGIDAS, APÓS AS QUEBRAS.

Linha	Ponto inicial	Início da jornada	Início da tabela	Término da tabela	Término da jornada	Ponto Final	Duração
259	Garagem	03:40	04:10	04:50	05:20	Garagem	01:40
242	Garagem	04:58	05:28	09:28	09:28	24202	03:30
181	Garagem	05:00	05:30	10:50	10:50	00001	05:50
342	Garagem	04:40	05:10	08:58	09:28	Garagem	04:48
260	00013	08:45	08:45	16:15	16:15	00013	07:30
226	Garagem	05:50	06:20	09:20	09:20	00006	03:30
181	00001	10:50	10:50	13:20	13:20	18101	02:30

FONTE: Mostra parcial de uma tabela para dias úteis da empresa Transporte Coletivo Glória Ltda.

Para facilitar as operações com os horários e evitar erros de arredondamento, os horários de início e término das escalas, bem como as durações das jornadas são

transformados em minutos. Analogamente, os minutos podem ser transformados em decimais de hora. Esta operação facilita a comparação entre os horários de início e término de jornadas para que os pesos das combinações de jornadas possam ser calculados.

Com os dados mostrados nesta seção, podem ser verificadas as condições impostas pelas leis trabalhistas, com o objetivo de calcular os pesos para a aplicação do algoritmo do *matching* de peso máximo nas tabelas de dias úteis e de finais de semana.

4.3. AS REGRAS IMPOSTAS PELAS LEGISLAÇÕES TRABALHISTAS E SUAS APLICAÇÕES COMPUTACIONAIS

Quando duas escalas de curta duração são combinadas, as mesmas devem ter um determinado intervalo fixo. De acordo com as legislações trabalhistas e acordos com o Sindicato dos trabalhadores desta categoria, este intervalo deve ter no mínimo 1 hora e no máximo 5 horas de duração. Para as combinações de escalas que não estão neste intervalo, deve-se atribuir um peso nulo.

As combinações que sobrepõem escalas também devem ser evitadas e recebem peso nulo. A figura 4.3 mostra alguns exemplos de escalas que podem ou não ser combinadas.

Para que o intervalo mínimo de 1 hora seja satisfeito, deve-se calcular a diferença entre o início da escala *i* e o final da escala *j*. No exemplo da figura 4.3, as escalas 1 e 3 satisfazem esta condição e podem ser combinadas. Generalizando, pode-se descrever esta condição da seguinte forma:

$$Início(i) - Fim(j) \geq 1 \text{ hora} \qquad (4.3.1)$$

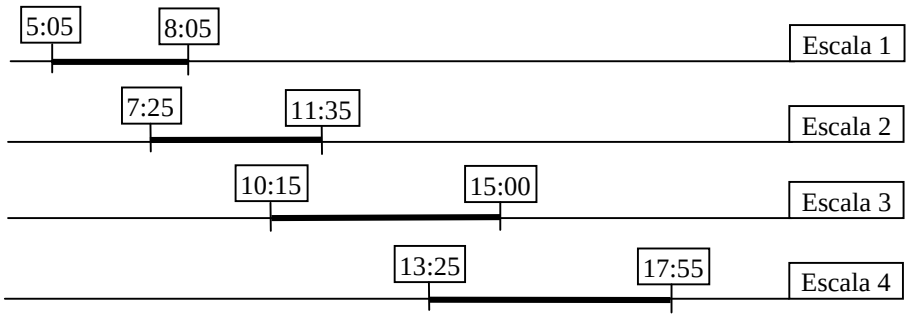
sendo *i* e *j* escalas distintas.

O intervalo máximo de 5 horas é satisfeito quando a diferença entre o início da escala *i* e o final da escala *j* for menor ou igual a 5. O início da escala *i* deve ser maior do que o final da escala *j*, em caso contrário, a diferença deve ser tomada entre o início da escala *j* e o final da escala *i*. No exemplo da figura 4.3, as escalas 1 e 4 não podem ser combinadas, pois o intervalo entre as mesmas é igual a 5:20 horas. Esta condição pode ser descrita do seguinte modo:

$$Início(i) - Fim(j) \leq 5 \text{ horas} \qquad (4.3.2)$$

onde *i* e *j* são escalas distintas e o início da escala *i* é maior do que o final da escala *j*. As condições (4.3.1) e (4.3.2) evitam a sobreposição de escalas, pois tais situações fornecem diferenças negativas entre o início de uma escala e o final de outra. No exemplo da figura 4.3, as jornadas 1 e 2 exemplificam uma sobreposição de jornadas.

FIGURA 4.3 – EXEMPLO DE COMBINAÇÕES DE ESCALAS.



Para que as combinações sejam feitas com o número mínimo de horas extras, cada empresa estipula uma duração máxima para as combinações de escalas. Esta duração varia de acordo com o tipo de tabela de cada empresa, e deve ser menor do que 8 horas, que é o valor da duração máxima de uma jornada, determinado pelas leis trabalhistas. Denotando-se esta duração como *Maior Duração*, tem-se a seguinte condição:

$$Duração(i) + Duração(j) \leq Maior\ Duração \qquad (4.3.3)$$

onde *i* e *j* são escalas distintas.

Outra regra imposta pelas legislações trabalhistas é que motoristas e cobradores devem ter, pelo menos, 11 horas de descanso entre dois dias consecutivos de trabalho. Como as empresas utilizam a maior combinação possível de duas jornadas com 8 horas de duração, esta condição não precisa ser verificada entre os dias úteis, pois a duração da maior combinação possível de escalas é de 8 horas, e o intervalo máximo é de 5 horas, resultando em uma jornada de trabalho com 13 horas de duração, onde o funcionário terá as 11 horas de descanso. No entanto, para a fase do trabalho onde a designação dos finais de semana para os dias úteis é feita, deve-se verificar se esta condição é satisfeita.

Com as condições descritas nesta seção, podem ser calculados os pesos das combinações de escalas para a aplicação do algoritmo do *matching* de peso máximo, evitando-se aquelas que não podem ser combinadas com a atribuição de pesos nulos.

4.4. A UTILIZAÇÃO DO ALGORITMO DO *MATCHING* DE PESO MÁXIMO PARA A FORMAÇÃO DAS JORNADAS DIÁRIAS DE TRABALHO

Algumas empresas elaboram as jornadas diárias manualmente, um trabalho demorado, exaustivo e que está sujeito a alguns erros de cálculo. As empresas citadas neste trabalho utilizam um algoritmo heurístico de combinações, baseado nas experiências dos funcionários que elaboravam estas tabelas manualmente.

O algoritmo heurístico inicia com as escalas que menos combinam com as demais, sempre procurando combinar aquelas que não formam horas extras. Quando duas escalas combinam, mas formam horas extras, procura-se a melhor combinação, isto é, aquela que forma o menor número de horas extras.

O padrão de jornada semanal de trabalho usado para os serviços de transporte coletivo em Curitiba, tanto para motoristas quanto cobradores é de 36 horas de trabalho. As jornadas semanais com duração total maior do que o 36 horas representam um custo maior para a empresa, pois esta diferença é contada como hora extra, e a empresa paga 50% a mais ao funcionário referente ao número de horas extras.

O padrão ideal de uma jornada diária para motoristas e cobradores é de 6 horas de duração, pois o mesmo tem 6 dias de trabalho por semana. Desta maneira, um peso máximo deve ser atribuído para as combinações de escalas com duração igual a 6 horas.

As demais combinações podem receber outro tipo de peso. A seguir, são mostradas as comparações entre algumas abordagens sobre os pesos aplicados ao algoritmo do *matching* de peso máximo e os resultados destas aplicações.

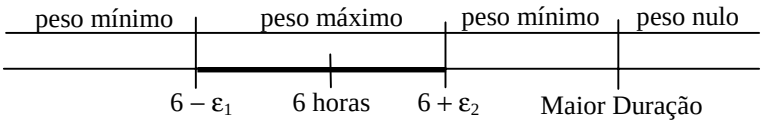
4.4.1. A utilização de pesos em intervalos de tempo

A primeira abordagem desenvolvida neste trabalho, para atribuição de pesos para a segunda fase do problema utiliza três tipos de pesos. Esta estrutura denomina-se pesos em intervalos de tempo.

Nesta estrutura, um peso máximo é atribuído às combinações de escalas que têm duração próxima de 6 horas. Esta proximidade é medida através dos parâmetros ε_1 e ε_2, e a empresa pode variar estes parâmetros de acordo com suas tabelas. Deste modo, se uma combinação de escalas tem o valor da duração total no intervalo $[6 - \varepsilon_1, 6 + \varepsilon_2]$, então seu peso será máximo. A figura 4.4 mostra a estrutura dos pesos em intervalos de tempo.

As combinações de escalas que têm duração menor do que $6 - \varepsilon_1$ ou maior do que $6 + \varepsilon_2$ recebem um peso mínimo, pois as escalas neste intervalo formam mais horas ociosas e horas extras, respectivamente.

FIGURA 4.4 – ESTRUTURA DOS PESOS EM INTERVALOS DE TEMPO.



Este peso mínimo deve ser maior ou igual à metade do peso máximo, pois se uma cadeia aumentante com três arcos for encontrada, onde dois arcos têm peso mínimo e um tem peso máximo, o algoritmo do *matching* combina as escalas com peso mínimo e a empresa reduz uma jornada de trabalho, e consequentemente, um funcionário. Deste modo, o número de motoristas é minimizado. Por outro lado, se o peso mínimo for menor do que a metade do peso máximo, o algoritmo não faz a troca, e uma jornada de trabalho a mais é feita.

Se duas escalas têm duração total maior do que a duração máxima, então seu peso é nulo.

O algoritmo usado para construir os pesos com intervalos é uma combinação das igualdades (4.3.1), (4.3.2) e (4.3.3) e pode ser escrito do seguinte modo:

Para $i = 1$ **até** n
    Para $j = 1$ **até** n
        Se $Início(i) > Fim(j)$
            Se $Início(i) - Fim(j) \geq$ 1 hora e
            $Início(i) - Fim(j) \leq$ 5 horas e
            $Duração(i) + Duração(j) \leq$ *Maior Duração*, **então:**
                Se $Duração(i) + Duração(j) < 6 - \varepsilon_1$ **ou**
                $Duração(i) + Duração(j) > 6 + \varepsilon_2$, **então:**
                $Peso(i, j) =$ Peso Mínimo
                Caso contrário:
                $Peso(i, j) =$ Peso Máximo
                Fim
            Caso contrário:
                $Peso(i, j) = 0$
            Fim
        Caso contrário:
            Se $Início(j) - Fim(i) \geq$ 1 hora e
            $Início(j) - Fim(i) \leq$ 5 horas e
            $Duração(i) + Duração(j) \leq$ *Maior Duração*, **então:**
                Se $Duração(i) + Duração(j) < 6 - \varepsilon_1$ **ou**
                $Duração(i) + Duração(j) > 6 + \varepsilon_2$, **então:**
                $Peso(i, j) =$ Peso Mínimo
                Caso contrário:
                $Peso(i, j) =$ Peso Máximo
                Fim
            Caso contrário:
                $Peso(i, j) = 0$
            Fim
        Fim
    Próximo j
Próximo i

Foram feitos diversos testes para os parâmetros ε_1 e ε_2, e os melhores resultados encontrados foram para $\varepsilon_1 = 20$ minutos e $\varepsilon_2 = 15$ minutos. Cada empresa utiliza um parâmetro de duração máxima de jornadas distinto, geralmente menor do que 8 horas.

Aplicando-se o algoritmo do *matching* com os mesmos parâmetros das empresas indicadas, obteve-se os resultados da tabela 4.2, que representam o número de jornadas diárias

de trabalho de cada tabela em cada empresa. Em algumas tabelas, o algoritmo heurístico obtém o mesmo resultado do algoritmo exato, mostrando que o algoritmo heurístico encontra o resultado ótimo em alguns casos.

TABELA 4.2 – RESULTADOS DA APLICAÇÃO DO ALGORITMO DO *MATCHING* COM PESOS EM INTERVALOS DE TEMPO.

	Dias Úteis		Sábados		Domingos	
	Heurístico	**Exato**	**Heurístico**	**Exato**	**Heurístico**	**Exato**
C. Sorriso	284	284	162	161	132	131
Redentor	291	291	165	162	134	133
Glória	305	299	174	174	151	151

NOTA: Os valores desta tabela representam o número de jornadas diárias de trabalho obtidos por cada algoritmo. O mesmo algoritmo heurístico está sendo utilizado por estas três empresas.

FONTE: Empresas Viação Cidade Sorriso Ltda., Auto Viação Redentor S.A. e Transporte Coletivo Glória Ltda.

O tempo computacional do algoritmo exato é, em média, 3 vezes maior do que do algoritmo heurístico. As tabelas são construídas separadamente, de acordo com as categorias das linhas de ônibus. As maiores tabelas utilizadas pelas empresas citadas neste trabalho têm, no máximo, 400 escalas. O algoritmo heurístico resolve o *matching* para dias úteis de uma tabela de 390 escalas, em um micro computador *Pentium* II de 400 MHz, em 30 segundos, e o exato em 1 minuto e 30 segundos nas mesmas condições que o heurístico. Como o tempo não é tão elevado e os resultados são melhores, o tempo computacional não é um problema.

Com relação a esta estrutura de pesos, a desvantagem encontrada nos resultados é o número elevado de horas extras e de horas ociosas que são formadas. Esta formação deve-se à penalidade igual à formação de poucas horas extras (ou ociosas) e de muitas horas extras (ou ociosas). Uma outra forma de estruturar os pesos para o algoritmo do *matching* é descrita a seguir, e sua construção procura amenizar o problema do grande número de horas extras e de horas ociosas encontrado nos pesos em intervalos de tempo.

4.4.2. A utilização de pesos proporcionais às durações das combinações de escalas de dias úteis

A segunda abordagem para a estrutura de pesos desenvolvida neste trabalho é chamada de pesos proporcionais às durações das combinações de escalas. Este tipo de abordagem sobre pesos para o *matching* tem a característica de penalizar mais a formação de horas extras e de horas ociosas nas escalas do que o método mostrado na seção 4.4.1. Portanto, o algoritmo do *matching* não estará minimizando o número de motoristas, e sim a quantidade total de horas extras e de horas ociosas.

As combinações de escalas que têm duração menor ou igual a 6 horas recebem um peso igual à sua duração. Deste modo, as combinações de escalas com curta duração têm pesos menores, o que evita a formação de muitas horas ociosas. Se *i* e *j* são escalas distintas, o peso da combinação (*i, j*) é dado por:

$$Peso(i, j) = Duração(i) + Duração(j) \tag{4.4.1}$$

onde as combinações com 6 horas receberão o peso máximo igual a 6.

Por outro lado, as combinações de escalas (*i, j*) que têm duração maior do que 6 horas recebem o seguinte peso:

$$Peso(i, j) = 6 - 1{,}5 \cdot (Duração(i) + Duração(j) - 6) \tag{4.4.2}$$

onde a constante 1,5 é utilizada como penalidade à formação de horas extras. O valor 1,5 tem relação com o custo que a empresa tem com as horas extras, ou seja, 50% a mais sobre o salário do funcionário. Na equação (4.4.2), o número de horas extras está sendo multiplicado à constante de penalidade 1,5 e quanto maior for o número de horas extras de uma combinação, menor será o seu peso e, consequentemente a sua possibilidade de combinar será pequena.

O algoritmo usado para formar os pesos para combinar as escalas de dias úteis utiliza as igualdades e desigualdades (4.3.1), (4.3.2), (4.3.3), (4.4.1) e (4.4.2) e pode ser escrito de forma análoga ao algoritmo de formação de pesos em intervalos de tempo, apresentado anteriormente. O algoritmo completo da formação deste tipo de pesos está descrito no Anexo II.

O algoritmo do *matching* de peso máximo não tem uma performance satisfatória com este tipo de pesos, pois como os mesmos procuram penalizar a formação de horas extras

e o algoritmo não combina algumas escalas que têm peso reduzido. Por este motivo, o número de jornadas não foi reduzido, e os resultados mostrados na tabela 4.3 evidenciam este fato.

TABELA 4.3 – RESULTADOS DA APLICAÇÃO DO ALGORITMO DO *MATCHING* COM PESOS PROPORCIONAIS ÀS DURAÇÕES DE ESCALAS.

	Dias Úteis		Sábados		Domingos	
	Heurístico	**Exato**	**Heurístico**	**Exato**	**Heurístico**	**Exato**
C. Sorriso	284	286	162	161	132	131
Redentor	291	294	165	163	134	133
Glória	305	307	174	174	151	151

NOTA: O mesmo algoritmo heurístico está sendo utilizado por estas três empresas

Após o cálculo dos pesos das combinações, tem-se uma idéia do maior peso que uma escala *i* possui, ou seja, pode-se saber qual é a escala que melhor combina com a escala *i* da tabela. A partir desta informação, todos os pesos das combinações das escalas com uma escala *i* são multiplicados por um coeficiente que faz com que o peso da melhor combinação da escala *i* tenha o peso máximo do grafo. Os demais pesos aumentam proporcionalmente a este coeficiente. Este coeficiente faz com que o peso da melhor combinação de uma escala *i* tenha peso máximo. Denotando-se o peso da melhor combinação da escala *i* com as demais escalas por *Maior Peso*(*i*) e o novo peso da combinação das escalas *i* e *j* por *Novo Peso*(*i, j*), os pesos das combinações de escalas podem ser recalculados através da seguinte regra de três:

$$6 - \textit{Maior Peso}(i)$$

$$\textit{Novo Peso}(i, j) - \text{Peso}(i, j)$$

ou seja,

$$\textit{Novo Peso}(i, j) = \frac{\textit{Peso Máximo} \cdot \textit{Peso}(i, j)}{\textit{MaiorPeso}(i)} \qquad (4.4.3)$$

onde *i* e *j* são escalas distintas.

Desta forma, o algoritmo do *matching* tem condições de fazer mais combinações, reduzindo o número total de motoristas sem aumentar demais o número de horas extras e de horas ociosas das jornadas de trabalho. A tabela 4.4 mostra os resultados da aplicação do

algoritmo do *matching* de peso máximo com a construção dos pesos proporcionais à duração das combinações e com os pesos corrigidos pela equação (4.4.3).

TABELA 4.4 – RESULTADOS DA APLICAÇÃO DO ALGORITMO DO *MATCHING* COM PESOS PROPORCIONAIS E CORRIGIDOS.

	Dias Úteis		Sábados		Domingos	
	Heurístico	**Exato**	**Heurístico**	**Exato**	**Heurístico**	**Exato**
C. Sorriso	284	284	162	161	132	131
Redentor	291	291	165	162	134	133
Glória	305	297	174	174	151	151

NOTA: O mesmo algoritmo heurístico está sendo utilizado por estas três empresas

O tempo computacional do algoritmo exato com este tipo de pesos é, em média, 4 vezes maior do que o heurístico. O algoritmo heurístico resolve o *matching* para dias úteis de uma tabela de 390 escalas, em um micro computador *Pentium* II de 400 MHz, em 30 segundos, e o exato em 2 minutos nas mesmas condições que o heurístico. Neste caso, o tempo computacional também não é tão elevado e os resultados são melhores do que o algoritmo heurístico e do que o algoritmo do *matching* aplicado com os pesos em intervalos de tempo.

As formações de horas extras e de horas ociosas podem ser reduzidas com um aproveitamento de horários, que consiste em uma redução ou ampliação das escalas que foram obtidas através das divisões das tabelas na fase inicial do problema. As empresas citadas neste trabalho utilizam este aproveitamento de horários, que é feito após a designação de dias úteis para finais de semana.

Com as jornadas de dias úteis e de finais de semana construídas, a terceira fase do problema consiste na designação de uma jornada de fim de semana para cada jornada de dia útil.

4.5. A UTILIZAÇÃO DO ALGORITMO DO *MATCHING* DE PESO MÁXIMO PARA A DESIGNAÇÃO DE JORNADAS DE DIAS ÚTEIS PARA AS JORNADAS DE FIM DE SEMANA

De acordo com as leis trabalhistas, cada funcionário, motorista ou cobrador de ônibus, pode ter uma jornada de fim de semana em sua jornada semanal de trabalho. Com as jornadas de dias úteis e finais de semana construídas através do algoritmo do *matching*, pode-se designar uma jornada de final de semana para cada jornada de dia útil.

Como foi visto no capítulo III, o modelo da designação pode ser resolvido pelo método húngaro, que é uma particularidade do algoritmo do *matching* com pesos quando aplicado em um grafo bipartido. Portanto, a designação das tabelas pode ser feita com o próprio algoritmo do *matching* de peso máximo, com as simplificações mostradas no capítulo III.

Fazendo-se a designação dos finais de semana para os dias úteis, tem-se as jornadas semanais completas de cada motorista ou cobrador. Por isso, deve-se calcular o número de horas extras e de horas ociosas para que as combinações entre as jornadas tenham pesos atribuídos.

Outra lei trabalhista assegura aos motoristas e cobradores de ônibus, pelo menos 11 horas de descanso entre dois dias consecutivos de trabalho. As jornadas que não têm o intervalo de 11 horas entre dia útil e fim de semana devem ter peso nulo.

Quando uma jornada de sábado for designada para uma jornada de dia útil, deve-se comparar o horário final da jornada de dia útil com o horário inicial da jornada de sábado. Esta condição pode ser escrita da seguinte forma:

$$(Início(j) + 24) - Fim(i) \leq 11 \text{ horas} \qquad (4.5.1)$$

onde *i* é uma jornada de dia útil e *j* uma jornada de sábado.

Quando uma jornada de domingo for designada para uma jornada de dia útil, a comparação deve ser feita com o horário final da jornada de domingo e com o horário inicial da jornada de dia útil. Pode-se escrever esta condição do seguinte modo:

$$(Início(i) + 24) - Fim(j) \leq 11 \text{ horas} \qquad (4.5.2)$$

onde *i* é uma jornada de dia útil e *j* uma jornada de domingo.

Os pesos utilizados para a designação são proporcionais às durações das escalas, priorizando aquelas que não formam horas extras e nem horas ociosas. A idéia é a mesma usada anteriormente na segunda fase do problema, e o número de horas extras e horas ociosas deve ser minimizado.

Sejam i uma jornada de dia útil e j uma jornada de final de semana. O valor da duração total da jornada para dias úteis deve ser multiplicado por 5, pois são 5 dias de trabalho. Assim, tem-se a duração total da jornada semanal dada por:

$$D = 5 \cdot Duração(i) + Duração(j). \tag{4.5.3}$$

Se D for menor ou igual a 36 horas, a combinação (i, j) recebe o peso:

$$Peso(i, j) = 5 \cdot Duração(i) + Duração(j) \tag{4.5.4}$$

onde as combinações que têm durações muito reduzidas terão peso também reduzido, pois formam horas ociosas.

Se D for maior do que 36 horas, a combinação (i, j) tem o peso:

$$Peso(i, j) = 36 - 1{,}5 \cdot (5 \cdot Duração(i) + Duração(j) - 36) \tag{4.5.5}$$

onde a constante 1,5 está relacionada com as horas extras formadas (pois a empresa paga 50% a mais sobre a hora extra), e quanto maior for o número de horas extras, menor o peso da combinação das escalas.

O algoritmo para construir os pesos da designação é uma combinação das equações e inequações de (4.5.1) a (4.5.5) e será descrito a seguir.

A matriz dos pesos deve ser simétrica[1], e no caso de grafos bipartidos sua forma geral é a seguinte:

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & \cdots & 0 & p(1,\, m+1) & p(1,\, m+2) & \cdots & p(1,\, m+n) \\ 0 & 0 & \cdots & 0 & p(2,\, m+1) & p(2,\, m+2) & \cdots & p(n,\, m+n) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & p(m,\, m+1) & p(m,\, m+2) & \cdots & p(m,\, m+n) \\ p(m+1,\, 1) & p(m+1,\, 2) & \cdots & p(m+1,\, m) & 0 & 0 & \cdots & 0 \\ p(m+2,\, 1) & p(m+2,\, 2) & \cdots & p(m+2,\, m) & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ p(m+n,\, 1) & p(m+n,\, 2) & \cdots & p(m+n,\, m) & 0 & 0 & \cdots & 0 \end{bmatrix}.$$

[1] Pois a matriz de um grafo não direcionado é simétrica, e este tipo de grafo é utilizado pelo algoritmo do *matching* de peso máximo.

Nesta matriz, as *m* primeiras linhas e colunas são de dias úteis e as *n* restantes são de finais de semana.

As combinações entre jornadas de dias úteis devem ter pesos nulos, assim como as combinações entre as jornadas de fim de semana. Por isso, os elementos das matrizes quadradas formadas pelas *m* primeiras linhas e colunas e pelas *n* últimas linhas e colunas da matriz de pesos são nulos.

Utilizando-se esta matriz, o algoritmo do *matching* de peso máximo foi aplicado com as simplificações apresentadas no capítulo anterior. Uma outra possibilidade é usar a matriz reduzida de pesos, que é formada pelas *m* primeiras linhas e pelas *n* últimas colunas da matriz de pesos. A mesma matriz reduzida pode ser formada pelas *m* últimas linhas e pelas *n* primeiras colunas da matriz de pesos.

A construção da matriz de pesos foi feita com o algoritmo descrito no Anexo III. Tal algoritmo é uma combinação das igualdades e desigualdades de (4.5.1) a (4.5.5).

Os pesos das combinações de jornadas são recalculados através da expressão (4.4.3), do mesmo modo que foram construídos os pesos para as combinações entre as escalas diárias de trabalho, na segunda fase do problema. Com isto, o número de designações deve ser máximo, evitando que muitas jornadas de dias úteis e de finais de semana fiquem isoladas.

Com os resultados da tabela 4.4, pode-se calcular o número de motoristas que cada empresa necessita para cumprir uma tabela.

Na empresa Transporte Coletivo Glória Ltda., tem-se 297 jornadas de dias úteis, 174 jornadas de sábados e 151 jornadas de domingos. Para cada jornada semanal de dia útil, uma jornada de final de semana foi designada. Neste caso, tem-se 297 jornadas de dias úteis e 325 jornadas de final de semana e, portanto, 28 jornadas de final de semana não têm designação. Em todas as tabelas das empresas de transporte coletivo de Curitiba, o número de jornadas de fim de semana é, em média, de 20% a 30% maior do que o número de jornadas de dias úteis. Deste modo, algumas jornadas de fim de semana ficam sem designação. Estas jornadas são escaladas para os funcionários plantonistas, que nos dias úteis fazem plantão na empresa, substituindo funcionários em casos de acidentes ou quando os mesmos não comparecem ao trabalho.

Para cada funcionário plantonista, devem ser designadas uma jornada de sábado e outra de domingo. Para que isto ocorra para todos os funcionários plantonistas, o número de jornadas de sábado sem designação, deve ser igual ao número de jornadas de domingo sem designação. Caso isto não ocorra, as empresas utilizam um algoritmo que faz trocas de jornadas de fim de semana designadas, procurando equilibrar o número de jornadas de sábado e de domingo sem designação. Esta é uma complementação final para a construção das escalas semanais dos funcionários. Deste modo, o número necessário de motoristas deve ser igual ao número de jornadas de dias úteis, que é sempre menor, somado com a metade do número de jornadas de fim de semana sem designação.

Na tabela 4.5, são mostradas as quantidades de motoristas que as empresas necessitam para cumprir a mesma tabela que foi construída na seção anterior. As tabelas mostradas neste trabalho são de motoristas de ônibus e as tabelas para cobradores podem ser construídas de maneira análoga às tabelas de motoristas, com as restrições adicionais mostradas na seção 4.3.

TABELA 4.5 – QUANTIDADE DE MOTORISTAS NECESSÁRIA EM CADA EMPRESA.

	Método heurístico		**Método exato**	
	Motoristas	**Plantonistas**	**Motoristas**	**Plantonistas**
C. Sorriso	289	5	288	4
Redentor	295	4	293	2
Glória	315	10	311	15

NOTA: O mesmo algoritmo heurístico é utilizado por estas três empresas

O número total de horas extras e horas ociosas é reduzido com a utilização de um aproveitamento de horários, que consiste em uma reconsideração de todos pontos de divisões das tabelas (semelhante ao método de divisões sucessivas, apresentado no capítulo II).

Através de trocas sucessivas, procura-se uma solução melhor, sem alterar as jornadas semanais construídas nas fases anteriores do problema. Além disso, as trocas não violam as restrições impostas pelas leis trabalhistas, descritas na seção 4.3.

Com as jornadas semanais construídas, a fase final do problema de construção das escalas de trabalho para motoristas e cobradores de ônibus é a designação dos funcionários às respectivas jornadas.

4.6. A DESIGNAÇÃO DOS FUNCIONÁRIOS PARA AS JORNADAS SEMANAIS DE TRABALHO

A última fase da construção de jornadas semanais para motoristas e cobradores de ônibus é a designação de cada funcionário a uma jornada de trabalho.

As empresas de transporte coletivo enfrentam problemas com alguns funcionários, a partir do momento em que são feitas modificações em suas jornadas de trabalho. Sempre existem novas escalas, outras são alteradas e algumas deixam de existir. Estas alterações são feitas de acordo com a demanda dos usuários. Com estas mudanças, as empresas fazem uma nova distribuição de jornadas de trabalho a seus funcionários, e as jornadas de trabalho de alguns funcionários mudam completamente.

Para amenizar estes problemas, as restrições impostas nesta fase do problema referem-se às preferências dos funcionários, e os pesos são atribuídos de acordo com as restrições de cada funcionário.

O algoritmo utilizado é novamente o *matching* de peso máximo, onde todas as designações possíveis[2] iniciam com peso máximo, que é descontado a cada restrição violada.

Nas empresas citadas neste trabalho, uma restrição imposta é a de que a nova jornada designada a um motorista ou cobrador, deve ser a mais próxima possível da sua jornada vigente, tanto em horário como em trajeto da linha.

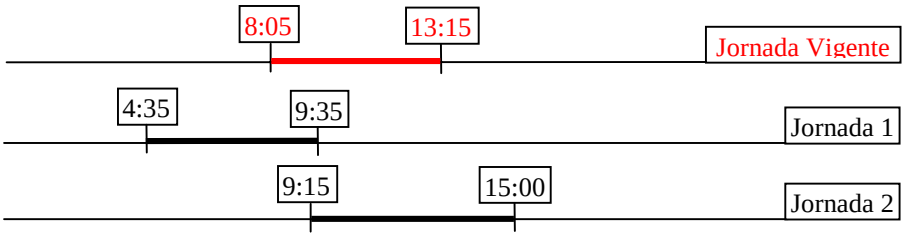
Para medir esta proximidade, a cada intervalo de tempo que a nova jornada difere da jornada vigente de um funcionário, o peso desta designação é descontado. Por exemplo, na figura 4.5, a jornada 1 tem 3:30 horas de diferença para a jornada que o funcionário tinha anteriormente. Se o peso máximo for igual a 100, e a cada meia hora de diferença desconta-se 10 pontos, a designação deste funcionário para a jornada 1 tem peso igual a 30.

[2] As designações possíveis ocorrem quando um funcionário pode trabalhar no horário e na linha da jornada.

Quando uma jornada tem o término posterior ao término da jornada vigente de um funcionário, deve-se descontar o peso da designação desta jornada para este funcionário.

As jornadas que têm horário de início posterior ao início da jornada vigente do funcionário não devem ter desconto na designação. Na figura 4.5, a jornada 2 inicia depois da jornada vigente do funcionário. Quando a escala termina antes do que a escala anterior do funcionário, o peso da designação também não deve ser descontado.

FIGURA 4.5 – EXEMPLO DE UMA COMPARAÇÃO ENTRE AS JORNADAS DE UM FUNCIONÁRIO.



As tabelas consideradas para esta verificação são de dias úteis, pois representam a maior parcela de trabalho do funcionário.

As restrições referentes aos descontos do peso da designação do funcionário *i* para a jornada de trabalho *j* podem ser escritas da seguinte forma:

Se $Início(Jornada\ Anterior(i)) > Início(j)$, **então**

$$Diferença = \frac{Início(Jornada\ Anterior(i)) - Início(j)}{Intervalo\ de\ tempo} \qquad (4.6.1)$$

$$Peso(i, j) = Peso(i, j) - Diferença \cdot Desconto1$$

Fim

Se $Fim(Jornada\ Anterior(i)) < Fim(j)$, **então**

$$Diferença = \frac{Fim(j) - Fim(Jornada\ Anterior(i))}{Intervalo\ de\ tempo} \qquad (4.6.2)$$

$$Peso(i, j) = Peso(i, j) - Diferença \cdot Desconto2$$

Fim

Uma outra restrição utilizada pelas empresas de transporte coletivo de Curitiba é a de que o funcionário deve permanecer com a mesma linha que tinha na jornada anterior. Se esta

restrição não for obedecida, o peso da designação do funcionário *i* para a jornada *j* sofre um desconto. Esta restrição pode ser escrita do seguinte modo:

Se *Linha*(*Jornada Anterior*(*i*)) ≠ *Linha Atual*(*j*), **então:**
Peso(*i*, *j*) = *Peso*(*i*, *j*) − *Desconto*3 (4.6.3)
Fim

Outra restrição importante refere-se aos horários disponíveis do funcionário. Na tabela de cadastros, cada funcionário indica os horários disponíveis para trabalhar na empresa. Além disso, existem os casos de funcionários que não podem ser designados para uma jornada de trabalho quando a mesma começa ou termina na madrugada. Para a combinação de uma jornada *j* para um funcionário *i* que não pode trabalhar na empresa naquele horário, o peso desta designação é nulo, ou seja,

Se (*Início*(*j*) > *Início Restrito*(*i*) **e** *Início*(*j*) < *Fim Restrito*(*i*)) **ou**
(*Fim*(*i*) > *Início Restrito*(*i*) **e** *Fim*(*j*) < *Fim Restrito*(*i*)), **então:**
Peso(*i*, *j*) = 0 (4.6.4)
Fim

Se um funcionário *j* mudar de jornada simples para jornada composta (combinação de duas jornadas menores), então o peso da designação do funcionário *j* para a jornada composta *i* tem um desconto que pode ser escrito da seguinte maneira:

Se *Tipo de Jornada*(*j*) = *Simples* **e** *Tipo de Jornada*(*i*) = *Composta*, **então:**
Peso(*i*, *j*) = *Peso*(*i*, *j*) − *Desconto*4 (4.6.4)
Fim

O conjunto de restrições de (4.6.1) a (4.6.4) permite a construção do algoritmo de montagem dos pesos, que está descrito com detalhes no Anexo IV.

O bom desempenho desta designação depende do uso dos parâmetros dos descontos nos pesos. Para as tabelas das empresas citadas neste trabalho, utilizou-se o conjunto de

parâmetros da tabela 4.6, e foram sugeridos pelas empresas de transporte coletivo. Os pesos atribuídos às designações podem ser vistos como a porcentagem de satisfação de um funcionário com a jornada de trabalho, pois o peso máximo é 100. O peso médio da solução final nas três empresas é de 90, e sob este ponto de vista, o nível de satisfação dos funcionários é de 90%. As empresas citadas neste trabalho utilizam um algoritmo heurístico, sem a atribuição de pesos, o que impede uma comparação entre os dois resultados.

TABELA 4.6 – PARÂMETROS USADOS NA DESIGNAÇÃO DE FUNCIONÁRIOS.

Peso Máximo	**100**
Desconto1	**10**
Desconto2	**10**
Desconto3	**20**
Desconto4	**50**

O tempo computacional desta designação foi de aproximadamente 25 minutos em um micro computador *Pentium* II, de 400 MHz com 305 tabelas e 353 funcionários. O algoritmo heurístico, específico para esta designação, leva em média 50 minutos para fazer a designação na mesma tabela.

A última restrição imposta é de que o funcionário deve ter folga no mesmo dia da tabela anterior, ou seja, em um sábado ou em um domingo. Esta folga é variável, e mensalmente troca-se o dia de folga dos funcionários. O problema é que as tabelas podem ser modificadas entre um mês e outro, e a folga do funcionário não pode ser alterada.

Em um último estágio, com as tabelas prontas e com as designações feitas, verificam-se as possibilidades de trocas de folgas entre os funcionários através de um módulo final de melhoria das tabelas.

Depois deste último estágio, as tabelas estão prontas e as jornadas semanais de trabalho de motoristas e cobradores de ônibus estão completas.

O esquema dado na figura 4.6 mostra um resumo de todas as fases da construção das jornadas de trabalho para motoristas e cobradores de ônibus.

FIGURA 4.6 – ESQUEMA GERAL DAS FASES DA RESOLUÇÃO DO PROBLEMA.

FASE INICIAL: Escolha das tabelas dos ônibus de certa categoria,
OBJETIVO: formar jornadas de trabalho, para designá-las aos motoristas e cobradores.

Dias úteis	Sábados	Domingos
...	...	...

⇓

1ª FASE: Quebra das escalas de longa duração.
RESULTADO: Escalas de menor duração.

Dias úteis	Sábados	Domingos
...	...	...

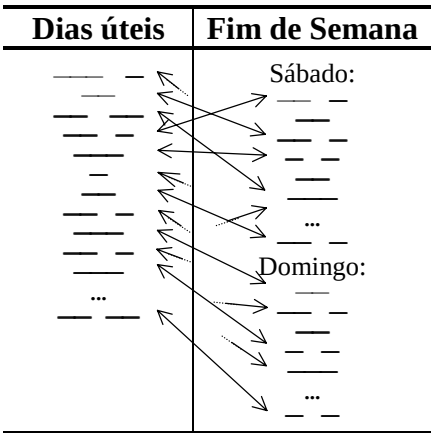
⇓

2ª FASE: Combinação das escalas de menor duração.
RESULTADO: Jornadas diárias de trabalho.

Dias úteis	Sábados	Domingos
...	...	...

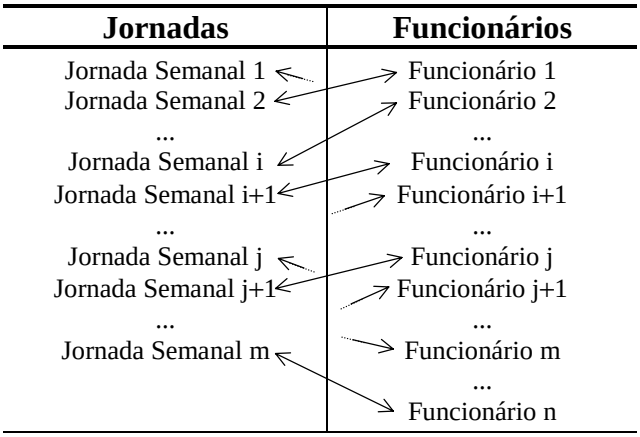
⇓

<u>3ª FASE</u>: Designação das jornadas de dias úteis para jornadas de fim de semana.
<u>RESULTADO</u>: Jornadas semanais de trabalho.



⇓

<u>4ª FASE</u>: Designação das jornadas semanais de trabalho para os funcionários.
<u>RESULTADO</u>: Tabela de tarefas completa.



CAPÍTULO V

5. CONCLUSÃO

O objetivo principal deste capítulo é fornecer a análise dos resultados apresentados no capítulo anterior, comparando-os com as soluções heurísticas das empresas de transporte coletivo mencionadas. Além disso, são fornecidas sugestões para a obtenção de melhores resultados.

5.1 Análise dos Resultados e conclusões

5.1.1 Resultados das combinações de escalas e das designações de jornadas de dias úteis para jornadas de fim de semana

A primeira abordagem sobre pesos para a combinação entre as escalas, e para a designação das jornadas de dias úteis para as jornadas de final de semana, foi a utilização dos pesos em intervalos de tempo. O número de horas extras e de horas ociosas foi, em média, igual ao dobro do número obtido pelas empresas. Este resultado era esperado, visto que este tipo de atribuição de pesos não penaliza a formação de horas extras e de horas ociosas de maneira adequada.

Neste caso, a solução encontrada neste trabalho foi o uso de pesos proporcionais às durações das escalas, cujos resultados foram melhores, tanto na quantidade de motoristas quanto no número de horas extras e de horas ociosas. Os resultados encontrados estão na tabela 5.1. Os pesos para a designação das jornadas de dias úteis para as jornadas de final de semana também foram proporcionais às durações das jornadas.

TABELA 5.1 – COMPARAÇÃO ENTRE OS RESULTADOS DAS COMBINAÇÕES DE ESCALAS COM OS ALGORITMOS EXATO E HEURÍSTICO.

	Motoristas	Horas extras	Horas ociosas
Algoritmo Heurístico	289	150	227
***Matching* com pesos em Intervalos de tempo**	288	320	350
***Matching* com pesos proporcionais às durações das escalas**	288	140	122

FONTE: Empresa Viação Cidade Sorriso Ltda.

O objetivo principal desta etapa do problema foi alcançado, visto que o algoritmo exato do *matching* obtém melhores resultados do que o algoritmo heurístico.

5.1.2 RESULTADOS DA DESIGNAÇÃO DAS JORNADAS SEMANAIS DE TRABALHO PARA OS FUNCIONÁRIOS

Nesta fase do problema, obteve-se um nível de satisfação dos funcionários com as novas jornadas de trabalho de aproximadamente 90% nas três empresas citadas neste trabalho. Este índice é considerado bom pelas empresas, pois é impossível fazer uma designação cuja resposta final contém apenas pesos máximos.

Além disso, o tempo computacional do algoritmo exato foi menor do que o algoritmo heurístico.

O objetivo desta fase do problema foi alcançado, pois os resultados da designação dos funcionários para as jornadas de trabalho foram bons, e o tempo computacional foi menor do que o tempo utilizado pelo algoritmo heurístico.

5.2 SUGESTÕES PARA TRABALHOS FUTUROS

Com o objetivo de melhorar as soluções encontradas neste trabalho, são propostas algumas sugestões que poderão servir para estudo em outros trabalhos.

a) Melhorar a fase inicial do problema, que consiste nas quebras das jornadas de longa duração, da seguinte forma:

a1) testando-se as escalas resultantes de uma quebra, para verificar as possibilidades de combinação com as escalas já obtidas;

a2) fazer com que o algoritmo do *matching* faça as combinações das escalas juntamente com as divisões das jornadas.

b) Elaborar um algoritmo que faça as primeiras fases de uma vez, ou seja, faça as quebras das jornadas, as combinações das escalas e a designação das jornadas de dias úteis para final de semana. Desta forma, os resultados devem ser melhores, pois uma fase dependerá da outra.

ANEXO I

EXEMPLO NUMÉRICO DA APLICAÇÃO DO ALGORITMO DO *MATCHING* DE PESO MÁXIMO

Considerando-se o grafo dado pela figura A1.2, encontrar o *matching* de peso máximo deste grafo. A representação dos valores das variáveis duais de cada vértice e a rotulação de cada vértice é feita como na figura A1.1.

FIGURA A1.1 – REPRESENTAÇÃO GRÁFICA DAS ROTULAÇÕES E DOS VALORES DAS VARIÁVEIS DUAIS DE CADA VÉRTICE.

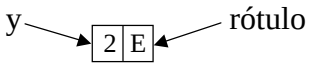
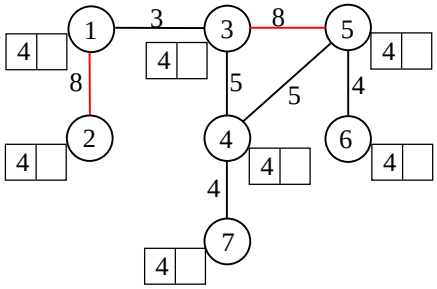


FIGURA A1.2 – GRAFO *G* NO PASSO INICIAL.



Passo 1:

O conjunto *matching* é vazio.

Atribui-se a metade do valor do peso máximo do grafo para cada y_i.

$$y_i = \frac{8}{2} = 4.$$

Passo 2:

Busca de um vértice exposto no grafo:

$E^* = \{(1, 2), (3, 5)\}$.

Vértice exposto: 3

A árvore gerada pelo vértice exposto deve conter apenas arcos do conjunto E^*.

Árvore gerada pelo vértice 3: 3 – 5

Rotulação: *Rótulo*(3) = E e *Rótulo*(5) = I

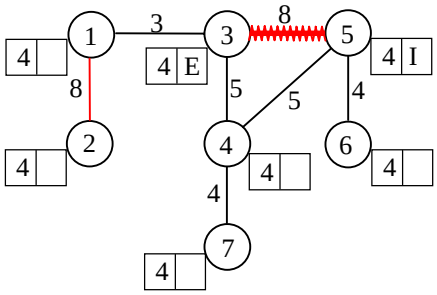
Esta é uma cadeia aumentante.

Passo 3:

Os arcos que não têm *matching* passam a ter, e os arcos que têm *matching* deixam de ter.

O arco (3, 5) entra no conjunto *matching* (Figura A1.3).

FIGURA A1.3 – GRAFO *G* NA PRIMEIRA ATUALIZAÇÃO.



Passo 2: Busca de um vértice exposto no grafo:

E^* = {(1, 2), (3, 5)}.

Vértice exposto: 4

Árvore gerada pelo vértice 4: 4

Rotulação: *Rótulo*(4) = E (como mostrado na figura A1.4)

Esta é uma árvore húngara.

Passo 5:

$d_1 = \text{mín}\{y_4 + y_3\ 2p(4, 3), y_4 + y_5\ 2p(4, 5), y_4 + y_7\ 2p(4, 7)\} = y_4 + y_3\ 2p(4, 3) = 3$

$d_2 = \infty$

$d_3 = \infty$

$d_4 = \text{mín}\{y_4\} = y_4 = 4$

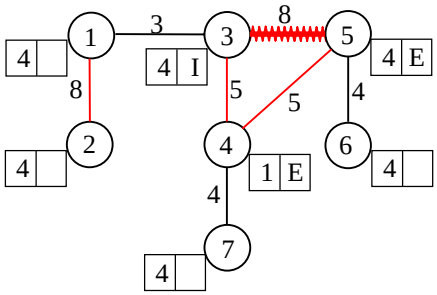
$d = \text{mín}\{d_1, d_2, d_3, d_4\} = d_1 = 3$

Atualização:

$y_4 = 4 - 3 = 1$

Como $d = d_1$, os arcos que representam este mínimo, (3, 4) e (4, 5), entram em E^* (figura A1.4).

FIGURA A1.4 – GRAFO *G* NA SEGUNDA ATUALIZAÇÃO.



Passo 2:

$E^* = \{(1, 2), (3, 5), (4, 3), (4, 5)\}$.

Vértice exposto: 4

Árvore gerada pelo vértice 4: 4 – 3 – 5 – 4

Rotulação: *Rótulo*(4) = E, *Rótulo*(3) = I, *Rótulo*(5) = E

Este é um ciclo ímpar.

Passo 4:

O ciclo ímpar é retraído em um vértice artificial A.

Retorna-se ao passo 2, com o vértice exposto artificial (Figura A1.5).

FIGURA A1.5 – GRAFO *G* NA TERCEIRA ATUALIZAÇÃO.
FIGURA A1.5a – GRAFO *G*. FIGURA A1.5b – GRAFO *G* MODIFICADO.

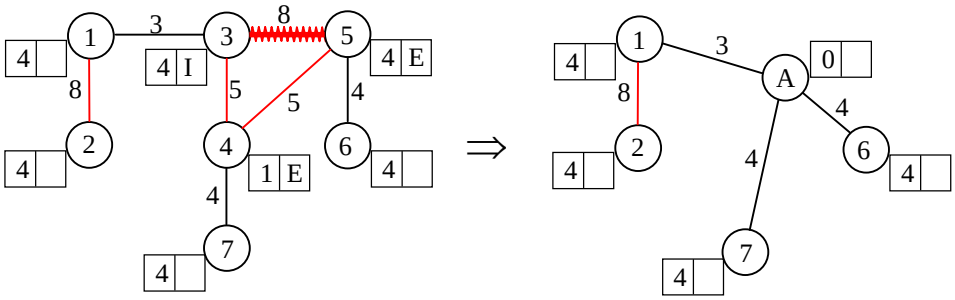


FIGURA A1.5a **FIGURA A1.5b**

Passo 2:

$E^* = \{(1, 2), (3, 5), (3, 4), (4, 5)\}$.

Vértice exposto: A

Árvore gerada pelo vértice A: A

Rotulação: *Rótulo*(A) = E, *Rótulo*(3) = E, *Rótulo*(4) = E, *Rótulo*(5) = E (figura A1.5a)

Esta é uma árvore húngara.

Passo 5:

$d_1 = \text{mín}\{y_3 + y_1\ 2p(3, 1), y_4 + y_7\ 2p(4, 7), y_5 + y_6\ 2p(5, 6)\} = y_4 + y_7\ 2p(4, 7) = 1$

$d_2 = \infty$

$d_3 = \infty$

$d_4 = \text{mín}\{y_3, y_4, y_5\} = y_4 = 1$

$d = \text{mín}\{d_1, d_2, d_3, d_4\} = d_1 = 1$

Atualização:

$y_3 = 4 - 1 = 3$

$y_4 = 1 - 1 = 0$

$y_5 = 4 - 1 = 3$

$z_A = 0 + 2 \cdot 1 = 2$

Como $d = d_1$, o arco que representa este mínimo, (4, 7), entra em E^*.

Passo 2:

$E^* = \{(1, 2), (3, 5), (4, 3), (4, 5), (4, 7)\}$.

Vértice exposto: A

Árvore gerada pelo vértice A: A – 7

Rotulação: *Rótulo*(A) = E, *Rótulo*(3) = E, *Rótulo*(4) = E, *Rótulo*(5) = E, *Rótulo*(7) = I

Esta é uma cadeia aumentante.

Passo 3:

Os arcos que não têm *matching* passam a ter, e os arcos que têm *matching* deixam de ter.

O arco (A, 7) entra no conjunto *matching* (Figura A1.6).

Passo 2:

$E^* = \{(1, 2), (3, 5), (4, 3), (4, 5), (4, 7)\}$.

Vértice exposto: 6

Árvore gerada pelo vértice 6: 6

Rotulação: *Rótulo*(6) = E (figura A1.6a)

Esta é uma árvore húngara.

FIGURA A1.6 – GRAFO *G* NA QUARTA ATUALIZAÇÃO.
FIGURA A1.6a – GRAFO *G*. FIGURA A1.6b – GRAFO *G* MODIFICADO.

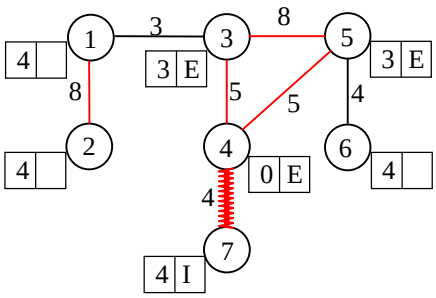


FIGURA A1.6a

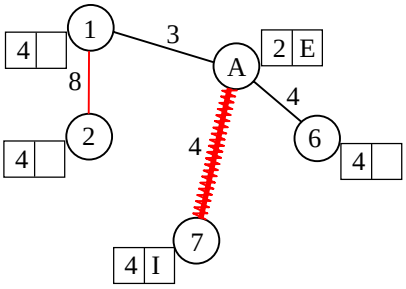


FIGURA A1.6b

<u>Passo 5</u>:

$d_1 = y_6 + y_5\ 2p(6, 5) = 3$

$d_2 = \infty$

$d_3 = \infty$

$d_4 = y_6 = 4$

$d = \text{mín}\{d_1, d_2, d_3, d_4\} = d_1 = 3$

<u>Atualização</u>:

$y_6 = 4 - 3 = 1$

Como $d = d_1$, o arco que representa este mínimo, (5, 6), entra em E^*.

<u>Passo 2</u>:

$E^* = \{(1, 2), (3, 5), (4, 3), (4, 5), (4, 7), (6, 5)\}$.

Vértice exposto: 6

Árvore gerada pelo vértice 6: 6 – A – 7

Rotulação: *Rótulo*(6) = E, *Rótulo*(A) = I, *Rótulo*(3) = I, *Rótulo*(4) = I, *Rótulo*(5) = I,

Rótulo(7) = E

Esta é uma árvore húngara (Figura A1.7).

FIGURA A1.7 – GRAFO *G* NA QUINTA ATUALIZAÇÃO.
FIGURA A1.7a – GRAFO *G*. FIGURA A1.7b – GRAFO *G* MODIFICADO.

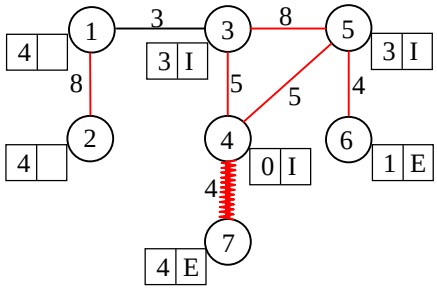


FIGURA A1.7a

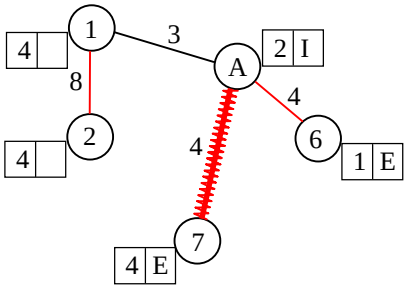


FIGURA A1.7b

<u>Passo 5</u>:

$d_1 = \infty$

$d_2 = \infty$

$d_3 = \frac{1}{2} z_A = 1$

$d_4 = \text{mín}\{\ y_6, y_7\ \} = y_6 = 1$

$d = \text{mín}\{d_1, d_2, d_3, d_4\ \} = d_3 = 1$

<u>Atualizações</u>:

$y_6 = 1 - 1 = 0$

$y_7 = 4 - 1 = 3$

$y_3 = 3 + 1 = 4$

$y_4 = 0 + 1 = 1$

$y_5 = 3 + 1 = 4$

$z_A = 2 - 2 \cdot 1 = 0$

Como $d = d_3$, a variável dual z_A torna-se nula, e o ciclo ímpar retraído neste vértice artificial deve ser expandido. O arco (3, 5) entra no *matching* (figura A1.8).

<u>Passo 2</u>:

$E^* = \{(1, 2), (3, 5), (4, 3), (4, 5), (4, 7), (6, 5)\}$.

Vértice exposto: 1

Árvore gerada pelo vértice 1: 1 – 2

Rotulação: *Rótulo*(1) = E, *Rótulo*(2) = I

Esta é uma cadeia aumentante.

FIGURA A1.8 – GRAFO *G* NA SEXTA ATUALIZAÇÃO.
FIGURA A1.8a – GRAFO *G* MODIFICADO. FIGURA A1.8b – GRAFO *G*.

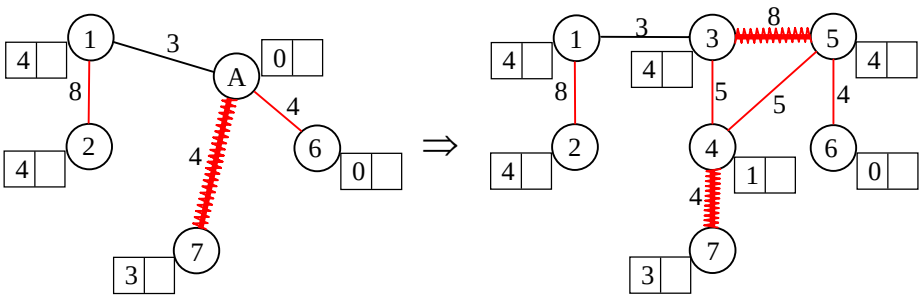


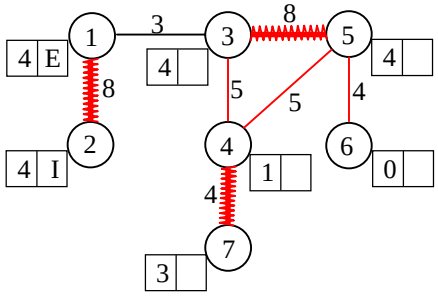
FIGURA A1.8a **FIGURA A1.8b**

Passo 3:

Os arcos que não têm *matching* passam a ter, e os arcos que têm *matching* passam a não ter.

O arco (1, 2) entra no conjunto *matching* (Figura A1.9).

FIGURA A1.9 – GRAFO *G* NA SÉTIMA ATUALIZAÇÃO.



Passo 2:

E^* = {(1, 2), (3, 5), (4, 3), (4, 5), (4, 7), (6, 5)}.

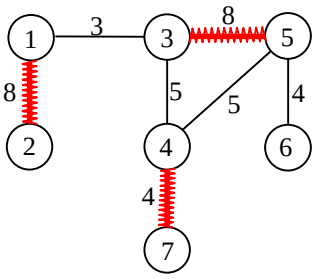
Não existe vértice exposto com a variável dual y positiva.

Passo 6:

Não existem vértices artificiais. Logo a solução encontrada é ótima (Figura A1.10). O peso total é dado por:

$$P = p(1, 2) + p(3, 5) + p(4, 7) = 20.$$

FIGURA A1.10 – SOLUÇÃO ÓTIMA.



ANEXO II

ALGORITMO PARA A CONSTRUÇÃO DOS PESOS PROPORCIONAIS ÀS DURAÇÕES DAS ESCALAS

Este algoritmo faz a construção dos pesos das combinações entre as escalas. Serve para formar as jornadas diárias de trabalho através do algoritmo do *matching* de peso máximo. Os parâmetros e as funções usadas estão definidas no Capítulo IV.

Sejam n o número total de escalas, *i* e *j* duas escalas distintas:

Para $i = 1$ **até** n
  Para $j = 1$ **até** n
    Se $Início(i) > Fim(j)$
      Se $Início(i) - Fim(j) \geq 1$ hora e
        $Início(i) - Fim(j) \leq 5$ horas e
        $Duração(i) + Duração(j) \leq Maior\ Duração$, **então:**
          Se $Duração(i) + Duração(j) > 6$ horas, **então:**
            $Peso(i, j) = 6 - 1{,}5 \cdot (Duração(i) + Duração(j) - 6)$
          Caso contrário:
            $Peso(i, j) = Duração(i) + Duração(j)$
          Fim
      Caso contrário:
        $Peso(i, j) = 0$
      Fim
    Caso contrário:
      Se $Início(j) - Fim(i) \geq 1$ hora e
        $Início(j) - Fim(i) \leq 5$ horas e
        $Duração(i) + Duração(j) \leq Maior\ Duração$, **então:**
          Se $Duração(i) + Duração(j) > 6$ horas, **então:**
            $Peso(i, j) = 6 - 1{,}5 \cdot (Duração(i) + Duração(j) - 6)$
          Caso contrário:
            $Peso(i, j) = Duração(i) + Duração(j)$
          Fim
      Caso contrário:
        $Peso(i, j) = 0$
      Fim
    Fim
  Próximo j
Próximo i

ANEXO III

ALGORITMO PARA A CONSTRUÇÃO DOS PESOS PARA A DESIGNAÇÃO DE JORNADAS DE DIAS ÚTEIS PARA JORNADAS DE FINAIS DE SEMANA

Este algoritmo serve para construir os pesos das combinações entre as jornadas de dias úteis e finais de semana. Desta forma, as jornadas semanais de trabalho são obtidas através do algoritmo do *matching* de peso máximo. Os parâmetros e as funções utilizadas estão definidas no Capítulo IV.

Sejam m o número de jornadas de dias úteis e n o número de jornadas de finais de semana. Considere *i* uma jornada de dia útil e *j* uma jornada de final de semana.

Para $i = 1$ **até** m
  Para $j = m + 1$ **até** $m + n$
    Se *Fim de Semana*(j) = Sábado,
      ($Início(j) + 24$) $- Fim(i) \leq$ 11 horas **e**
      $Duração(i) + Duração(j) \leq$ *Maior Duração*, **então:**
        Se $Duração(i) + Duração(j) > 36$ horas, **então:**
          $Peso(i, j) = 36 - 1{,}5 \cdot (5 \cdot Duração(i) + Duração(j) - 36)$
        Caso contrário:
          $Peso(i, j) = Duração(i) + Duração(j)$
        Fim
    Caso Contrário:
        Se *Fim de Semana*(j) = Domingo,
          ($Início(i) + 24$) $- Fim(j) \leq$ 11 horas **e**
          $Duração(i) + Duração(j) \leq$ *Maior Duração*, **então:**
            Se $Duração(i) + Duração(j) > 36$ horas, **então:**
              $Peso(i, j) = 36 - 1{,}5 \cdot (5 \cdot Duração(i) + Duração(j) - 36)$
            Caso contrário:
              $Peso(i, j) = Duração(i) + Duração(j)$
            Fim
        Caso Contrário:
          $Peso(i, j) = 0$
        Fim
    Fim
  Próximo j
Próximo i

Para $i = 1$ **até** m
    Para $j = 1$ **até** m
        $Peso(i, j) = 0$
    Próximo j
Próximo i

Para $i = m + 1$ **até** n
    Para $j = m + 1$ **até** n
        $Peso(i, j) = 0$
    Próximo j
Próximo i

ANEXO IV

ALGORITMO PARA A CONSTRUÇÃO DE PESOS PARA A DESIGNAÇÃO DE FUNCIONÁRIOS PARA AS JORNADAS SEMANAIS DE TRABALHO.

Este algoritmo serve para construir os pesos das combinações entre as jornadas semanais de trabalho e os funcionários (motoristas ou cobradores). A designação das jornadas para os funcionários é feita através do algoritmo do *matching* de peso máximo. Os parâmetros e as funções utilizadas estão definidas no Capítulo IV.

Sejam um funcionário i e uma jornada semanal j.

$Peso(i, j) = Peso\ Máximo$

Se $Início(Jornada\ Anterior(i)) > Início(j)$, **então**

$$Diferença = \frac{Início(Jornada\ Anterior(i)) - Início(j))}{Intervalo\ de\ tempo}$$

$Peso(i, j) = Peso(i, j) - Diferença \cdot Desconto1$

Fim

Se $Fim(Jornada\ Anterior(i)) < Fim(j)$, **então**

$$Diferença = \frac{Fim(j) - Fim(Jornada\ Anterior(i))}{Intervalo\ de\ tempo}$$

$Peso(i, j) = Peso(i, j) - Diferença \cdot Desconto2$

Fim

Se $Linha(Jornada\ Anterior(i)) \neq Linha\ Atual(j)$, **então:**

$Peso(i, j) = Peso(i, j) - Desconto3$

Fim

Se ($Início(j) > Início\ Restrito(i)$ **e** $Início(j) < Fim\ Restrito(i)$) **ou**
($Fim(i) > Início\ Restrito(i)$ **e** $Fim(j) < Fim\ Restrito(i)$), **então:**

$Peso(i, j) = 0$

Fim

Se $Tipo\ de\ Jornada(j) = Simples$ **e** $Tipo\ de\ Jornada(i) = Composta$, **então:**

$Peso(i, j) = Peso(i, j) - Desconto4$

Fim

REFERÊNCIAS BIBLIOGRÁFICAS

[1] BERGMANN, D. R. and BODIN, L. (eds.), **Preprint workshop on automated technique for scheduling of vehicle operators for urban public transportation services**, Chicago, 1975.

[2] BLAIS, J. Y., LAPORTE, G., LESSARD, R., ROUSSEAU, J. M. and SOUMIS, F., **The problem of assigning drivers to bus routes in na urban transit system**, Report, n. 44, Centre de recherche sur les Transports, Université de Montréal, 1976.

[3] CAPRARA, A., FISCHETTI, M., TOTH, P., VIGO, D. and GUIDA, P. L., **Algorithms for railway crew management**, *Mathematical programming*, n. 79, p. 125-141, 1997.

[4] CARNIERI, C. e STEINER, M. T. A., **Um sistema de Programação de Escala para Motoristas de ônibus**, *Anais/ XX CNMAC* (Congresso Nacional de Matemática Aplicada e Computacional), Gramado – RS, 1997.

[5] CARNIERI, C. e STEINER, M. T. A., **A System for Bus Drivers Scheduling**, *Euro XV* (Informs XXXIV – Institute for Operations Research and the Managements Sciences), Barcelona, Espanha, 1997.

[6] CARRARESI, P. and GALLO, G., **A multilevel bottleneck assignment approach to the bus drivers' rostering problem**, *European Journal of Operation Research*, n. 16, p. 163-173, 1984.

[7] CHRISTOFIDES, N., **Graph theory, an algorithmic approach**, Academic Press, New York, 1975.

[8] CPLEX Reference Manual, **Using the CPLEX callable library and CPLEX mixed integer library**. CPLEX Optimization, Inc., Incline Village, NV, 1992.

[9] DERIGS, U. and ZIMMERMANN, U., **An augmenting path method for solving linear bottleneck assignment problems**, *Computing*, n. 19, p. 285-295, 1978.

[10] DESROCHERS, M. and SOUMIS, F., **A column generation approach to the urban transit crew scheduling problem**, *Transportation Science*, v. 23, n. 1, p. 1-13, 1989.

[11] DESROCHERS, M., **A new algorithm for the shortest path problem with resourse constraints**, Report 421A, Centre de recherche sur les transports, Université de Montréal, 1986.

[12] DESROCHERS, M., **La fabrication d'horaires de travail pour les conducteurs d'autobus par une méthode de génération de colonnes**, Ph.D. thesis, Report 470, Centre de recherche sur les transports, Université de Montréal, 1986.

[13] EDMONDS, J. and JOHNSON, E. L., **Matching: A well solved class of integer linear programs**, *Combinatorial Structures and their applications*, Gordon and Breach, New York, 1970.

[14] EVERGARY, E., **On combinatorial properties of matrices**, descrito por H. W. Kuhn, Princeton University, 1953.

[15] FISHER, M.L., **The Lagrangian relaxation method for solving integer programming problems**, *Management Science*, n. 27, p. 1-18, 1981.

[16] GAMACHE, M., SOUMIS, F., MARQUIS, G. and DESROSIERS, J., **A column generation approach for large-scale aircrew rostering problems**. *Operations Research*, v. 47, n. 2, p. 247-263, 1999.

[17] KONIG, D., **Theorie der endlichen und unendlichen graphen**, Chelsea, New York, 1950.

[18] KUHN, H. W., **The Hungarian method for the assignment problem**, Nav. Res. Log. Quart., n. 2, 1955.

[19] KUHN, H. W., **Variants of the Hungarian method for the assignment problem**, Nav. Res. Log. Quart., n. 3, 1956.

[20] LESSARD, R., ROSSEAU, J. M. and DUPUIS, D., "**Hastus I: A mathematical programming approach to the bus driver scheduling problem**", em [28].

[21] MANINGTON, B. and WREN, A., **A general method for bus crew scheduling**, em [1].

[22] MINIEKA, E., **Optimization algorithms for networks and graphs**, Marcel Dekker, New York, 1978.

[23] ROUSSEAU, J. M. (ed.), **Computer scheduling of public transport 2**, Elsevier, Armsterdam, 1985.

[24] ROUSSEAU, J., LESSARD, R., and BLAIS, J.Y., **Enhancement to the Hastus crew scheduling algorithm**, em [23].

[25] RYAN, D. M. and FOSER, B. A., **An integrer programming approach to scheduling**, em [1].

[26] THACKER, B. G., **Matchings in weights graphs**, M.Sc. Thesis, Imperial College, London University, 1972.

[27] WILHELM, E. B., **Overview of the Rucus package driver run cutting program (RUNS)**, em [1].

[28] WREN, A. (ed.), **Computer scheduling of public transport urban passenger vehicle and crew scheduling**, Elsevier, Amsterdan, 1981.

[29] WREN, A., SMITH, B. M. and MILLER, A. J., **Complementary approaches to crew scheduling**, em [23].

[30] ZIONTS, S., **Linear and integer programming**, Prentice-Hall Inc., New Jersey, 1974