PROPOSTA DE SISTEMA DE SUPORTE À DECISÃO PARA ADAPTAÇÃO DE HORÁRIOS EM TRANSPORTE URBANO UTILIZANDO LÓGICA FUZZY

Trabalho de Conclusão de Curso

Engenharia da Computação

Nome do Aluno: Igor de Lima e Souza
Orientador: Prof. Mêuser Jorge Silva Valença

Universidade de Pernambuco
Escola Politécnica de Pernambuco
Graduação em Engenharia de Computação

IGOR DE LIMA E SOUZA

PROPOSTA DE SISTEMA DE SUPORTE À DECISÃO PARA ADAPTAÇÃO DE HORÁRIOS EM TRANSPORTE URBANO UTILIZANDO LÓGICA FUZZY

Monografia apresentada como requisito parcial para obtenção do diploma de Bacharel em Engenharia de Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Recife, novembro de 2012.

Agradecimentos

Agradeço a todos os meus professores que me guiaram durante esta jornada, me proporcionando uma visão singular através do conhecimento científico e questionamentos que me foram atribuídos.

Darei continuidade a minha caminhada repassando esta experiência para todos que possuem sede de aprendizado que cruzarem meu caminho, dividindo um pouco desta eterna dívida, cifrada em informação, para o bem comum.

Escola Politécnica de Pernambuco
Graduação em Engenharia de Computação
Projeto de Final de Curso

ESCOLA POLITÉCNICA
DE PERNAMBUCO

MONOGRAFIA DE FINAL DE CURSO

Avaliação Final (para o presidente da banca)*

No dia 14 de 12 de 2012, às 10:00 horas, reuniu-se para deliberar a defesa da monografia de conclusão de curso do discente IGOR DE LIMA E SOUZA, orientado pelo professor Mêuser Jorge Silva Valença, sob título Proposta de Sistema de Suporte à Decisão para Adaptação de Horários em Transporte Urbano Utilizando Lógica Fuzzy, a banca composta pelos professores:

Sérgio Campello Oliveira

Mêuser Jorge Silva Valença

Após a apresentação da monografia e discussão entre os membros da Banca, a mesma foi considerada:

☒ Aprovada ☐ Aprovada com Restrições* ☐ Reprovada

e foi-lhe atribuída nota: 10,0 (Dez)

*(Obrigatório o preenchimento do campo abaixo com comentários para o autor)

O discente terá 07 dias para entrega da versão final da monografia a contar da data deste documento.

SÉRGIO CAMPELLO OLIVEIRA

MÊUSER JORGE SILVA VALENÇA

* Este documento deverá ser encadernado juntamente com a monografia em versão final.

Resumo

Desde a antiguidade há questionamentos filosóficos de como seria o funcionamento para atribuição de decisões humanas [1] a partir uma grande quantidade de fatores. Com o avanço da tecnologia, a possibilidade de inteligência artificial em computadores obteve ênfase, gerando ferramentas para execução de tarefas sem necessidade de uma pessoa para definir soluções. Entre elas, a lógica difusa casou grande impacto, resolvendo diversos problemas [2] com contexto vago, onde não existe uma única resposta, mas pertinências entre vários níveis de certeza. Este trabalho visa apresentar a proposta de um sistema adaptável com foco no ajuste de horários em transporte urbano, utilizando lógica *fuzzy* à medida que o trânsito de uma localidade é modificado, precisando de compensação, sem a necessidade de alterar a infraestrutura urbana. O sistema propõe a melhoria do gerenciamento dos itinerários em grandes centros urbanos, onde há maior fluxo, consequentemente, focando no bem estar da maior parte da população. Objetiva intensificar a melhoria do serviço prestado pelas empresas de transporte urbano, assim como o desenvolvimento e utilização de sistemas inteligentes, para solucionar problemas do cotidiano, onde a adaptação com o meio não é levada em consideração.

Abstract

Since ancient times there are philosophical questions of how it works the human mind to make decisions [1] on several factors. With the advancement of technology, the possibility of artificial intelligence in computers got focus, creating tools to perform tasks without requiring a person to develop solutions. Among them, fuzzy logic had major impact solving problems [2] with vague context, where there is no single answer, but pertinence between various levels of certitude. This work presents a proposal for an adaptive system, focused in adjusting schedules for urban transport using fuzzy logic as the flow from a specific location is modified, needing to be compensated without change the urban infrastructure. The system aims to improve the itinerary management in large urban centers, where there is more flow, thus focusing on the welfare of most people. Featuring the goal of enhancing the services provided by urban transport companies, as well as the development and use of intelligent systems to solve real-world problems, where adaptation to the environment is not taken into consideration.

Sumário

Índice de Figuras

Índice de Tabelas

Tabela de Símbolos e Siglas

GPS – Global Positioning System (Sistema de posicionamento global)

GUI – Graphical User Interface (Interface Gráfica para o Usuário)

IEEE – Institute of Electrical and Electronics Engineers (Instituto de Engenheiros Eletricistas e Eletrônicos)

IPEA – Instituto de Pesquisa Econômica Aplicada

MIMO – Multiple inputs, multiple outputs (Múltiplas entradas e múltiplas saídas)

SISO – Single input, single output (Uma entrada e uma saída)

UML – Unified Modeling Language (Linguagem de Modelagem Unificada)

Capítulo 1

1 Introdução

A ciência, no intuito de desvendar a natureza, faz uso de modelos para explicar fenômenos e sistemas. Mas estes sistemas muitas vezes utilizam uma lógica que avalia a situação de forma limitada. O que acaba mostrando-se insuficiente para averiguar o mundo como ele realmente é [3].

A criação de modelos matemáticos é o tópico central de diversas disciplinas acadêmicas na engenharia, atribuindo entendimento ao comportamento de sistemas complexos com análise, modelagem e simulações. Porém diversos problemas práticos apresentam entradas com estimativas e parâmetros imprecisos em ambientes caóticos, dificultando a aproximação de uma solução com um sistema de modelo linear.

Com a inferência natural no controle de tarefas complexas por humanos, a ciência foi estimulada a dar um passo à frente da modelagem clássica, procurando técnicas alternativas para solução de tarefas que não possuem certeza dos atributos. Técnicas estas são chamadas de “inteligentes” não só por solucionar problemas de características e efeitos não lineares, mas por conseguir adaptar-se ao meio com o tempo. Dando um controle dinâmico de acordo com a realidade do problema e aproximando do processo dedutivo humano na formação do conhecimento, possibilitando o processamento computacional [4].

A partir da publicação de Lotfi A. Zadeh [5], em 1964, foi dado inicio a uma nova teoria de conjuntos com utilização de pertinências. Zadeh, professor da Universidade da Califórnia, já reconhecido e influente na comunidade acadêmica, começava a desenvolver uma lógica para solucionar problemas não lineares chamada de conjuntos nebulosos (*Fuzzy Sets*), atribuindo valores a variáveis sem definições precisas.

1.1 Descrição do problema

O fluxo de veículos em grandes centros urbanos, como exemplo de sistema complexo, é fator crucial para a definição de metas no dia a dia. Desta forma o usuário pode definir uma rotina semanal de circulação.

Um estudo divulgado em maio de 2011 pelo IPEA (Instituto de Pesquisa Econômica Aplicada) [6] aponta que 65% da população das capitais usam transporte público para se deslocar, e grande parte deste serviço é prestada por meio de ônibus. Porém, a malha rodoviária possui problemas, especialmente administrativos, que prejudica o bem estar e utilização durante a rotina dos usuários.

O transporte público urbano possui uma rotina semanal de circulação dividida em horários específicos predeterminados, mas contratempos fazem com que esses horários não sejam seguidos à risca. Sendo necessária modificação desta rotina à medida que vão surgindo eventos externos, empecilhos estes que podem determinar que um veículo permaneça mais tempo parado em um trecho ou atrasado em seu ponto de saída para circular novamente, denegrindo o sistema como o todo.

A proposta de um sistema de suporte à decisão de horários é essencial para manter o bem estar da população, modificando o paradigma estático na definição de itinerários. Utilizando como base o conhecimento adquirido por especialistas em fluxo urbano em cidades conturbadas, sendo modificado quando necessária adaptação. Respondendo uma pergunta simples, porém não trivial, de todo cidadão: “Que horas devo sair?”

1.2 Objetivo

Dado que os recursos das companhias de ônibus são escassos, a solução não é definida pela modificação da infraestrutura, mas pela melhoria do gerenciamento dos horários das linhas e adaptação do itinerário baseados nos atrasos mais comuns.

Em itinerários que se costuma ter atraso no horário de chegada, o sistema sugere que o veículo circule mais cedo que o horário de partida padrão para compensar o tráfego. Quando há adiantamento na chegada o problema é contornado com atraso na saída.

Contudo, a adaptação não pode ser definitiva, visto que as causas de alterações no horário são passíveis de mudanças. Horários de pico podem variar de acordo com a época do ano, o número de passageiros varia com eventualidades, além de diversas outras casualidades e fatores externos. Se a adaptação for dinâmica e controlada diretamente por um especialista, um sistema de decisão subjetivo pode estar sujeito a frequentes falhas na atribuição de novos valores para compensar atrasos e adiantamentos.

Visto que o problema possui inúmeras variáveis que interferem nas viagens, muitas delas complexas para serem avaliadas em separado e possuindo interferências estocásticas. A sugestão deste sistema é de administrar de uma forma mais generalizada em níveis de urgência de atrasos e adiantamentos. Abstraindo as variáveis de âmbito específico na construção de uma regra fixa de acordo com o peso que possuem, realizando alterações de rotinas e padrões em constante modificação ao longo do tempo com *soft computing*.

Um sistema de apoio à decisão focado na mudança e atualização de horários baseado em lógica difusa auxiliaria nas decisões utilizando cálculos *fuzzy*, desenvolvendo assim uma solução de suporte à decisão no gerenciamento de itinerários de transporte urbano, aperfeiçoando a qualidade do serviço prestado pelas empresas de ônibus, dando um passo a frente no bem estar da maioria da população.

1.3 Relevância prática de sistemas fuzzy

Atualmente muitas aplicações utilizam controle *fuzzy* em diversas áreas [7][9], [8]**Erro! Fonte de referência não encontrada.**. Reconhecida como ferramenta em scensão pelo IEEE (Instituto de Engenheiros Eletricistas e Eletrônicos), tem sido utilizada mundialmente em diversas aplicações em biomedicina, ambientes industriais, reconhecimento de padrões e diversas outras áreas. Provendo soluções especialmente em controle, automação, tecnologia, processamento de imagens e sinais, aumentando a eficiência e segurança.

Diversos produtos comerciais estão sendo desenvolvidos principalmente nos últimos 20 anos. Funções amplamente utilizadas estão presentes em câmeras digitais e celulares com foco, iluminação e *zoom* automático. Outros exemplos de aplicações de sucesso estão em reconhecimento de voz para digitação, grafoscopia em bancos para autenticação de assinaturas ou até o clássico controle de operação do metrô de Sendai desenvolvido pela Hitachi [9], no Japão, que opera diariamente desde 1987 prevendo com *fuzzy* a aceleração de maneira confortável para os passageiros e como diminuir o consumo de eletricidade calculando em tempo real a distância inicial e a desaceleração de acordo com a velocidade atual.

Capítulo 2

2 Teoria *fuzzy*

2.1 Teoria de conjuntos *fuzzy*

Para solucionar problemas com pequenas ambiguidades os conjuntos *fuzzy* generalizam valores *booleanos* definindo níveis de pertinência. A necessidade de existir definições vagas [10] é utilizada, pois a matemática tradicional falha quando não há exatidão de um conceito para resolver um problema determinado por entradas imprecisas.

Sendo uma alternativa das noções de pertinência e da lógica tradicional de origem filosófica clássica grega que possui definição bi valorada, o pensamento de forma difusa (*fuzzy)* retomou por Platão a discussão sobre a "lei do valor médio excluído" que define que toda preposição deve possuir um valor verdadeiro ou falso sem objeção.

Este conceito vago é quebrado em partes de acordo com a quantidade de divisões que forem necessárias para adicionar semântica a informação, dependendo sempre do problema e do conhecimento prévio de especialistas na área para resolvê-lo. As divisões entre conjuntos é essencial para definir um padrão de entendimento, porém temos que considerar o grau pertinência de cada conjunto para definir um conjunto *fuzzy* e suas interseções.

Um bom exemplo de um valor vago seria a definição de idade de uma pessoa como: novo, velho, criança ou adolescente. Uma pessoa com doze anos poderia ser definida como nova, criança e adolescente. Desta forma teríamos a interseção de três conjuntos e ainda sim não poderíamos inferir a proximidade da idade. Para ilustrar a Figura 1 demonstra um tipo de conceito vago baseado no exemplo de estipulação de idade utilizando a lógica clássica de conjuntos. Este conceito é suficiente para resolver problemas em diversas áreas, mas pode ser facilmente descartado pela falta de flexibilidade.

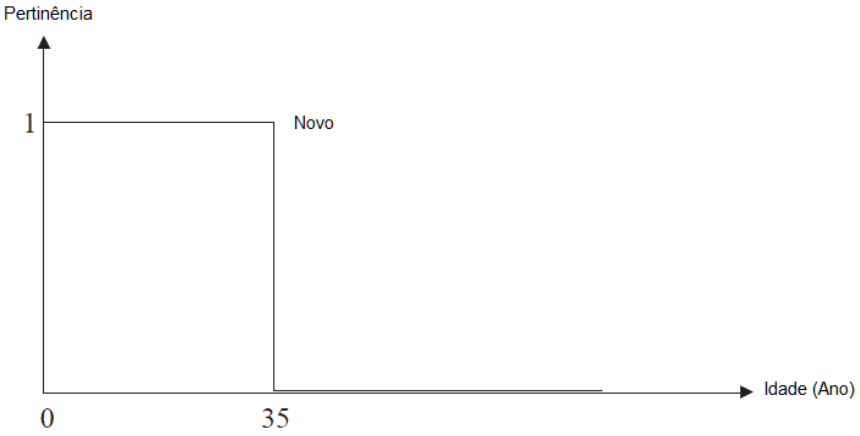

Figura 1. Exemplo de descrição vaga para idade utilizando lógica clássica de conjuntos.

Se definirmos a idade como pouco criança, não muito novo e quase adolescente e não velho poderíamos ter uma ideia de valor mais exato para deduzir. Ou melhor, afirmarmos 10% de pertinência em criança, 80% novo e 90% adolescente. Definindo assim um conjunto *fuzzy*, com valores de pertinência para cada variável de saída. Para ilustrar a Figura 2 demonstra a pertinência utilizando conceito vago de *fuzzy* baseado no exemplo de estipulação de idade generalizando o conceito clássico de conjuntos. Neste caso cada entrada está associada a um peso para ser processado, definindo a associação dos valores para se obter uma reposta.

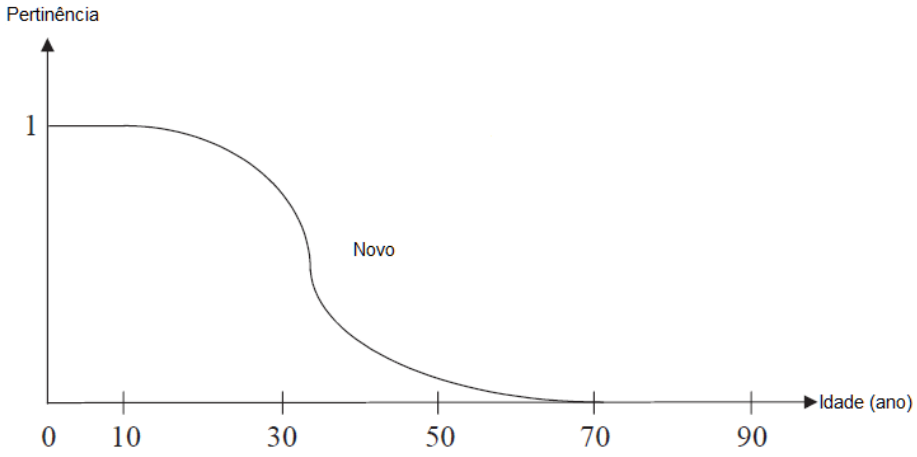

Figura 2. Exemplo de descrição vaga para idade utilizando conjuntos *fuzzy.*

Essas induções podem se alongar cada vez mais dependendo do problema [11] e da pessoa que define a idade de alguém ou participa da formulação dos dados de entrada. O problema pode ficar ainda mais complexo quando atribuído a pessoas diferentes, pois cada um pode possuir um conjunto *fuzzy* diferente para a mesma ideia.

2.2 Operações em conjuntos nebulosos

As operações e propriedades básicas de conjuntos nebulosos são similares aos de conjuntos clássicos, porém os operadores foram introduzidos por autores diferentes à medida que discussão sobre a estrutura, estabilidade [12], limitações **Erro! Fonte de referência não encontrada.** e aplicações foram desenvolvidas. As operações mais relevantes para dar embasamento a sistemas *fuzzy* são os operadores lógicos binários AND, OR que serão discutidos abaixo separadamente.

O operador mais utilizado no desenvolvimento deste protótipo foi inserido por Zadeh, em seu primeiro artigo sobre lógica *fuzzy* **Erro! Fonte de referência não encontrada.**, sugerindo a intersecção entre conjuntos, vide Figura 3, formando a união entre os membros de um conjunto A com um conjunto B. Esta operação é semelhante ao operador booleano AND descrito na tabela da verdade, Tabela 1, para comparação.

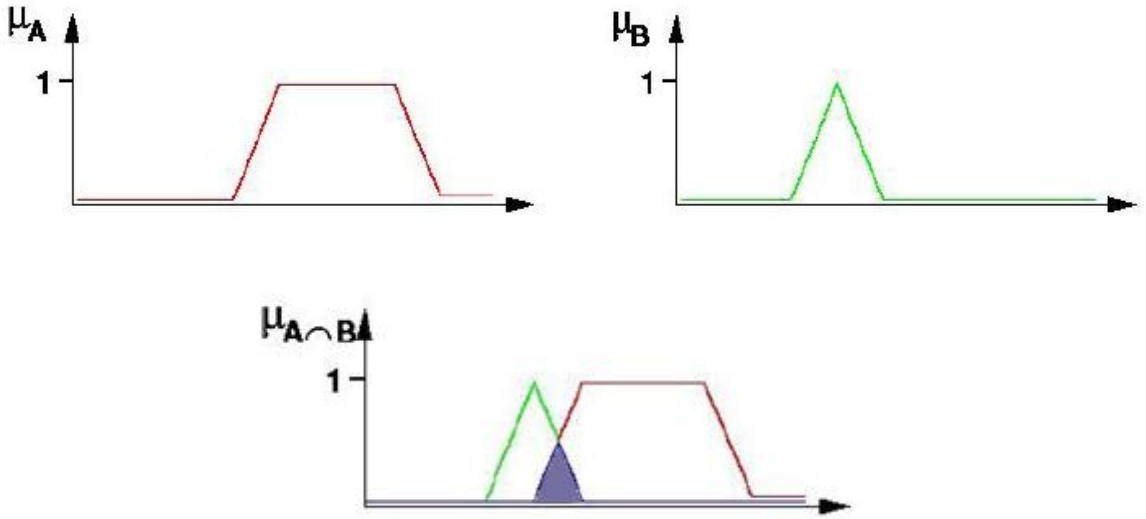

Figura 3. Intersecção entre dois conjuntos *fuzzy.*

Tabela 1. Tabela da verdade para operação AND.

Entrada (A)	Entrada (B)	Saída (A “and” B)
0	0	0
1	0	0
0	1	0
1	1	1

Com intuito de complementar o grau de pertinência em estruturas multivaloradas, e ampliar a forma de pensamento para sistemas distintos, foram adicionadas as operações de negação e união.

Semelhante à lógica OR, a união foi formulada por Lukasiewicz [15] para conjuntos *fuzzy* para cobrir a linha de pensamento de controladores *fuzzy* SISO (Uma entrada e uma saída) e MIMO (Múltiplas entradas e múltiplas saídas), possuindo o comportamento descrito na figura 4 e tabela 2 abaixo:

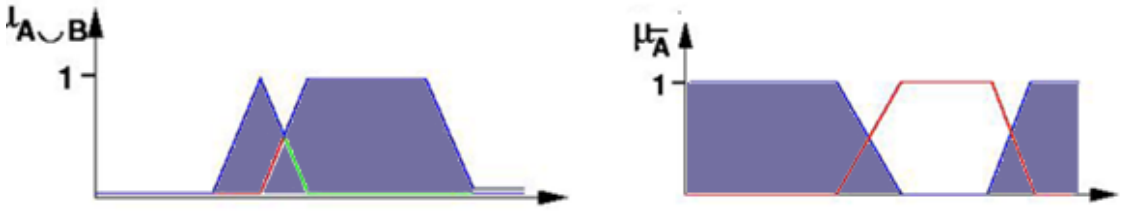

Figura 4. União e negação entre dois conjuntos *fuzzy.*

Tabela 2. Tabela da verdade para operação OR.

Entrada (A)	Entrada (B)	Saída (A “or” B)
0	0	0
1	0	1
0	1	1
1	1	1

2.3 Sistema fuzzy

Os sistemas *fuzzy* foram divididos em dois grupos: sistemas *fuzzy* puros, que foram desenvolvidos completamente com base na matemática difusa, e sistemas *fuzzy* híbridos, construídos a partir da mistura entre técnicas com variáveis difusas e sistemas lineares.

É importante diferenciar sistemas *fuzzy* de probabilidade, pois há similaridade na operação dos conjuntos entre intervalos. Os dois tipos podem possuir valores booleanos: verdadeiro e falso, entre 0 e 1. A visão probabilística define porcentagem de pertinência, como por exemplo: existe 50% de chance de João ser baixo, enquanto *fuzzy* infere o grau de pertinência com a altura baixa. Utilizando o mesmo exemplo podemos dizer que João possui 0,50 de pertinência em estatura baixa, porém isso não infere que João seja baixo ou alto.

No caso do protótipo desenvolvido podemos inferir que constitui de um sistema *fuzzy* puro, porém durante a *fuzzyficação* e *defuzzificação* houve ferramentas matemáticas lineares para determinar as funções de transferência com conjuntos difusos, com intuito de aumentar o controle do usuário no suporte a decisão durante a construção das regras, que podem variar entre regiões e políticas de empresa diferentes.

Segundo Mamdani [16], para computar uma resposta a partir de um sistema *fuzzy*, é necessário passar por seis passos básicos utilizando a informação de entrada de conjuntos não clássicos (*fuzzy*). Os passos são:

1. Determinar um conjunto de regras *fuzzy*.
2. *Fuzzificar* a entrada de dados utilizando conjuntos/subconjuntos *fuzzy*.
3. Combinar os valores *fuzzificados* de acordo com o conjunto de regras *fuzzy* determinados e estabelecer uma regra base.
4. Definir as consequências do padrão combinando a regra base com a saída das funções de transferência.
5. Combinar as consequências para obter uma distribuição de saída.
6. *Defuzzificar* as distribuição de saída (apenas se for necessário obter uma saída de conjuntos não nebulosos).

Para ilustrar, na Figura 5 há um diagrama contendo a estrutura básica a partir de um conjunto de regras baseado em um sistema *fuzzy* típico com múltiplas entradas e múltiplas saídas, dividindo as fases em blocos e fazendo referência as ferramentas utilizadas.

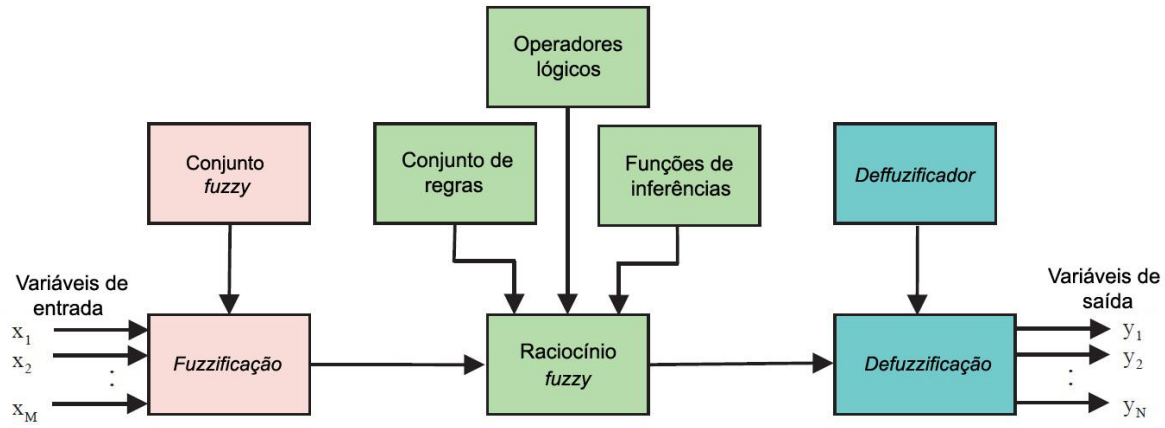

Figura 5. Estrutura típica de sistemas *fuzzy* (MIMO) baseada em conjunto de regras.

2.4 Conjunto de regras

Para criar regras utilizando as operações entre conjuntos *fuzzy,* foi necessário adicionar predicados para conexão destes conjuntos. A concepção de padrões é utilizada assim com regras linguísticas, fazendo novamente alusão ao pensamento humano para inferências, possuindo dois blocos para combinação similar à semântica utilizada para programação em computadores.

O conhecimento do especialista na área pode ser utilizado de forma análoga à semântica linguística, utilizando “SE” e “ENTÃO” para formular regras. A combinação de várias regras determina um conjunto de regras para o sistema, como exemplo a Tabela 3, a partir de conjuntos *fuzzy* possuindo características diferentes. Observa-se que apesar de considerar todos os valores de entrada, dependendo do sistema, não é necessário avaliar todas as combinações possíveis de entrada.

Tabela 3. Exemplo de conjunto de regras utilizando SE e ENTÃO.

Regra	Característica A	Característica B	Característica C	Característica D	Classe
1:	baixo	médio	médio	médio	classe1
2:	médio	alto	médio	baixo	classe2
3:	baixo	alto	médio	alto	classe3
4:	médio	médio	médio	médio	classe4
...:	...	...	...	...	...
N:	baixo	alto	médio	baixo	classeN

As conclusões representadas pelas classes são definidas de acordo com a combinação das funções de transferência [17] utilizando os operadores de conjuntos nebulosos (AND, OR, NOT) de acordo com necessidade do sistema, obtido a partir de valores *fuzzificados*.

2.5 Fuzzificação

O primeiro passo após coletar os dados é de *fuzzificar* a informação para abstrair valores determinísticos que possam estar entre as entradas. Este processo associa a informação de entrada do sistema aos conjuntos nebulosos, de forma que haja pertinência entre os valores.

Para abstrair esses valores de conjuntos clássicos, são utilizadas técnicas para determinar o grau de pertinência em um conjunto ou subconjunto. Um bom exemplo seria de uma pessoa no deserto com sede, que encontra uma garrafa informando que possui 0,9 de pertinência sem veneno. A quantidade de veneno estaria *fuzzificada*, pois há 10% de veneno nesta garrafa. Comparando com uma abordagem estatística, se outra garrafa possuísse 90% de probabilidade de veneno, seria mais interessante escolher esta garrafa, pois haveria a possibilidade de 10% de chance de haver veneno, enquanto a outra teria certamente veneno em pouca quantidade.

2.6 Regras de inferência (regra base)

Para definir saída de forma nebulosa, o sistema deve combinar todas as regras de inferência para gerar uma saída única ou múltipla, mas de forma nebulosa. A combinação desses parâmetros pode ser definida por diversas técnicas para se criar uma regra base, e depende do tipo de sistema que está sendo gerado para seleção estratégica mais adequada.

Há vários tipos de métodos que podem ser referenciados para se obter ligação entre funções de inferência **Erro! Fonte de referência não encontrada.**, REF _Ref341115005 \r \h **Erro! Fonte de referência não encontrada.**. Os mais utilizados são os modelos clássicos propostos por Mandami [20] e Sugeno [21]. Neste trabalho o foco será dado na inferência de Mandami e será descrito a razão no Capítulo 3, que fará referência a teoria explicada neste capítulo, com a utilidade prática do sistema e detalhes de implementação em código.

Essa inferência define a decisão do sistema e normalmente é aplicada em função das pertinências dos dados no conjunto nebuloso, sendo os resultados agregados por meio de uma operação matemática.

2.6.1 Inferência de Mandami (Max-min)

As regras de inferência definem qual a classe que vai ser determinada em um novo conjunto *fuzzy*, de acordo com a combinação das características do sistema por um especialista na área, porém o peso não é levado em consideração. Para isto o valor definido, sendo ele o máximo ou mínimo, determina a pertinência que falta para um novo conjunto, implicando na relação dos pesos entre as combinações.

2.7 Defuzzificação

Na maioria dos sistemas é necessária a obtenção de um valor *crisp* (não nebuloso) a partir da saída das regras de inferência. Com isso um passo a mais deve ser realizado para transferência da resposta nebulosa, dada por sistemas *fuzzy*, para um valor novamente com semântica booleana. Os métodos mais utilizados, descritos na Figura 6, são: o da média dos máximos, e o do centro de área.

A figura define que a partir de um conjunto *fuzzy*, criado a partiri da união entre dois conjunto, é possível identificar pelo método do máximo selecionando o ponto mais alto entre os dois conjuntos (MAX), a média dos máximos é inferida a partir da média dos maiores valores do conjunto (MOM) e o centro de massa (COA) corresponde ao centro de gravidade da união.

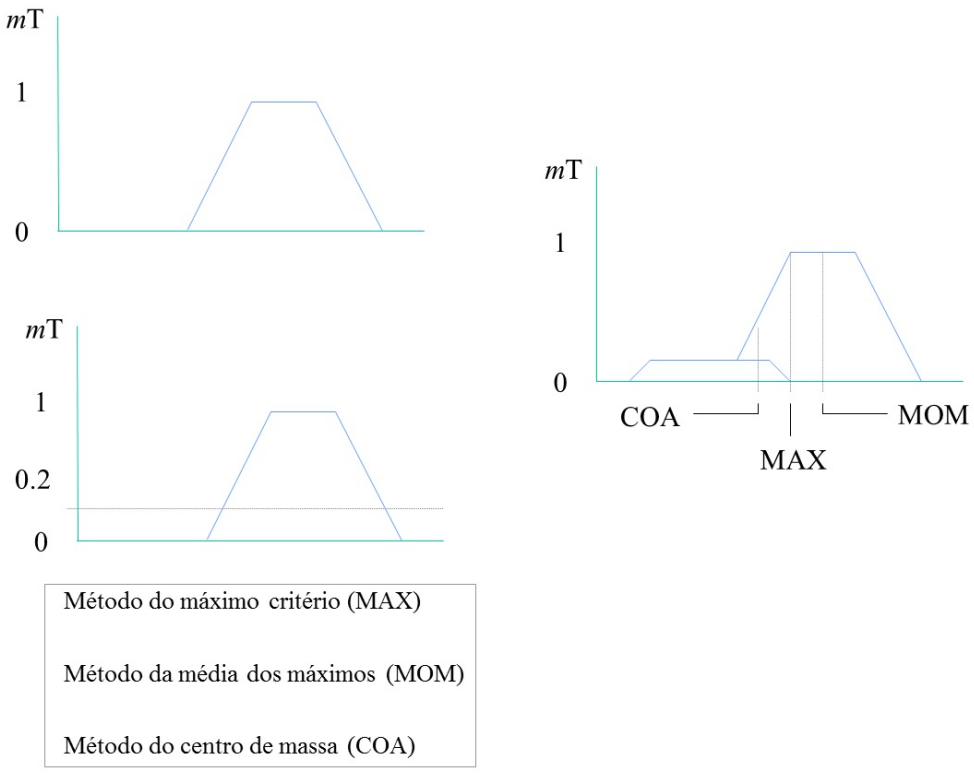

Figura 6. Métodos clássicos utilizados para desfuzzificação.

Capítulo 3

3 Sistema de suporte à decisão para adaptação de horários em transporte urbano utilizando lógica *fuzzy*

3.1 Metodologia

Com lógica difusa, cada viagem realizada em uma linha de ônibus é classificada em três pertinências: atrasada, sem atraso e adiantada, considerando a frequência dos desvios do horário e padrão de chegada em dias diferentes: frequência baixa, média e alta.

O horário de partida e chegada de uma viagem deve ser sempre armazenado para aplicar, quando necessário, uma função difusa que agrega valor considerando as características de atraso e adiantamento. Em seguida, há aplicação de regras de inferência para estas características, atribuindo pesos a cada uma delas, obtendo uma resposta difusa (*fuzzy*). Por fim, o processo de *defuzzificação* traduz esta resposta em uma unidade de interesse ao sistema, indicando se nas próximas viagens o ônibus circulará mais cedo, ou tarde, e o quanto o horário deve ser deslocado.

Para armazenamento de informações será utilizado um sistema de gerenciamento de banco de dados gratuito chamado Firebird [22]. A seleção desta ferramenta é devida a facilidade de utilização, possuindo uma interface poderosa e leve ao mesmo tempo, com mínima necessidade de configuração e administração.

Além disso, um simples servidor Firebird pode manipular múltiplas bases de dados independentes, cada uma com múltiplas conexões de clientes e utilizando linguagem SQL para consulta, manipulação e controle de dados.

No desenvolvimento do protótipo será utilizado o ambiente Delphi [23], auxiliando na criação da interface gráfica, ponto importante, visto que o sistema é de suporte à decisão. O Delphi é um ambiente de desenvolvimento de aplicações *Windows*, que possui interface gráfica amigável com o desenvolvedor que utiliza linguagem orientada a objeto baseada em Pascal.

O embasamento teórico será feito a partir de artigos, livros, revistas científicas e dissertações, validando com sistemas em utilização bem sucedidos com impacto significativo. O projeto do protótipo utilizará uma ferramenta de modelagem em UML (Linguagem de Modelagem Unificada) na sua edição gratuita intitulada *Astah Community Edition* [24] para documentação e referência, assim como uma ferramenta de modelagem de banco de dados relacional.

3.2 Estudo de casos de uso

Foi desenvolvida uma proposta de sistema de suporte à decisão para o gerenciamento de itinerários de transporte urbano baseada em lógica fuzzy. Os veículos possuem rotina semanal de circulação dividida em dias e, estes dias, em horários específicos, mas contratempos fazem com que estes horários não sejam seguidos à risca, sendo necessário adaptação do itinerário.

À medida que vão surgindo divergências, o sistema de horários precisa ser modificado. Destes contratempos podemos citar: engarrafamentos em trechos da viagem; repentino aumento do número de passageiros; crescimento desenfreado da venda de carros; mudança de estações de ano; fenômenos naturais imprevistos, etc. implicando em mais tempo parado em um trecho ou ponto de parada.

A adaptação do itinerário em uma determinada linha de transporte urbano não pode ser definitiva, visto que as causas de alterações de horário são passíveis de mudanças. Horários de pico podem variar de acordo com a época do ano, variação do número de passageiros e casualidades. Se a adaptação for dinâmica, um sistema de decisão subjetiva pode estar sujeito a frequentes falhas. O problema será administrado em níveis de urgência de atrasos e adiantamentos, e não simplesmente generalizando alterações de rotina. Um sistema de apoio à decisão de

mudança de horário baseado em lógica difusa auxiliará nas decisões usando cálculos com lógica *fuzzy*.

A Figura 7 possui a modelagem do sistema a partir do desenvolvimento do diagrama de casos de uso em UML e em seguida a descrição textual da funcionalidade principal do sistema. A Tabela 4 possui a descrição dos atores e seus papeis para execução e utilização do sistema para melhor entendimento.

As esferas informam os casos de uso, as setas apontadas para esferas determinam o ator que está realizando o caso e setas tracejadas definem herança entre casos de uso.

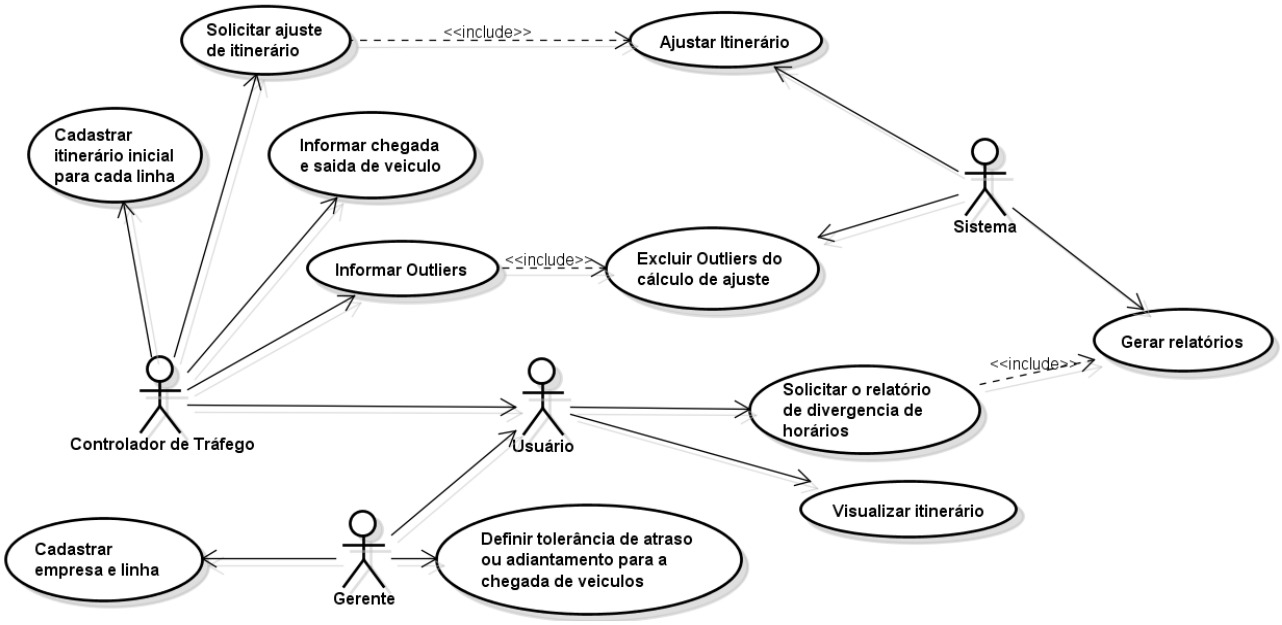

Figura 7. Diagrama de caso de uso do sistema.

Solicitar Ajuste de Itinerário

- **Breve Descrição**

Este caso de uso diz respeito à opção no sistema de ajuste de itnerário de viagens de uma linha de ônibus, caso o controlador de tráfego julgue necessário. Ao emitir a solicitação, caberá ao sistema sugerir um novo itinerário calculado a partir de horários anotados de chegada e saída de veículos.

- **Atores**

  Controlador de Tráfego
- **Pré-Condição: Autenticação do ator**
  a. O ator deve ter sido autenticado pelo sistema.
  b. Este caso de uso é iniciado quando o ator percebe que o itinerário. atual não está sendo mais eficiente.
- **Fluxo de Eventos: Fluxo Básico**

1. O ator acessa no menu a opção de suporte a decisão.
2. O sistema retorna tela para seleção de empresa e linha com horários sugeridos em branco.
3. O ator seleciona empresa e linha que dejesa ajustar itinerário.
4. O sistema retorna os valores de horários sugeridos atuais, quantidade atrasos e chegadas confirmadas para cada viagem desta linha.
5. O ator aciona o botão de ajuste de itinerário.
6. O sistema descarta os últimos horários definidos, substitui por novos horários e limpa a quantidade de atrasos e chegadas para nova contagem exibindo mensagem de confirmação.
7. O caso de uso é encerrado.

Tabela 4. Descrição do papel de atores presentes no sistema e no diagrama de casos de uso

Nome	Descrição
Ônibus	Agente direto nas iterações da solução. Influencia e é influenciado.
Usuário	Acompanha os relatórios do sistema, atualiza horários sugeridos e faz cadastros.
Gerente	Extensão de usuário com privilégios para funcionalidades mais críticas, passíveis de alteração de acordo com o perfil da empresa.
Controlador de tráfego	Tipo de usuário para cadastros básicos, entradas periódicas de horários dos veículos e solicitante de ajuste de horários quando necessário.
Sistema	Agente de suporte a decisão relatando horários, relatórios e cálculos realizados.

O gerenciamento se dará da seguinte maneira:

1- Em horários que se costuma ter atraso no horário de chegada, os veículos começam a circular mais cedo que o horário de partida padrão, com intuito de compensar o horário que será perdido posteriormente. Quando há adiantamento na chegada, o problema é contornado com atraso da saída.
2- O gestor possuirá um monitor de atrasos para ter a base de atualização de horários de uma linha determinada, com a quantidade de atrasos e a quantidade total de viagens que foram tomados nota.
3- Utilizando lógica difusa, cada viagem realizada em uma linha de ônibus é classificada em três pertinências: atrasada, sem atraso e adiantada. Também é considerada a frequência dos desvios do horário padrão de chegada: frequência baixa, média e alta, definindo o padrão dinâmico de tempo que será compensado quando houver atualização dos horários sugeridos pelo sistema.
4- O horário de partida e chegada de uma viagem deve ser tomado em nota pelo gestor no momento de chegada de um veículo, ou deve ser realizada de forma automática para evitar possíveis adulterações.
5- A atualização da jornada pode ser realizada sempre quando necessário.
6- É aplicada uma função difusa que *fuzzifica* este valor considerando as características de atrasado, sem atraso e adiantado da viagem. Em seguida, são aplicadas as regras de inferência para estas características atribuindo pesos a cada uma delas, calculando-se a partir de uma função *fuzzy* mais aplicável ao caso. Por fim, o processo de *defuzzificação* traduz esta resposta em uma unidade de interesse ao sistema, indicando se as viagens começarão a circular mais cedo ou mais tarde e o quanto o horário deve ser deslocado.
7- Considerando que alguns empecilhos são situações de exceção raras e que sua importância não deve ser tomada em nota, com o intuito de não alterar significantemente o itinerário quando atualizado, o programa possui um botão para o gestor definir *outliers* para a viagem específica não ser tomada nota. Exceções como: quebra de ônibus, troca de linha por consequência de defeito para compensação de outras linhas mais importantes, etc.

8- O cadastro inicial é definido pelo horário atual utilizado para ter como base de adaptação algo preexistente.
9- É necessário registro de empresas, linhas e horários, onde as linhas são associadas às empresas e os horários associados às linhas. É possível a existência de linhas com descrição duplicada, mas nunca empresas com a mesma descrição para não haver violação de chave primária no banco de dados.
10- O gestor deve selecionar a empresa e linha para poder consultar o horário de saída dos veículos.
11- O gestor deve definir tolerância de atraso ou adiantamento que um veículo pode atingir.

O sistema define a atualização entre dois pontos para o protótipo, porém para o funcionamento do ajuste em tempo real a tomada de tempo pode ser realizada entre múltiplos pontos, dando a possibilidade de ajuste de itinerários mais dinamicamente. Esta implementação foi desconsiderada no sistema para evitar o aumento de custo na implantação.

Na tabela 5 há o resumo da descrição de necessidades funcionais e não funcionais do sistema:

Tabela 5. Descrição de necessidades do sistema.

	Descrição	F/NF
1	Registrar empresas	F
2	Registrar linhas	F
3	Registrar horários	F
4	Consultar horário de empresa e linha específica	F
5	Alterar tolerância	F
6	Consultar atrasos e a quantidade total de viagens de uma determinada linha	F
7	Registrar tipos de dias (dias da semana com horários similares - Ex.: Feriado, segunda, fim de semana)	F
8	Atualizar de horários sugeridos (jornadas)	F
9	Tomar nota sobre o momento de chegada e saída de um veículo	F

10	Ignorar informações em não se deve tomar nota (*Outliers*)	F
11	O cadastro inicial de jornadas deve ser definido pelo horário atual utilizado	NF
12	Empresas não podem possuir a mesma descrição	NF
13	Uso de lógica *fuzzy*	NF
14	Atualizar horários sugeridos	F

3.3 Banco de dados

Para facilitar a interação e manipulação com a informação, foi selecionado o *Firebird* como banco de dados para construção do protótipo, e, para implementação e administração, foi utilizado o *software Flamerobin* [25].

Firebird é um banco de dados relacional que oferece muitas características com padrão ANSI SQL, que funciona em *Linux*, *Windows* e uma variedade de plataformas *Unix*. *Firebird* oferece excelente concorrência, de alto desempenho, suporte a procedimentos armazenados (*Stored procedures*) e gatilhos (*Triggers*), tendo sido utilizado em sistemas de produção, ainda ativos, por aproximadamente um milhão de desenvolvedores em todo o mundo.

Firebird é um software livre implementado em linguagem C e C++ por programadores e consultores técnicos com intúito de desenvolver uma ferramenta de banco de dados robusta, multiplataforma, de foco acadêmico e de código livre, liberado pela empresa Inprise Corp em julho de 2000, atual *Borland Software Corporation*. Foi derivado do banco de dados Interbase, agregado pela *Borland* em 1991 em seus produtos, sendo concorrente direto do *Microsoft Access* na época, persistindo no mercado por ser leve e dar suporte a bancos de dados maiores que dois *gibabytes*.

Como ferramenta de apoio à implementação e administração foi selecionada uma GUI (Interface gráfica para o usuário) chamada Flamerobin, ferramenta de manipulação/administração, na Figura 8 demonstrando a interface para manipulação de dados e tabelas, para bancos de dados Firebird. Ele é projetado para ser pequeno, simples, multiplataforma, e, ainda, oferecer todos os recursos básicos necessários para administrar servidores Firebird na criação e manutenção de bancos de dados. Flamerobin é um *software opensource*, possuindo seu código fonte disponível no site oficial para *Windows*, *Unix*, *Linux* e *MacOS X*.

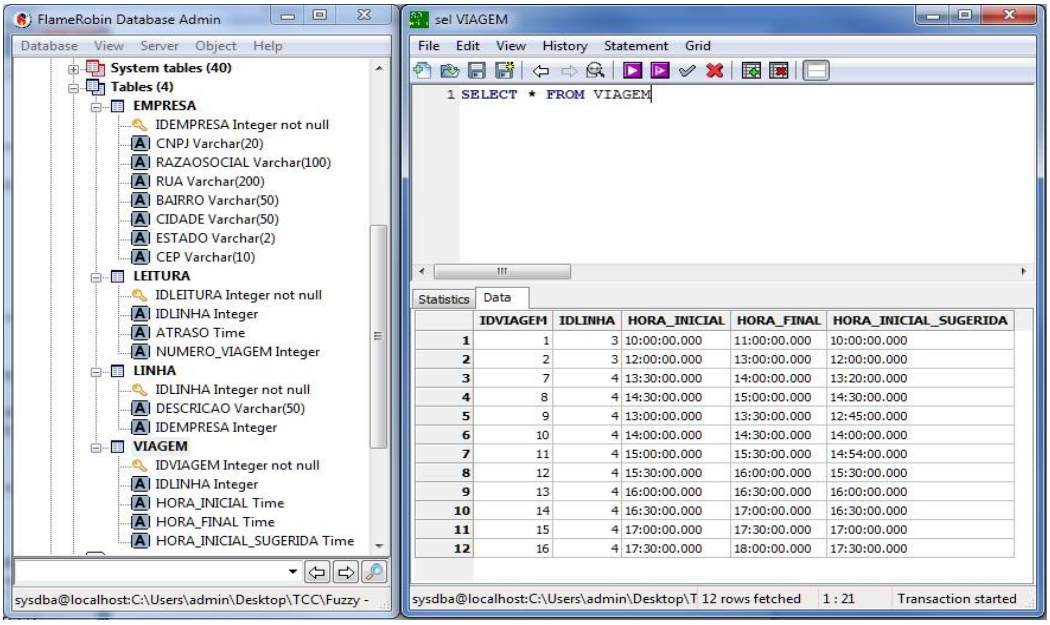

Figura 8. Software Flamerobin para manipulação de dados e administração de tabelas em bancos de dados Firebird.

O protótipo do sistema foi concebido com quatro tabelas para manipular os dados das funções básicas do sistema de acordo com os casos de uso pela modelagem da Figura 9. Duas das tabelas são utilizadas para cadastros de empresa e linha, uma tabela para cadastro e atualização do itinerário de cada linha e outra tabela para leitura dos atrasos e adiantamentos de cada viagem, se houver relevância de guardar a informação.

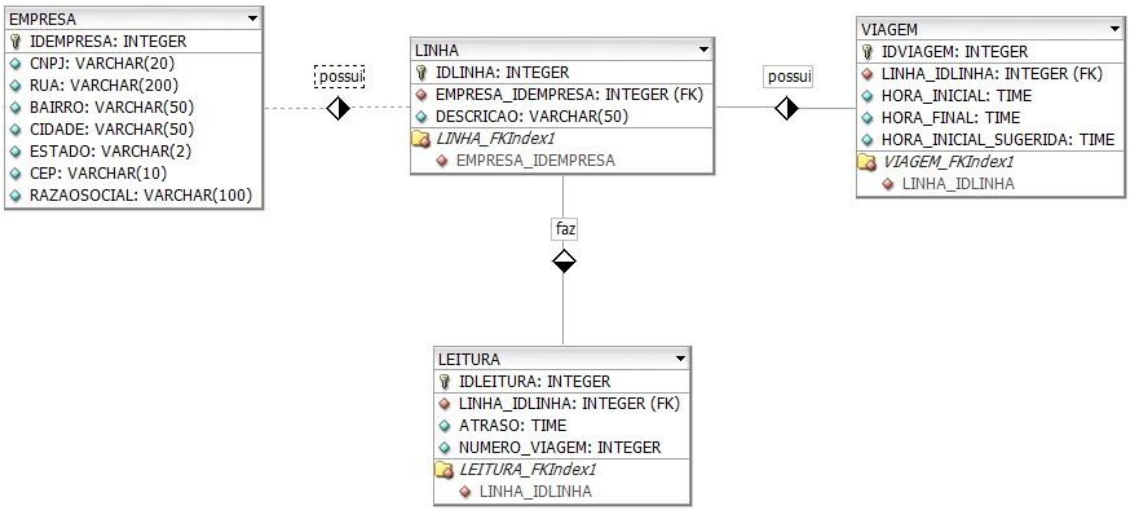

Figura 9. Modelagem de banco de dados do protótipo.

A tabela de EMPRESA contém, além do campo autoincremento para realizar chave estrangeira com a tabela de LINHA, sete campos para descrição das empresas, sendo a tabela de cadastro inicial para utilização do sistema. Na prática seria o cadastro de todas as empresas que participam de um terminal ou ponto de espera, onde o usuário de controle de tráfego está presente.

A tabela de LINHA possui chave estrangeira com todas as outras, realizando interligações da informação, além do campo descritivo. A comunicação com a tabela de EMPRESA define que empresas pertencem as linhas cadastradas. A ligação com a tabela VIAGEM define a rotina de uma determinada linha, construindo seu itinerário. A conexão com a tabela LEITURA guarda as informações das viagens tendo os valores da tabela de VIAGEM como referência.

A tabela VIAGEM define os horários das viagens de cada linha, possuindo campos para armazenar a hora inicial e estática definida pela empresa sem o sistema. A hora final de cada viagem é necessária para identificar o tempo de uma viagem em relação ao horário sugerido após utilização do sistema. Esta tabela define a principal tela do sistema, descrito com mais detalhes neste capítulo na seção sobre a interface com o usuário, dando suporte a decisão no horário de saída dos veículos.

A tabela de LEITURA guarda todas as informações relevantes para utilização do cálculo *fuzzy* com intuito de sugerir ao usuário como o horário deve ser alterado. Para isso possui três campos: um campo para identificar qual linha está sendo entrada de informação, um campo para guardar quanto tempo houve de atraso e qual o número sequencial da viagem no dia.

3.4 Interface com o usuário

Para sistemas de suporte à decisão, a interface com o usuário possui extrema importância por interagir diretamente com o usuário para definição de metas. O protótipo possui quatro telas principais, três de cadastros e uma para interagir com o controle de tráfego.

As telas de cadastro correspondem às tabelas de EMPREA, LINHA e VIAGENS. Possuindo um padrão, vide Figuras 10 e 11, para consulta de registros cadastrados e botões de inserção, alteração e exclusão. Possuindo campos relevantes para consulta se houver alimentação do banco de dados do sistema com várias empresas. Além disso, todos os cadastros possuem um *grid* para visualização dos campos cadastrados previamente.

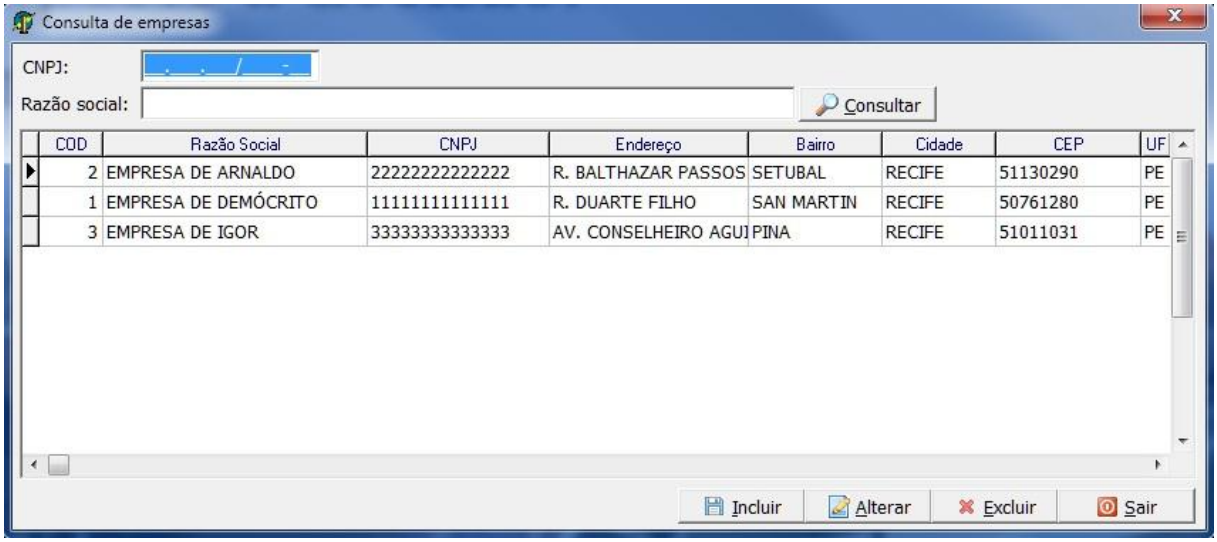

Figura 10. Interface padrão de consulta para cadastros.

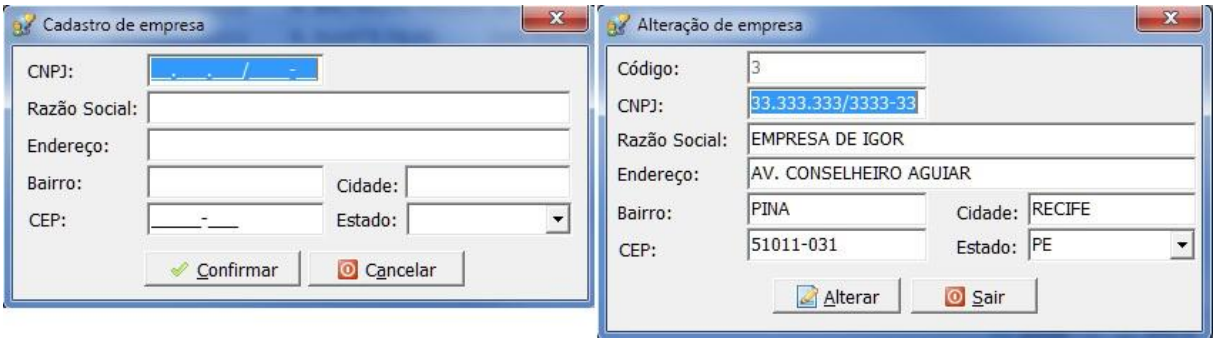

Figura 11. Interface padrão de cadastro e alteração.

Para o cadastro de viagens, há uma função diferencial no padrão na consulta (vide Figura 12) para adicionar todo o itinerário de uma determinada linha, definindo o horário de partida do ponto e o horário de chegada nele, ou em outro ponto, caso o sistema possua terminal auxiliar de parada. Para este sistema foi considerado que a viagem é cíclica, fazendo com que o veículo possua o mesmo ponto de saída e chegada, porém, se houver a necessidade de múltiplos pontos, seria impreterível a adição de comunicação entre eles.

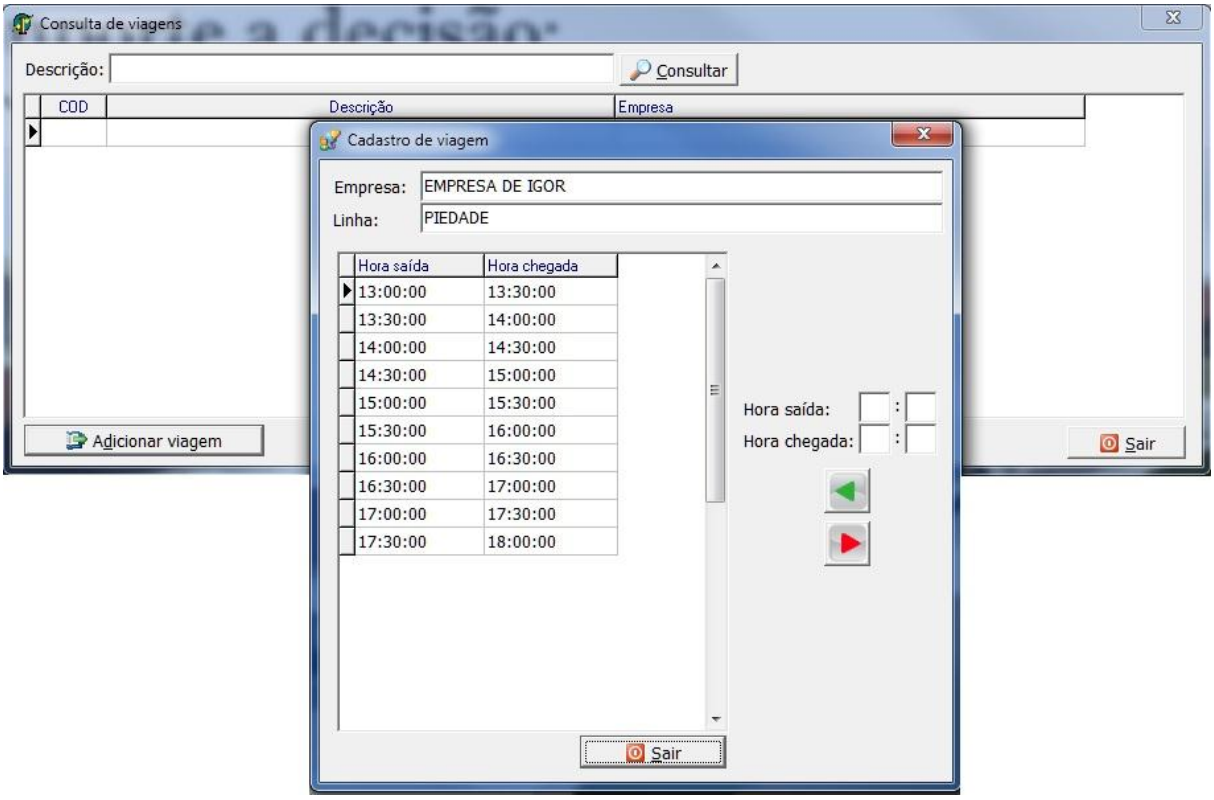

Figura 12. Interface de cadastro de viagem.

O consulta de viagens é aberta com a informação de cada linha já cadastrada, sendo necessária a seleção de uma linha para adicionar e remover horários de saída e chegada dos pontos. Na adição de horários o usuário informa a hora e minuto e utiliza o botão com seta verde para gravação do registro, e, para remoção de horário, é necessário selecionar no *grid* e em seguida selecionar o botão com seta vermelha.

Para finalizar a descrição da interface, a Figura 13 representa a função mais importante do sistema: a tela de suporte à decisão de horários das linhas cadastradas. A tela possui dois *comboboxes* para seleção da linha que será utilizada para receber informação de suporte. O *combobox* de empresa, automaticamente, é alimentado pelo cadastro da tabela EMPRESA com a razão social de cada registro, e, ao ser selecionado, preenche o *combobox* abaixo com todas as linhas da empresa que foi escolhida.

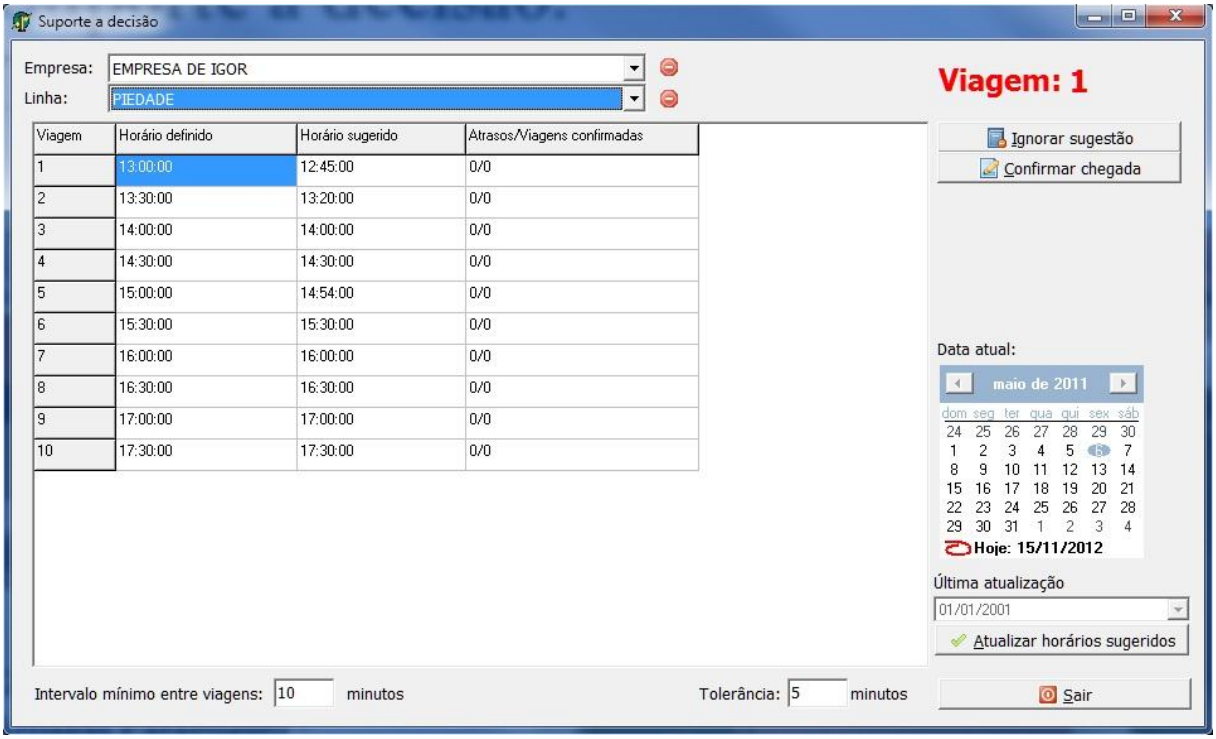

Figura 13. Interface de suporte a decisão.

O *grid layout* informa o número da viagem da linha selecionada, qual o horário que foi definido inicialmente como referência, o horário de saída do ponto sugerido pelo sistema após atualização e o quantitativo de atrasos e viagens com chegadas confirmadas. Com o quantitativo de atrasos, o controlador de tráfego pode estipular se o itinerário necessita de adaptação. Se houver diversos atrasos por viagem confirmada em diversas viagens diferentes, é essencial que ocorra atualização dos horários, porém, se apenas uma viagem estiver com atraso mais frequente, fica sempre a critério do usuário. Como o sistema é de suporte a decisão, pode se adaptar a qualquer momento, sem perda, porém não é realizado automaticamente, deixando esta função para trabalhos futuros, por fugir do escopo do estudo.

O botão de ignorar sugestão foi adicionado para o sistema não registrar horários de chegadas que não são relevantes ao cálculo, exceções como: veículo quebrado, acidentes, etc., sendo uma função essencial para o controle de tráfego informar ao sistema que uma determinada viagem é um *outlier*, ignorando assim está viagem no ajuste de horário ao atualizar.

O botão de atualizar horários sugeridos é utilizado para atualização dos horários de saída de um ponto determinado, usando a função de cálculo com *fuzzy,*

tendo como referência os atrasos de cada viagem. Após inferência de um novo horário, a coluna de atrasos/viagens confirmadas, que consta no *grid*, é zerada para dar inicio a nova contagem, estabelecendo uma nova referência a partir da ultima data de atualização. A adaptação mais recente do itinerário consta acima do botão para complementar a informação.

O botão de confirmar chegada determina o horário que o veículo finalizou a viagem a partir de uma nova tela do protótipo (vide Figura 14), que na prática poderia ser realizada de forma automática se os veículos possuíssem GPS (Sistema de Posicionamento Global) para localização de chegada ao ponto de parada, partindo do princípio que o pior caso: orçamento limitado e ambiente caótico, a interface de entrada de dados do protótipo atenderia a funcionalidade. Observe que o veículo não precisou ser informado neste caso, pois a utilidade do *software* independe de que ônibus esteja circulando. A informação necessária será se a viagem foi realizada ou não, que preencha o horário e atenda os requisitos do itinerário.

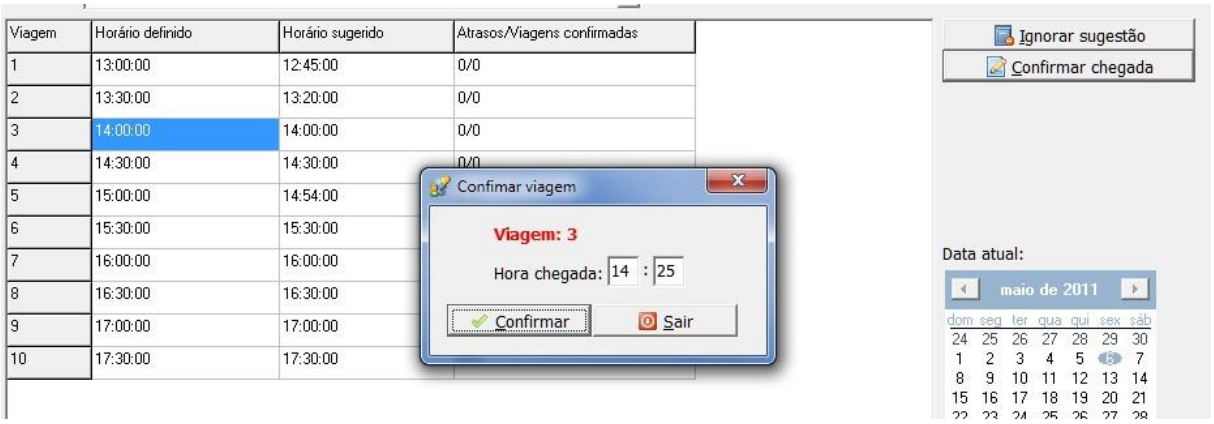

Figura 14. Interface para confirmação viagem concluída.

Por fim há dois atributos auxiliares de entrada para definição de atrasos e o menor tempo entre viagens diferentes. A tolerância é utilizada para evitar guardar dados de atraso para cálculo *fuzzy* que não sejam significativos e para evitar o monitoramento errado do quantitativo de atrasos por viagens confirmadas pelo controle de tráfego. Este tempo tolerante é identificado a partir do horário de chegada cadastrado na tabela VIAGEM adicionando e subtraindo o tempo de tolerância estipulado para obter um intervalo aceitável de atraso ou adiantamento, identificando pertinência pelo horário de chegada pela viagem da linha.

O intervalo mínimo entre viagens corresponde ao máximo que pode ser adaptado um horário levando em consideração o horário da próxima viagem, para evitar a saída de dois veículos simultaneamente. Porém como as empresas possuem raramente veículos sobrando parados para circular em um ponto, se este intervalo mínimo for atingindo em viagens em sequência constatasse que, em um determinado horário, há a necessidade de mais veículos em circulação para atender a demanda com o tráfego atual.

3.5 Fuzzificação

Anteriomente ao processo de *fuzzificação*, o sistema pesquisa no banco de dados a quantidade de viagens cadastradas na tabela de LEITURA, de acordo com a viagem atual, determinada pelo campo NUMERO_VIAGEM. Em seguida há a abertura da consulta dos atrasos para determinar a pertinência dentro do conjunto difuso.

Como há relevância na frequência de atrasos e o tempo atraso médio, os dois valores foram fuzzificados. A média foi utilizada no tempo de atraso para abstrair o peso de uma viagem em específico com muito ou pouco atraso.

Para atribuir os valores do mundo real para conjuntos *fuzzy,* foi utilizado como referência o valor máximo da frequência de atrasos de uma viagem em específico, em vários dias diferentes, ou seja: valor máximo determinado é a quantidade de viagens registradas diferentes. Para a referência de tempo atraso foi utilizado o dobro da média de atrasos.

As frequências e tempos atrasos foram *fuzzificadas* em três níveis: baixa, média e alta. Todas definidas em função das quantidades de viagens registradas diferentes e dividida em três partes para inferir seu subconjunto dentro do conjunto de atrasado.

Em seguida todos esses valores são armazenados em um vetor dinâmico. A Figura 15 demonstra por fluxograma a preparação e implementação para *fuzzificação* destes valores e respectivo código fonte se encontra no Apendice A.

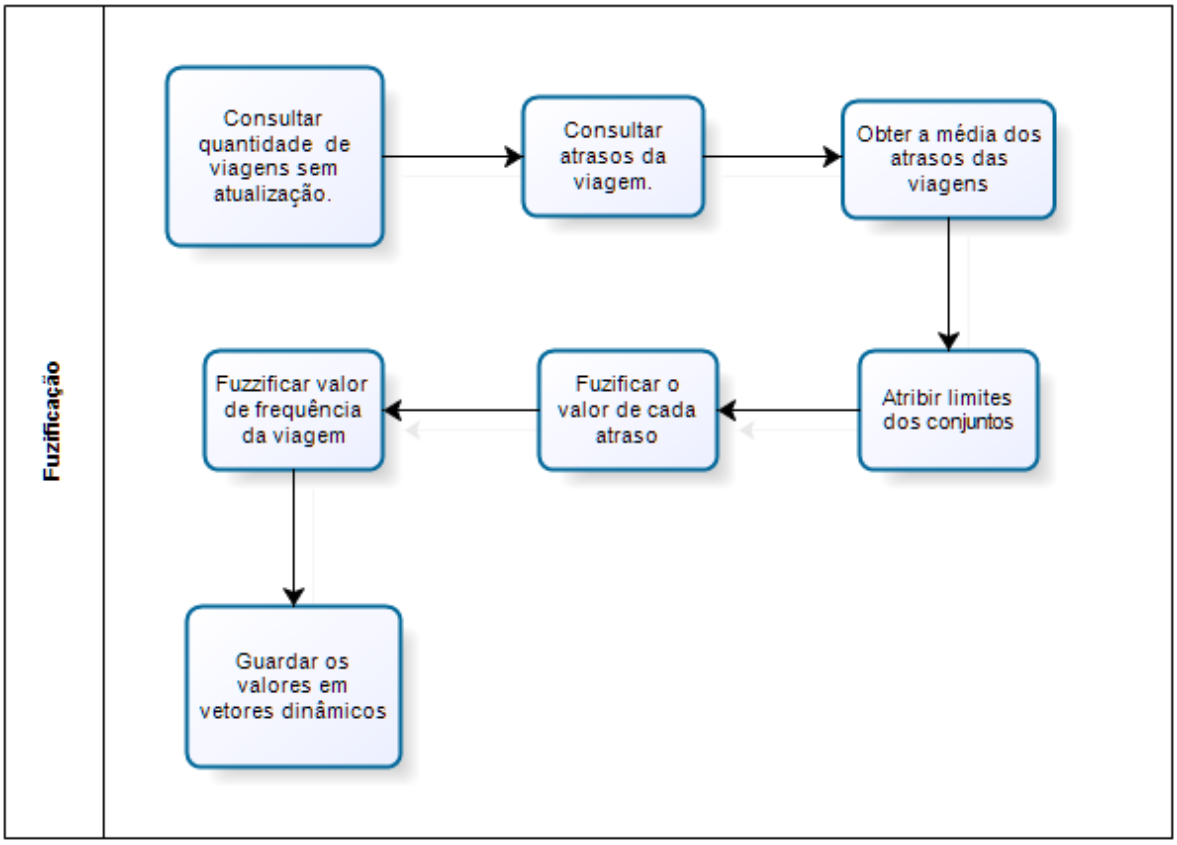

Figura 15. Fluxograma da preparação e fuzzificação de valores.

3.6 Regras de inferência (regra base)

Mantendo o padrão de três níveis para formação da regra base, o produto final da tomada de decisão do protótipo define como ajuste de tempo: baixo, médio e alto, levando em consideração os dois conjuntos *fuzzy* que foram determinados durante a *fuzzificação* e os pesos pela inferência de máximos de Mandami. Os valores difusos são combinados entre atrasos e frequências de acordo com a Tabela 6 e implementação no Apendice A.

Tabela 6. Conjunto de regras de inferência do sistema.

Regra	Característica A (Atraso)	Característica B (Frequência)	Classe (Ajuste de tempo)
1:	baixo	baixa	baixo
2:	baixo	média	baixo
3:	médio	baixa	baixo
4:	médio	média	médio
5:	médio	alta	médio
6:	alto	média	médio
7:	alto	alta	alto

Os valores estão definidos como 1, 2 e 3 (respectivamente de baixo, médio e alto) de acordo com o padrão de *fuzzificação* presente nos códigos das implementações.

3.7 Defuzzificação

Para defuzzificar os valores foi utilizada a média dos máximos para se obter o valor final para ajuste do itinerário. Para isso, foram utilizados os pesos definidos com as regras de inferência e o tempo médio de atraso para adaptação de horários de cada viagem, de acordo com fluxograma na Figura 16 e implementação no Apendice A.

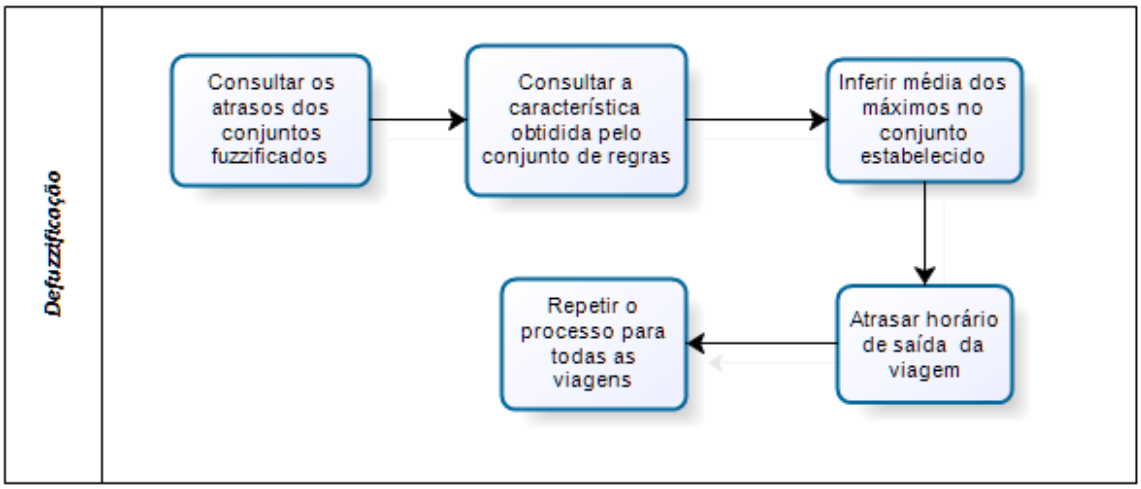

Figura 16. Implementação da defuzzificação de valores.

3.8 Resultados e impactos esperados

O sistema propõe a melhoria do gerenciamento dos itinerários das linhas de transporte em grandes centros urbanos, consequentemente, focando no bem estar da maior parte da população.

Com a melhoria do serviço prestado pelas empresas de transporte urbano, o deslocamento da população em horários de fluxo será contornado de forma menos caótica.

O funcionamento do sistema deverá forçar a modificação do padrão de definição de horários pelas agências reguladoras, consentindo que a adaptação é fator importante para o deslocamento da população, assim como a transparência na definição de horários e facilidade de acesso a horários modificados.

3.9 Limitações

Conjuntos *fuzzy* são ferramentas valiosas para se transferir o pensamento de engenheiros do mundo real para sistemas complexos computáveis, porém, apesar do foco do trabalho ser em inteligência artificial, o conceito de lógica *fuzzy* não se aplica a este escopo, mas na utilização da cibernética, havendo retroalimentação da informação processada pelo usuário, quando é necessária atualização de dados para o funcionamento do sistema.

Por necessitar de um especialista na área, neste caso um especialista em tráfego, o sistema supõe que a adaptação deve ocorrer e remete a função de inferência de Mandami (Mínimo) para geração da regra base, ou seja, desta forma a atualização do itinerário é nivelada sempre por baixo para não ocorrer extrapolação no ajuste.

Segundo Ying, os sistemas *fuzzy* são construídos normalmente na prática em vez de projetados teoricamente. Partindo desta premissa, a validação desta inferência poderia ser modificada para função mais adequada, mas sem a necessidade de grandes mudanças na implementação ou no conceito.

Capítulo 4

4 Conclusão

No oriente, a cultura fez com que os conceitos da lógica nebulosa fossem aceitos com maior facilidade do que no mundo ocidental, investindo em soluções baseadas em modelagem e controle *fuzzy*. Inúmeras aplicações surgiram, principalmente no Japão, onde algumas aplicações se tornaram exemplos clássicos.

Este trabalho tem o intuito de intensificar o desenvolvimento e utilização de sistemas inteligentes para solucionar problemas do cotidiano, onde a adaptação com o meio não é levada em consideração, causando retrabalho no desenvolvimento para se adequar a realidade que está em constante mudança.

A mudança de paradigma de sistemas estáticos favorece a evolução natural de ambientes em desenvolvimento. Com a alteração da arquitetura do sistema de controle de horários em função da economia local, focada na venda de carros e aumento desenfreado da população, usuários frequentes de transporte urbano poderão usufruir de um serviço focado na melhoria da qualidade de vida pelo tempo ganho na execução de outras atividades.

Os sistemas inteligentes possuem um maior tempo de desenvolvimento, devido ao estudo mais aprofundado do problema e custo adicional da análise de especialistas na área, possuindo a médio/longo prazo a economia no retrabalho devido à atualização dinâmica de modificações no núcleo.

4.1 Trabalhos futuros

O principal problema na criação de sistemas *fuzzy* é de lidar com diversas entradas possíveis e obter um estudo especial para inferir possíveis consequências a partir do conjunto de regras que foi formado. Sistemas com neurônios são mais recomendados quando há maior quantidade de informação a ser processada.

Como este trabalho possui ênfase em adaptação e sistemas inteligentes, um trabalho futuro seria de obter as funções de inferência de forma automática. Para isso a melhoria com sistemas neurais aprimoraria a lógica na geração de funções de transferência de acordo com o necessário, ou ainda mantendo a decisão do usuário para informar quando se deve ocorrer atualização de horário a partir de referências.

Essa adição iria aperfeiçoar a lógica de inferência, mas o desempenho durante a classificação faria que o sistema *neurofuzzy* levasse mais tempo para ser executado. Partindo do princípio que uma rede neural deve ser inicializada de forma randômica e treinada para obter funções otimizadas, esta seria a diferença principal em relação ao sistema desenvolvido apenas em *fuzzy*. Para melhoria do processo Sugeno sugere um algoritmo adaptativo para geração de regras base [25] similar a Madami, que pode ser dividido em cinco passos:

1. Projetar um sistema de inferência apropriado para classificação do problema.
2. Aperfeiçoar a função de inferência de acordo com a entrada de dados.
3. Estabelecer parâmetros de treinamento e validação.
4. Execução do algoritmo.
5. Testar os resultados usando os dados de teste.

Bibliografia

[1] KORNER, S. **Laws of thought. Encyclopedia of philosophy**, vol. 4, MacMillan, NY: 1967, p. 414-417.

[2] YEN, J., LANGARI, R. e ZADEH, L. A. **Industrial applications of fuzzy logic and inteligent systems**. IEEE Press, 1995.

[3] YING, H. **Fuzzy control and modeling: Analytical foundations and applications**. IEEE Press, 2000.

[4] ZADEH, L. A. **Making computers think like people**. IEEE, Spectrum, 8/1984, p. 26-32.

[5] ZADEH, L. A. **Fuzzy sets, information and control**, vol. 8, p. 338-353.

[6] AGÊNCIA BRASIL, http://agenciabrasil.ebc.com.br/noticia/2011-05-04/estudo-do-ipea-mostra-que-65-da-populacao-usam-transporte-publico-nas-capitais, 2011, acessado em jullho de 2012

[7] LEE, C. C. **Fuzzy logic in control systems: fuzzy logic controller**, part I e II, IEEE Transactions on systems, Man and cybernetics, 1990, p. 408-435.

[8] KOSKO, B. **Neural networks and fuzzy systems: A dynamical systems approach to machine Intelligence**, Prentice-Hall, Englewood Cliffs, 1991.

[9] KOSKO, B. e ISAKA, S. **Fuzzy logic**, 1993, p. 78-79

[10] SHAPIRO S. **Vagueness in context**, Oxford University Press, 2006.

[11] SMITH N. J. **Vagueness and truth degrees**, Oxford University Press, 2008.

[12] CHEN, G. e YING, H. **BIBO stability of nonlinear fuzzy PI control systems, Journal of intelligent and fuzzy systems**, 5, 1997, p. 245-256.

[13] YING, H. **General analytical structure of typical fuzzy controllers and their limited structure theorems**, Automatica, 29, 1993, p. 1139-1143.

[14] ZADEH, L. A. **Fuzzy sets information and control**, 1965.

[15] BECCHI, A. **Logic and determinism in Jan Lukasiewicz's philosophy**, 2002.

[16] SÁNCHEZ, G. e TORRES, C. **A Mandani-type fuzzy inference system to automatically assess Dijkstra's algorithm simulation**, International Journal "Information Theories and Applications" **,** 2010, vol. 17, n.1, p. 38-47.

[17] BEZDCK, J. C., **A review of probabilistic. Fuzzy and neural models for pattern recognition**, vol.1, 1993, p. 1-25.

[18] ZADEH, L. A. **Fuzzy sets information and control**, 1965.

[19] CHEN, G. e YING, H., **BIBO stability of nonlinear fuzzy PI control systems, Journal of intelligent and fuzzy systems**, 5, 1997, p. 245-256.

[20] CASTELLANO, G., FANELLI, A. e MENCAR, C. **Design of Transparent Mamdani Fuzzy**, 2003.

[21] T. TAKAGI e M. SUGENO, **Fuzzy identification of systems and its applications to modeling and control**, IEEE transactions on systems, Man and cybernetics, vol. 15, n. 1, pp. 116-132, 1985.

[22] Firebird, http://www.firebird.org/en/about-firebird, acessado em outubro de 2012.

[23] Embarcadeiro, http://www.embarcadero.com/br/products/delphi, acessado em dezembro de 2012.

[24] Astah, http://astah.net/editions/community, acessado em dezembro de 2012.

[25] Flamerobin, http://www.flamerobin.org, acessado em dezembro de 2012.

[26] JANG, S. R., **ANFIS: Adaptive network based fuzzy inference systems**. IEEE Transactions on systems, man and cybernetics, 1993, vol. 23, n. 3, p. 665-685.

Apêndice A

Código fonte

1. *Fuzificação*

```
//consulta quantidade de viagens na linha e viagem selecionada (sem atualização)
qntViagensConfirmadas := quantidadeViagensConfirmadas(viagemAtual);

//Consulta todos os atrasos da linha e viagem na query1
consultaQueryViagens(viagemAtual);

quantidadeSomaMinutos := 0;
media := 0;
somaAtrasos := StrToTime('00:00:00');

//Soma todos atrasos da linha e viagem em segundo e tira média
while not DataModule2.IBQuery1.Eof  do
begin
  atraso := StrToTime(DataModule2.IBQuery1.FieldByName('atraso').AsString);
  DecodeTime(atraso, Hour, Min, Sec, MSec);
  quantidadeSomaMinutos := quantidadeSomaMinutos + (hour*60) + min;
  DataModule2.IBQuery1.Next;
end;

// Checa se haverá viagens
if (qntViagensConfirmadas = '0') or (quantidadeViagensAtrasadas(viagemAtual) = '0') then
begin
  viagemAtual := viagemAtual + 1;
  Continue;
end
else
begin
  media := quantidadeSomaMinutos / StrToInt(qntViagensConfirmadas);
end;
//Atribui os limites dos conjuntos fuzzy para frequencia (baixo, médio, alto)
limiteFreqAlta  := StrToInt(qntViagensConfirmadas);
tamanhoConjuntoFuzzy := limiteFreqAlta / 3;
limiteFreqBaixa := tamanhoConjuntoFuzzy;
limiteFreqMedia := tamanhoConjuntoFuzzy * 2;

freqAtual := StrToInt(quantidadeViagensAtrasadas(viagemAtual)) / limiteFreqAlta;

//Atribui valor fuzzy
if (freqAtual <= limiteFreqBaixa) then
begin
  FreqAtualfuzzy := 1; // 'baixa';
  freqIntervalo := freqAtual / StrToInt(qntViagensConfirmadas);
end
else if ((freqAtual > limiteFreqBaixa) and (freqAtual <= limiteFreqMedia)) then
begin
  FreqAtualfuzzy := 2; //'média';
  freqIntervalo := freqAtual / 2 * (StrToInt(qntViagensConfirmadas));
end
else
begin
  FreqAtualfuzzy := 3; //'alta';
  freqIntervalo := freqAtual / 3 * (StrToInt(qntViagensConfirmadas));
end;

//Atribui os limites dos conjuntos fuzzy para atraso (baixo, médio, alto)
limiteAtrasoAlto  := 2 * media;
tamanhoConjuntoFuzzy := limiteAtrasoAlto / 3;;
limiteAtrasoBaixo := tamanhoConjuntoFuzzy;
limiteAtrasoMedio := tamanhoConjuntoFuzzy * 2;
```

```
//Atribui tamanho do vetor dinâmico pela quantidade viagens que atrasaram
SetLength(vetorAtraso,StrToInt(quantidadeViagensAtrasadas(viagemAtual)));
SetLength(vetorAtrasoIntervalo,StrToInt(quantidadeViagensAtrasadas(viagemAtual)));
SetLength(inferencia,StrToInt(quantidadeViagensAtrasadas(viagemAtual)));
SetLength(viagemAtraso,StrToInt(quantidadeViagensAtrasadas(viagemAtual)));
SetLength(inferenciaMaximo,StrToInt(quantidadeViagensAtrasadas(viagemAtual)));

//Abre query com todas as viagens que atrasaram
consultaQueryViagensAtrasadas(viagemAtual);
//Para cada valor atrasado
x := 0;
while not DataModule2.IBQuery1.Eof  do
begin
  //Atribui valor fuzzy
  atraso := StrToTime(DataModule2.IBQuery1.FieldByName('atraso').AsString);
  DecodeTime(atraso, Hour, Min, Sec, MSec);
  quantidadeSomaMinutos := 0;
  quantidadeSomaMinutos := quantidadeSomaMinutos + (hour*60) + min;

  if (atraso <= limiteAtrasoBaixo) then
  begin
    vetorAtraso[x] := 1; //baixa
    vetorAtrasoIntervalo[x] :=  quantidadeSomaMinutos / (media);
  end
  else if ((freqAtual > limiteAtrasoBaixo) and (freqAtual <= limiteAtrasoMedio)) then
  begin
    vetorAtraso[x] := 2; //média
    vetorAtrasoIntervalo[x] :=  quantidadeSomaMinutos / (2 * media);
  end
  else
    vetorAtraso[x] := 3; //alta
    vetorAtrasoIntervalo[x] :=  quantidadeSomaMinutos / (3 * media);

  if vetorAtrasoIntervalo[x] > 1 then
  begin
    vetorAtrasoIntervalo[x] := 1;
  end;

  viagemAtraso[x] := DataModule2.IBQuery1.FieldByName('NUMERO_VIAGEM').AsInteger;

  x := x + 1;
  DataModule2.IBQuery1.Next;
end;
```

2. Regras de inferência

```
//Regra de inferência (Adiante pouco, médio e muito)
x := 0;
while x <= Length(vetorAtraso) - 1 do
begin
  //Atrasos e Frequencias
  if (vetorAtraso[x] = 1) and (FreqAtualfuzzy = 1) then
  begin
    inferencia[x] := 1;
  end
  else if ((vetorAtraso[x] = 1) and (FreqAtualfuzzy = 2)) or  ((vetorAtraso[x] = 2) and (FreqAtualfuzzy = 1)) then
  begin
    inferencia[x] := 1;
  end
  else if (vetorAtraso[x] = 2) and (FreqAtualfuzzy = 2) then
  begin
    inferencia[x] := 2;
  end
  else if ((vetorAtraso[x] = 2) and (FreqAtualfuzzy = 3)) or ((vetorAtraso[x] = 3) and (FreqAtualfuzzy = 2)) then
  begin
    inferencia[x] :=  2;
  end
  else if (vetorAtraso[x] = 3) and (FreqAtualfuzzy = 3) then
  begin
    inferencia[x] := 3;
  end
  else if ((vetorAtraso[x] = 3) and (FreqAtualfuzzy = 1)) or ((vetorAtraso[x] = 1) and (FreqAtualfuzzy = 3))then
  begin
    inferencia[x] := 2;
  end;

  inferenciaMaximo[x] := maximo(vetorAtrasoIntervalo[x],freqIntervalo);
```

3. Defuzzificação

```
// Defuzzificando utilizando média dos máximos
viagemAtual := 1;
contAtrasos := 1; // Contador de atrasos
while  viagemAtual <= 10 do
begin
  mediaTotal := 0;
  count := StrToInt(quantidadeViagensAtrasadas(viagemAtual));
  qntViagensAtrasadas := StrToInt(quantidadeViagensAtrasadas(viagemAtual));
  if count <> 0 then
  begin
    mediaTotal := 0;
    y := 0;
    while y < qntViagensAtrasadas do
    begin
      if (vetorAtraso[contAtrasos - 1] = 1) then
      begin
        mediaTotal := mediaTotal + ((media) * inferenciaMaximo[y]);
      end
      else if (vetorAtraso[contAtrasos - 1] = 2) then
      begin
        mediaTotal := mediaTotal + ((2 * media) * inferenciaMaximo[y]);
      end
      else if (vetorAtraso[contAtrasos - 1] = 3) then
      begin
        mediaTotal := mediaTotal + ((3 * media) * inferenciaMaximo[y]);
      end;

      y := y + 1;
      contAtrasos := contAtrasos + 1;
    end;

    mediaTotal := mediaTotal / qntViagensAtrasadas;
    horaSugerida := StrToTime(grid.Cells[2,viagemAtual]);
    DecodeTime(horaSugerida, Hour, Min, Sec, MSec);
    //Adiciona horário adicional adquirido pela defuzzificação
    quantidadeSomaMinutos := (hour*60) + min - round(mediaTotal);
    hora := quantidadeSomaMinutos div 60;
    minuto := quantidadeSomaMinutos mod 60;
    novaHoraSugerida := StrToTime(IntToStr(hora)+':'+IntToStr(minuto)+':00');
    atualizaHoraSugerida(viagemAtual,TimeToStr(novaHoraSugerida));
```