Universidade do Estado do Rio de Janeiro
Centro de Tecnologia e Ciências
Faculdade de Engenharia

Ricardo Fortes Pedrozo

Sistema De Estimativa Do Melhor Caminho Multiponto a Multiponto Em Um Espaço Geográfico

Ricardo Fortes Pedrozo

Sistema De Estimativa Do Melhor Caminho Multiponto a Multiponto Em Um Espaço Geográfico

Dissertação submetida ao corpo docente da Faculdade de Engenharia da Universidade do Estado do Rio de Janeiro – UERJ, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Computação – área de concentração Geomática.

Orientador: Professor Orlando Bernardo Filho, D. Sc.

Co-orientador: Flávio Joaquim de Souza, D. Sc.

Programa de Pós-Graduação em Engenharia de Computação – Área de Concentração Geomática

Rio de Janeiro
2008

PEDROZO, RICARDO FORTES

Sistema de Estimativa do Melhor Caminho Multiponto a Multiponto em um Espaço Geográfico

[Rio de Janeiro] 2008

x, 78 p. 29,7 cm, (FEN/UERJ, M.Sc., Engenharia de Computação – Área de Concentração Geomática, 2008)

Dissertação – Universidade do Estado do Rio de Janeiro, FEN

1. Sistemas de Informações Geográficas
2. Lógica *Fuzzy*
3. Busca Exaustiva

I. FEN/UERJ II. Título

Ricardo Fortes Pedrozo

Sistema de Estimativa do Melhor Caminho Multiponto a Multiponto em um Espaço Geográfico

Dissertação submetida ao corpo docente da Faculdade de Engenharia da Universidade do Estado do Rio de Janeiro – UERJ, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Computação – área de concentração Geomática.

Aprovado em: 19/09/2008

Banca Examinadora:

_____

Orientador: Orlando Bernardo Filho, D.Sc., UERJ

_____

Co-orientador: Flávio Joaquim de Souza, D.Sc., UERJ

_____

Guilherme Lúcio Abelha Mota, D. Sc., UERJ

_____

Jorge Lopes de Souza Leão, Dr. Ing., UFRJ

DEDICATÓRIA

Aos meus pais pelo apoio incondicional
durante o curso e no decorrer deste trabalho.
À minha pequena filha Laura
pelo incentivo a seguir com esse trabalho

AGRADECIMENTOS

À UERJ.

Ao departamento de Engenharia de Sistemas e Computação e ao departamento de Engenharia Cartográfica, pelo padrão do curso.

Ao professor orientador Orlando Bernardo Filho, por ter sido um orientador exemplar e bastante atencioso.

Ao meu colega de trabalho Michel Demitrov, pela ajuda em um ponto crucial no desenvolvimento do trabalho.

Ao professor co-orientador Flávio Joaquim, por ter contribuído bastante no decorrer desse trabalho com o seu vasto conhecimento na área de Inteligência Computacional.

À minha querida amiga Luciana Campos Mota que foi quem abriu a chance para o começo desse trabalho.

À amiga Renata Mello da Silva Brito que contribuiu na parte referente à língua estrangeira no presente trabalho.

Aos meus amigos de mais de vinte anos de convivência que sempre me incentivaram ao longo da confecção desse trabalho.

RESUMO

Esta dissertação apresenta o desenvolvimento de um sistema de tomada de decisão que propõe uma metodologia inteligente, de tal maneira a efetuar a melhor alocação possível de um grupo de usuários a um grupo de recursos em um espaço geográfico. Tal metodologia se baseou na lógica *fuzzy* e ao longo da dissertação foram feitas comparações com outras técnicas, como o Algoritmo Ingênuo e a Busca Exaustiva. O conjunto de dados que foi adotado como o escopo desse trabalho foi a matrícula de alunos do município de Nova Iguaçu.

Palavras-Chave: Sistemas de Informações Geográficas, Lógica *Fuzzy*, Busca Exaustiva.

ABSTRACT

This thesis presents the development of a decision system which proposes an intelligent methodology whose main purpose is making the best allocation as possible of a user's set to a resource's set in a geographic space. This methodology was supported on fuzzy logic and during the thesis; it was done comparisons with other techniques like Exhaustive Search and Plane Sweep. Two datasets were used in this thesis: a schools set and a students group, both situated in the city of Nova Iguaçu, RJ, Brazil.

Keywords: Exhaustive Search, Fuzzy Logic, Geographic Information System.

ÍNDICE

CAPÍTULO 1 – INTRODUÇÃO

1.1 – Apresentação

Existem diversas aplicações na área da geomática que envolvem a necessidade de distribuição de usuários em recursos como, por exemplo, a alocação de alunos em escolas da rede pública (FADEL, 2003; PIRES, 2002; MARIA *et. al.*, 2004; COLOMBO *et. al.*, 2006), eleitores em seções de zonas eleitorais, embarcações de resgate marítimo em embarcações acidentadas (ANDRADE, 2004) etc. Nesses casos, o que será chamado de usuários (alunos, eleitores, embarcações de resgate etc.) são alocados ao que chamar-se-á de recursos (escolas, seções eleitorais, embarcações sinistradas etc.) de acordo com a posição geográfica de tais recursos e usuários, ou seja, cada usuário seria alocado ao recurso mais próximo dele e que possuísse vaga e/ou condições técnicas para atendê-lo.

Tomando como base esse critério de menor distância entre usuário e recurso, ocorreria um problema quando se tivessem vários usuários próximos a um mesmo recurso o qual não possuísse vagas e/ou condições técnicas suficientes para alocar (ou atender) a todos. Sendo assim, alguns usuários acabariam ficando alocados nas suas outras opções de recursos mais próximos. Essa situação acaba por conduzir à questão de quais usuários seriam deslocados de sua opção ótima de recurso para dar lugar a outros.

Problemas de alocação de vários usuários a vários recursos envolvem árvores de possibilidades visto que, ao se alocar os usuários a qualquer um dos recursos, o primeiro usuário escolherá dentre os ***n*** recursos, o segundo escolherá dentre os ***n – 1*** recursos restantes e assim por diante até chegar ao último recurso. Isso recai em um problema de análise combinatória. Então, diante de muitos problemas que envolvem essas situações, começaram a aparecer sistemas que envolvem Busca Exaustiva, uma vez que a obtenção da solução ótima global exige a inspeção uma a uma das soluções possíveis.

Ocorre que a Busca Exaustiva depende intrinsecamente do tamanho da distribuição de usuários em recursos. Caso se tenha uma distribuição, por exemplo, de 100 usuários para serem organizados em 20 recursos com quatro vagas cada um, a Busca Exaustiva resultará em um número muito grande de possibilidades, tornando o processamento muito demorado e, por conseguinte, levar-se-ia muito tempo para se gerar a solução ótima do problema. É exatamente a busca de uma solução dessa dificuldade encontrada no método exaustivo que surgiu a proposta desta dissertação aqui apresentada.

1.2 – Motivação

Como o tema apresentado neste trabalho se trata de um problema de otimização, cabe aqui serem citadas duas outras técnicas que poderiam ser aplicadas como soluções válidas para a Alocação Multiponto a Multiponto. Tais técnicas são o Algoritmo Guloso (*Greedy Algorithm* – GA) e a Programação Dinâmica. Tais técnicas alternativas englobam o princípio de dividir e conquistar, em outras palavras, elas dividem todo o problema em subproblemas, de forma a encontrar a solução ótima para o mesmo.

O Algoritmo Guloso escolhe em um dado instante dentre os subproblemas gerados que vem a ser as iterações do mesmo qual deles gera a solução ótima. Essa estratégia determina previamente a subestrutura que venha a ser tal solução. É claro que, com esse critério nem sempre o GA conseguirá obter a solução ótima, principalmente em problemas de explosão combinatória considerável, mas é uma estratégia eficiente para problemas mais simples.

Já a Programação Dinâmica também subdivide o problema em vários subproblemas distintos, porém existe uma interdependência entre eles, ou seja, há um compartilhamento dos mesmos. Então existem as etapas correspondentes e cada uma delas representa um subproblema até que se chega à solução ótima.

Pelo exposto nas duas técnicas, se a complexidade do problema for alta, como é o caso do tema aqui apresentado que possui uma intensa explosão combinatória, geralmente essas técnicas tendem a ter um alto índice de falhas, uma vez que, à medida que se aumenta o número de usuários e recursos aumentam o número de combinações a serem feitas e, com isso, o número de iterações fica cada vez maior gerando lentidão no processamento dos subproblemas.

Tendo em vista a questão da alocação multiponto a multiponto (vários usuários em vários recursos), alguns trabalhos anteriores já foram propostos como tentativa para solucionar esse problema. Em um desses trabalhos (PIRES, 2002), foi empregado algoritmo genético na resolução da minimização do problema de acesso de alunos a escolas, isto é, proceder com a alocação de alunos, de forma a fazer com que as rotas desses alunos às respectivas unidades escolares fossem as mínimas possíveis. Em outras palavras, matricular os alunos em escolas as mais próximas possíveis dos mesmos.

No trabalho de (PIRES, 2002), o objetivo era fazer uma alocação mais racional dos alunos às escolas, de tal forma a resolver dois problemas de uma só vez: evitar a superlotação de determinadas escolas e obter uma rota mínima para o transporte dos alunos, evitando um

deslocamento grande de um aluno à sua escola. Destaca-se que a solução para esses problemas é toda calcada em otimização combinatória, através da utilização da teoria dos grafos com processamento evolucionário.

A restrição observada nessa solução comentada refere-se ao fato da otimização combinatória ser similar ao método de busca exaustiva e, por conseguinte, acaba esbarrando novamente no problema de processamento muito intenso, resultando em algoritmos baseados em recursão, que usados com muitas repetições, pode provocar até a escassez de memória para a busca nas árvores empregadas como principais estruturas de dados que armazenam as informações do problema considerado.

Uma segunda proposta apresentada para contornar o problema da alocação multiponto a multiponto pode ser destacada no trabalho de (MARIA *et alia*, 2004), no qual o enfoque é na obtenção do caminho mínimo de um lugar para outro, usando para tanto algoritmos que empregam heurística. A técnica de (MARIA *et alia*, 2004) emprega o algoritmo de Dijkstra como ponto de partida da seguinte maneira: em face da complexidade da busca da rota mínima em um espaço geográfico, foi feita uma divisão do problema completo em subproblemas nos quais foi usado o algoritmo de Dijkstra e, depois com o emprego de heurística, a solução completa foi sendo construída.

Além dessas propostas anteriores, é preciso também citar a dissertação de (FADEL, 2003), que utilizou um método para alocação de alunos em escolas baseado exclusivamente na menor distância entre cada um desses alunos e cada uma dessas escolas, sendo assim, o primeiro aluno cadastrado no banco seria matriculado na escola com vaga para a sua série mais próxima de si. No presente trabalho, o método empregado por (FADEL, 2003) será chamado de Algoritmo Ingênuo que será visto ao longo do capítulo V de resultados.

A motivação para o trabalho de (FADEL, 2003), que levou ao desenvolvimento de um sistema chamado MatriGeo, foi encontrar uma alternativa à busca exaustiva que possibilitasse a alocação multiponto a multiponto com um processamento mais rápido. Esse método funciona conforme o seguinte algoritmo:

1. Fazer leitura das coordenadas da residência do aluno (*Rxy*), através de um *clique* do *mouse* na posição do mapa onde se localiza o seu logradouro e criar lista de escolas inválidas vazia (*ListEinvalidas* ← *nil*).
2. Achar a escola (*E*) mais próxima de *Rxy*, consultando a localização de todas as escolas no banco de dados do MatriGeo (*bdMatriGeo*) com exceção daquelas em *ListEinvalidas*.

3. Caso a escola *E* possua vaga para o aluno, então efetuar a matrícula e FIM.
4. Caso a escola *E* não possua vaga para o aluno, fazer *ListEinvalidas* ← *ListEinvalidas* + *E*, vá para o passo 2.

Os passos do algoritmo anterior foram aplicados a cada usuário (aluno) previamente cadastrado (pré-matriculado) em *bdMatriGeo*. Uma vez que não havia uma ordenação com base em algum critério dos alunos em *bdMatriGeo*, o algoritmo alocava a melhor opção de escolas para os primeiros alunos cadastrados o que acabava excluindo os outros alunos cadastrados mais tarde de escolas mais próximas de suas residências, pois as mesmas já estariam com vagas esgotadas pelos primeiros alunos pré-matriculados.

Nesse método (Algoritmo Ingênuo), fica evidente a existência de uma fila de usuários na qual, os primeiros cadastrados levam vantagem sobre os demais na alocação das suas primeiras opções de recursos.

O ideal seria um tipo de alocação que buscasse uma otimização global, ou seja, que todos os usuários estivessem nas suas melhores opções, mas sem prejudicar em muito as melhores opções dos demais, em outras palavras, se certo usuário U_i foi alocado como sua primeira opção no recurso R_x com custo C_{ix} e um outro usuário U_j por não ter mais vaga em sua melhor opção, também em R_x, ficou em sua segunda opção em R_y com custo C_{jy}, totalizando um custo global até esse momento de $C_{ix} + C_{jy}$. Todavia, caso U_j ficasse com sua primeira opção em R_x com custo C_{jx} e U_i ficasse com a sua segunda opção em um outro recurso R_z com custo C_{iz}, poderia ser possível que o custo total agora $C_{iz} + C_{jx}$ fosse menor do que $C_{ix} + C_{jy}$, dependendo das posições geográficas de U_i, U_i, R_x, R_y, e R_z.

Para chegar a uma otimização global nesse cenário, seria necessário fazer todas as distribuições possíveis de usuários em recursos, calculando o custo global de cada uma e, por fim, escolhendo-se a distribuição com menor custo total. No entanto, esse seria o caso da Busca Exaustiva que já se sabe ser impraticável, dependendo do tamanho do problema.

É importante ressaltar que o trabalho de (FADEL, 2003) propôs uma alternativa à Busca Exaustiva, contudo o seu Algoritmo Ingênuo não conseguiu atingir o ideal do que trata a dualidade **Tempo x Solução Ótima**, pois ele é extremamente rápido, mas não há qualquer compromisso com a obtenção de uma solução próxima da ótima, uma vez que a solução ótima só se atinge com tempos proibitivos na própria Busca Exaustiva.

A dualidade **Tempo x Solução Ótima** tem como ideal uma solução com o menor tempo possível de processamento e uma solução próxima da ótima. Todavia pelo mostrado até aqui, ficou evidente que não existe ainda uma técnica que tenha maximizado o **Tempo x Solução**

Ótima. Por conta disso, a motivação desta dissertação é apresentar uma técnica que consiga conciliar um tempo mínimo de processamento concomitante com uma solução que não será a ótima, mas sim um resultado que seja próximo da mesma para o problema da alocação multiponto a multiponto (de usuários em recursos).

1.3 – Objetivos

O propósito deste trabalho é então apresentar uma técnica, empregando lógica nebulosa, para o problema da alocação de vários usuários a vários recursos dispostos em um espaço geográfico, de forma a se chegar a uma solução próxima da ótima, somente obtida pela busca exaustiva, e que ainda assim, possua um desempenho semelhante ao do algoritmo ingênuo.

A técnica aqui apresentada pretende aproveitar as vantagens da busca exaustiva e do algoritmo ingênuo, ou seja, um processamento tão rápido quanto o do algoritmo ingênuo, mas obtendo uma distribuição multiponto a multiponto com base em critérios que vise a descoberta da solução ótima somente garantida pela busca exaustiva.

O ponto de partida da técnica proposta será empregar um algoritmo semelhante ao ingênuo, alocando os usuários nos seus recursos preferenciais, todavia, no caso presente, não será considerado apenas o custo (distância) entre um usuário e um recurso para proceder com a alocação e sim um **Índice de Merecimento** desse usuário ao recurso.

Tal índice será fornecido por um conjunto de dois sistemas de inferência nebulosos que, para tanto, considerarão a segunda opção de recurso do usuário em questão, a competição existente pelo recurso pretendido para a alocação, além do custo, isto é, a distância entre o usuário e o recurso.

1.4 – Estrutura Da Dissertação

A documentação deste trabalho encontra-se dividida em seis capítulos e dois apêndices. O capítulo 2 apresenta os principais conceitos da Lógica Nebulosa e do algoritmo de Busca Exaustiva que serviram de base para a formulação da técnica de alocação multiponto a multiponto em um espaço geográfico.

No Capítulo 3, será mostrada a modelagem de um sistema protótipo de alocação multiponto a multiponto, criado para testar a técnica deste trabalho. Nas seções desse capítulo, serão descritos o Diagrama de Contexto, o Diagrama de Fluxo de Dados (DFD), o Modelo de Entidades e Relacionamento (Modelo Conceitual) e o Modelo Lógico de Dados do Sistema.

O Capítulo 4 contém todos os detalhes referentes à implementação do sistema protótipo, tais como Tabelas do Sistema, Algoritmo de Alocação e o Algoritmo de Inferência Nebulosa.

O Capítulo 5 trata dos resultados obtidos pela técnica implementada no protótipo e a comparação dos mesmos com a busca exaustiva e o algoritmo ingênuo. O Capítulo VI encerra a dissertação discorrendo sobre as conclusões inerentes a este trabalho, destacando os problemas que porventura continuam sem solução e o que pode ser aperfeiçoado nesta proposta de alocação multiponto a multiponto.

CAPÍTULO 2 – FUNDAMENTOS TEÓRICOS

2.1 – Introdução

Conforme já ilustrado na introdução, propostas anteriores foram apresentadas como soluções para a Alocação Multiponto a Multiponto, através de Algoritmos Genéticos, Busca Exaustiva e Teoria dos Grafos. Neste capítulo, serão apresentadas as bases teóricas empregadas na proposta para a solução do problema de Alocação Multiponto a Multiponto. Inicialmente, será vista a lógica *fuzzy* e depois os conceitos da busca exaustiva.

2.2 – Lógica *Fuzzy*

Toda vez que se trabalha com modelagem de uma situação elaborada por um usuário a fim de se projetar um sistema de informação sempre se procura desenvolver módulos baseados em certezas, ou seja, ou acontece um dado fato ou não ocorre tal fato. Isso advém do raciocínio baseado na lógica tradicional, isto é, a lógica clássica. Todavia, nem sempre na realidade para a qual se está desenvolvendo o modelo mais adequado possível ocorre o Sim ou Não, pois o raciocínio humano é freqüentemente inexato. Ele não funciona sempre com base em certezas, visto que de pessoa para pessoa existe também a sensibilidade e outros fatores que determinam as incertezas. Um exemplo disso é a percepção da temperatura, ou seja, dependendo do local considerado, uma temperatura de 29 graus pode ser muito alta para alguns, alta para outros e até amena para um outro grupo de pessoas. Poder-se-ia dizer até que, ao tentar-se modelar um problema do mundo real, muitas vezes depara-se com uma série de incertezas ao longo de tal modelagem. Então, como se pode encontrar uma solução adequada para dois aspectos a serem levados em conta na informação que são a imprecisão e a incerteza?

Algumas teorias foram desenvolvidas de forma a ajudar na solução de modelagens de uma realidade que envolve incertezas e imprecisões. A Teoria das Probabilidades foi o primeiro passo nesse sentido. Contudo, a probabilidade consegue apenas quantificar a freqüência de um evento; ela não consegue lidar com a imprecisão de uma informação, bem como a Teoria dos Conjuntos que aqui será denominada de Clássica para efeitos de comparação com o que será exposto a seguir. A Teoria dos Conjuntos Clássica também não consegue lidar com a incerteza e imprecisão das informações tratadas pelos seres humanos.

Diante disso, a partir da década de 60, um cientista chamado Lofti Zadeh (ZADEH, 1965) começou a desenvolver uma teoria, a ser chamada de Teoria dos Conjuntos *Fuzzy*, para tratar do aspecto vago da informação. Essa teoria é uma das vertentes da Inteligência Artificial, hoje denominada por muitos de Inteligência Computacional. Essa teoria visa modelar exatamente os conceitos de imprecisão e incerteza tão freqüentes na modelagem das informações do mundo real. No tópico seguinte, isso será visto mais detalhadamente, porém para se definir melhor a Teoria dos Conjuntos *Fuzzy*, ela se baseia no conceito de pertinência de um elemento a um conjunto. Em um conjunto *fuzzy*, um elemento tem um certo grau de pertinência. Em outras palavras, o elemento pode pertencer mais ao conjunto ou quase não aparecer a ele, ao contrário da lógica tradicional onde um elemento pertence ou não pertence a um certo conjunto. Então, o princípio da lógica ambivalente não consegue solucionar boa parte dos problemas do mundo real, visto que ele não consegue solucionar as imprecisões e incertezas do raciocínio humano.

Zadeh desenvolveu a partir do final da década de 70, a Teoria das Possibilidades que trata exatamente da incerteza da informação. Esta teoria é intrinsecamente ligada à Teoria dos Conjuntos *Fuzzy*, pois quando se modela um SIF como será focado adiante, antes de se modelar um conjunto *fuzzy*, é necessário se analisar as possibilidades que uma dada informação possui. Assim, pode-se analisar de forma mais clara a imprecisão e a incerteza de um conjunto de informações do mundo real.

A Teoria dos Conjuntos *Fuzzy* é largamente utilizada em sistemas baseados em conhecimento. Quando utilizada em um sentido Lógico, ela se transforma na área que conhecemos como Lógica *Fuzzy*

A Lógica *Fuzzy* encontra muitas aplicações no mundo real, tendo seu uso em sistemas como freios ABS, controle de eletrodomésticos, satélites e, também no auxílio em sistemas híbridos em conjuntos com as Redes Neurais, através do desenvolvimento de sistemas neuro-*fuzzy*. Por esse motivo, a Lógica *Fuzzy* tem uma vastidão de sistemas que se baseiam nela. A partir da sua utilização, tem-se desenvolvimento de sistemas de controle de processos sofisticados, bem como controladores simples, de fácil manutenção e baixo custo. Esses sistemas construídos com base na Lógica *Fuzzy* são denominados de Controladores Nebulosos ou Controladores *Fuzzy*. Eles serão descritos a seguir.

2.2.1 - Conceitos Básicos Da Teoria Dos Conjuntos *Fuzzy*

Conforme exposto no tópico anterior, a grande dificuldade de modelar problemas do mundo real é tratar duas questões: a imprecisão e a incerteza do raciocínio humano; o que é uma temperatura quente para uma pessoa, pode ser morna para outrem. Sendo assim, surgiu a Teoria dos Conjuntos *Fuzzy*.

Na Teoria Clássica dos Conjuntos o que se tem é o conceito de que dado um conjunto qualquer, um elemento pertence ou não a esse conjunto, ou seja, o grau de pertinência é ***0 OU 1***. Na Teoria dos Conjuntos *Fuzzy*, há a extensão desse conceito de pertinência. O elemento passa a ter um grau de pertinência compreendido no intervalo [0,1]. Isso faz com que se tenham inúmeros graus de pertinência dentro do intervalo em questão.

Antes de se prosseguir com o estudo corrente, é necessário se colocar alguns conceitos que serão largamente utilizados a frente. É preciso se definir Universo de Discurso, Termos Lingüísticos e Variável Lingüística.

Termos Lingüísticos são os conjuntos *fuzzy* que representarão os valores componentes de uma Variável Lingüística. Exemplo: ao se definir uma variável Lingüística Temperatura, podemos estabelecer 4 conjuntos *fuzzy*: ALTA, BAIXA, MÉDIA E AMENA. Um Universo de Discurso U é o intervalo no qual a Variável Lingüística encontra-se definida.

Uma Variável Lingüística é uma coleção de valores que pode ser representada por uma quádrupla (X, U, T(X), F), onde X é o nome da variável, U é o Universo de Discurso de X, T(X) é o conjunto dos seus Termos Lingüísticos e F é a função que associa uma função de pertinência a cada valor de T(x).

Os conjuntos *fuzzy* possuem um Universo de Discurso. Formalmente, um conjunto *fuzzy* A é definido através de uma função de pertinência definida dentro desse Universo de Discurso. A representação do conjunto *fuzzy* pode ser feita da seguinte maneira:

$$\mu A : \quad U \rightarrow [0,1] \qquad (2.1)$$

onde μA representa a função de pertinência que define o conjunto *fuzzy*, sendo que essa função mapeia o Universo de Discurso U ao intervalo de valores reais [0,1]. A função de pertinência determina o grau de quanto um determinado elemento x de U é pertencente a esse conjunto *fuzzy*. Assim esse grau é definido da seguinte maneira:

- μ A(x) = 1 significa que x pertence completamente ao conjunto *fuzzy*
- μ A(x) = 0 significa que x não pertence ao conjunto *fuzzy*
- 0 < μ A(x) < 1 significa que x tem um grau de pertinência relativo no conjunto *fuzzy*, cujo valor é μ A(x). Cabe ressaltar que quanto mais alto for esse valor, maior é a possibilidade de x pertencer a esse conjunto.

Na Teoria dos Conjuntos *Fuzzy*, se certo conjunto tiver apenas os valores 0 ou 1, isto é, se os valores são apenas "sim" ou "não", tal conjunto é denominado ***crisp***.

2.2.2 - Sistemas De Inferência *Fuzzy* (SIF's)

Os Sistemas de Inferência *Fuzzy* (SIFs) são sistemas baseados em conhecimento desenvolvidos a partir dos princípios da Teoria dos Conjuntos *Fuzzy* e da Lógica *Fuzzy*. Aqui nesta seção serão vistos os componentes desse sistema detalhadamente. O desenvolvimento das técnicas de inferência para os SIF's tiveram início com as pesquisas e projetos de E . H. Mamdani (MAMDANI, 1976).

Ao contrário dos controladores convencionais em que o algoritmo de controle é descrito analiticamente por um modelo matemático, em Controles *Fuzzy* utilizam-se regras lógicas no algoritmo de controle com o intuito de descrever em uma rotina a heurística, o raciocínio humano e intuição para controlar um processo.

2.2.2.1 – Componentes de Um SIF

Como visto na Teoria dos Conjuntos *Fuzzy*, para se projetar um SIF, é preciso ter antes, de mais nada, as Variáveis Lingüísticas nas quais os conjuntos *fuzzy* atuarão. Além disso, ter-se-á no SIF também, a base de regras e as Interfaces de *fuzzificação* e *defuzzificação*. Ilustrando esse conceito de Variável Lingüística, a figura abaixo representa a Variável Lingüística Velocidade.

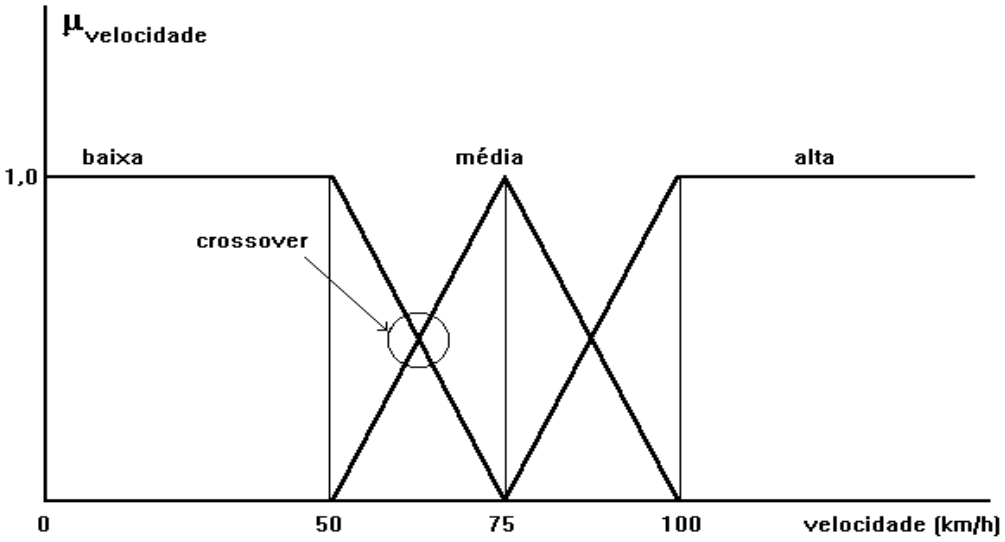


Figura 1 – VARIÁVEL LINGÜÍSTICA VELOCIDADE

A partir dela verificam-se os Conjuntos *fuzzy* que representam os termos lingüísticos. Cada termo lingüístico tem um intervalo que determina em que momento a velocidade torna-se Positiva Baixa ou Positiva Alta, por exemplo.

O grau com que o valor x* em U satisfaz um termo lingüístico Y é a pertinência de x* em Y, dada por μ Y (x).

A figura a seguir representa um SIF (Sistema de Inferência *Fuzzy*) e seus componentes. Em geral, os SIF's têm maior utilidade em sistemas de controle não-lineares, sendo capazes de superar perturbações e plantas com níveis de ruídos.

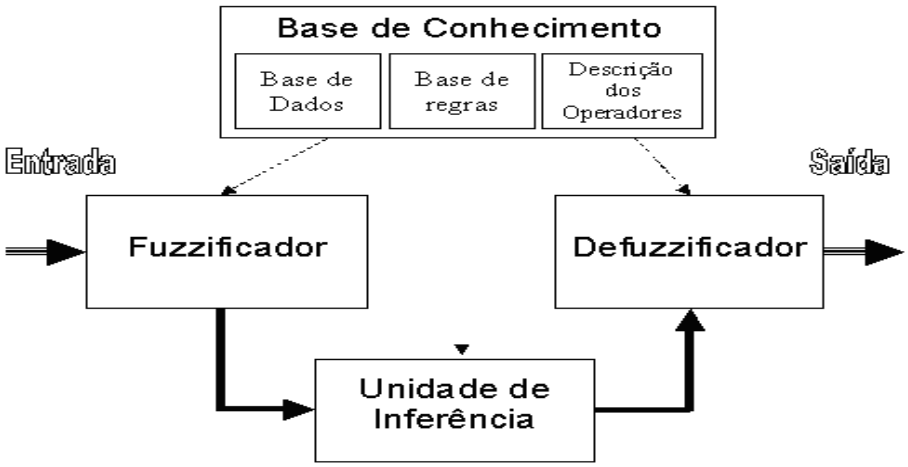


Figura 2 – UM SISTEMA DE INFERÊNCIA *FUZZY*

A seguir, serão descritos os componentes do Sistema de Inferência *Fuzzy*.

2.2.2.2 - Interface De *Fuzzificação*

Fuzzificação é o primeiro passo na computação de um Sistema de Inferência *Fuzzy* e deve ser executado para cada valor preciso de entrada, ou seja, um valor do universo de discurso de cada antecedente das regras de inferência nebulosas. Neste trabalho, a *fuzzificação* adotada foi a do método *singleton* o que corresponde, na prática, a simplesmente avaliar a função de pertinência dos termos de cada antecedente com os valores precisos desses antecedentes fornecidos como entrada.

2.2.2.3 - Base De Conhecimento

A base de conhecimento se compõe de duas partes: uma base de dados e uma base de regras, de tal forma, a deixar bem claro a estratégia de controle e seus objetivos.

Na base de dados se encontram armazenadas as definições sobre discretização e normalização dos universos de discurso, e as definições das funções de pertinência dos termos *fuzz*y.

A base de regras é formada por estruturas do tipo:

Se <premissa> **Então** <conclusão>

Um exemplo é a variável lingüística velocidade que foi apresentada anteriormente, com a qual se poderia considerar a seguinte regra:

SE *Erro é Negativo Grande e* Δ*Erro é Positivo Pequeno* ***ENTÃO*** *Velocidade é Positiva Pequena*

Essas regras, juntamente com os dados de entrada, são processadas pelo procedimento de inferência, que infere as ações de controle de acordo com o estado de sistema, aplicando o operador de implicação, conforme o procedimento de inferência que será descrito na próxima seção.

Em um dado SIF, é importante ressaltar que existem tantas regras quantas forem necessárias para mapear totalmente as combinações dos termos das variáveis, ou seja, que a base seja completa, de forma a assegurar que sempre haverá ao menos uma regra a ser disparada para qualquer entrada. Também são fundamentais: a consistência, a fim de se evitar

a mínima possibilidade de contradições; e, a interação entre as regras, gerenciada pela função de implicação de modo a contornar as situações de ciclo. É bom lembrar, todavia, que essa base de regras deve ser a mais simples possível de tal maneira que o SIF possa ser processado com eficiência.

2.2.2.4 - Procedimento De Inferência

Um SIF é um sistema especialista simplificado onde a conseqüência de uma regra não é aplicada como antecedente de outra. De acordo com (SANDRI *et al.*, 1999), o procedimento de inferência consiste em:

1. Verificação do grau de compatibilidade entre os fatos e as condições nas premissas das regras;
2. Determinação do grau de compatibilidade global da premissa de cada regra;
3. Determinação do valor de conclusão em função do grau de compatibilidade da regra com os dados e ação de controle constante na conclusão (exata ou não);
4. Adição dos valores obtidos como conclusão das várias regras, obtendo-se uma ação de controle global.

Os tipos de SIF's mais comuns encontrados dentro deste assunto são os modelos tradicionais de Mamdani e Larsen e, em relação aos modelos de interpolação, os mais comuns são os de Takagi-Sugeno e Tsukamoto. Os modelos se distinguem no tocante à representação dos termos nas premissas, quanto às ações de controle e em relação aos operadores utilizados para implementação do sistema *fuzzy*.

Nos modelos *fuzzy* tradicionais, quando se conclui cada regra, há a especificação do termo *fuzzy* dentre um conjunto fixo de termos (em geral em número inferior ao de regras). Esses termos são conjuntos *fuzzy* compreendidos entre os perfis mais comuns que são o triangular, o sino (*bell-shaped*) e o trapezoidal.

Sejam as regras Rj codificadas como:

*Rj : **Se** x1 é A1,j **e ...e** xn é An,j **Então** yj é Cj*

No modelo tradicional (SANDRI *et al.*, 1999), o processamento de inferência é efetuado da seguinte maneira:

. **Passo 1**: Seja xi uma variável de estado, definida no universo Xi, a realização de xi é definida como o valor x*i $\in$ Xi que essa assume em Xi em um dado instante;

. **Passo 2**: A *compatibilidade* da i-ésima premissa da j-ésima regra com x*i, ou seja, a compatibilidade de x*i, $1 \leq i \leq n$, com A i,j da regra Rj $1 \leq j \leq m$;

. **Passo 3**: Com as premissas de uma dada regra avaliadas, a *compatibilidade global* α j da regra Rj , $1 \leq j \leq m$, com os x*i é determinada com uma t-norma T:

$$\alpha j = T(\alpha j, \alpha j, ..., \alpha j), \quad 1 \leq j \leq m \tag{2.2}$$

. **Passo 4**: O α j assim obtido é relacionado com o respectivo conjunto *fuzzy* Cj do conseqüente da regra Rj, dando origem a um conjunto C'j, $1 \leq j \leq m$, através de um operador de implicação I:

$$\mu C'j(y) = I(\alpha j, \mu C'j(y)), \quad \forall y \in Y \tag{2.3}$$

. Passo 5: Um operador ∇ faz a *agregação* das contribuições das várias regras acionadas C'j num único conjunto *fuzzy* C':

$$\mu C'j(y) = \nabla (\mu C'1(y), ..., (\mu C'm(y)), \quad \forall y \in Y \tag{2.4}$$

O operador ∇ habitualmente funciona como uma *t-conorma* quando o operador de implicação I é uma *t-norma* e uma *t-norma* em caso contrário.

Os modelos tradicionais seguem estritamente os passos mostrados anteriormente, lembrando que em Mamdani tem-se T(a,b) = min (a,b), I = min(a,b) e ∇(a,b) = max(a,b) e em Larsen as fórmulas ficam da seguinte forma: T(a,b) = a * b, I = a * b e ∇(a,b) = Max(a,b). As figuras seguintes mostram os dois modelos e todo o desenvolvimento de raciocínio de ambos os modelos.

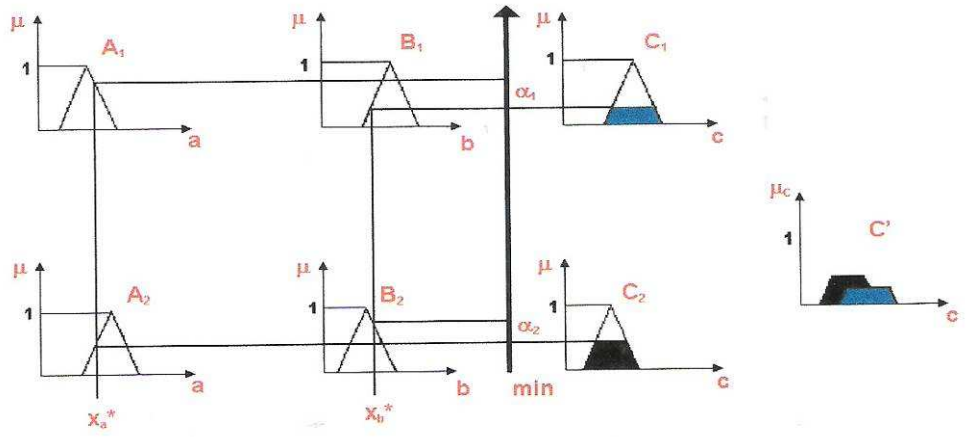


Figura 3 - MODELO DE MAMDANI

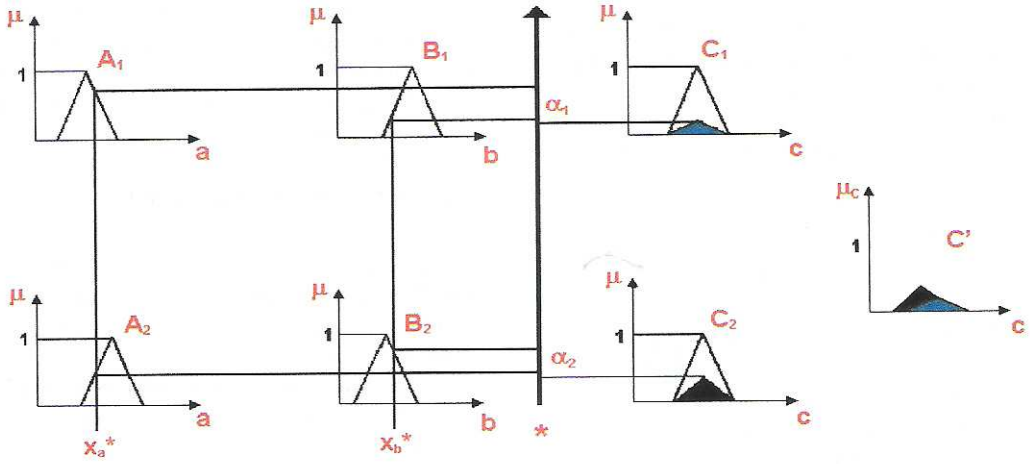


Figura 4 - MODELO DE LARSEN

2.2.2.5 - Interface De *Defuzzificação*

Nos SIF's tradicionais, a interface de *defuzzificação* serve para se obter uma única solução precisa a partir de um conjunto *fuzzy* C' resultante. O procedimento abrange a identificação de domínio das variáveis de saída em um correspondente universo de discurso e com a ação de controle *fuzzy* inferida evolui-se uma saída de controle já *defuzzificada*. Os Métodos de *defuzzificação* mais utilizados de acordo com a literatura são:

Primeiro Máximo: Encontra o valor de saída através do ponto em que o grau de pertinência da distribuição da ação de controle atinge o primeiro valor máximo.

Método da Média dos Máximos (Média Ponderada): Encontra o ponto médio entre os valores que possuem o maior grau de pertinência inferido pelas regras. Este Método combina os conseqüentes das regras *fuzzy* com o nível de disparo de cada uma delas gerando uma saída *crisp* Z . A fórmula que gera essa saída é dada por:

$$Z = \frac{\sum_{i=1}^{n} (\mu_i * Z_i)}{\sum_{j=1}^{n} \mu_i} \tag{2.5}$$

Método do Centro de Área: Este é um dos métodos mais utilizados, senão o mais utilizado. Ele consiste em utilizar o centróide (centro de massa) da função de distribuição de possibilidade da regra de disparo. A expressão que representa este método é dada por:

$$Z = \frac{\sum_{i=0}^{m} (\mu_c(z_i) * Z_i)}{\sum_{j=1}^{n} \mu_c(z_i)} \tag{2.6}$$

onde, ‘m’ é o número de intervalos de quantização da saída, Zi é a saída para o intervalo de quantização ‘i’ e $\mu_C(Zi)$ seu grau de pertinência.

É bom ressaltar que, apesar do Método do Centro de Área (COA) ser o mais comumente utilizado, a seleção do método dependerá muito das circunstâncias em que o SIF foi modelado, pois se for empregado o método errado poder-se-á incorrer em equívocos quando do resultado de saída do Sistema de Inferência *Fuzzy*.

2.3. Busca Exaustiva

Certos problemas de computação envolvem buscas onde se tem um número interminável de soluções para se achar a solução ótima ou são problemas de complexidade tamanha que não têm soluções por algoritmos eficientes. Neste tópico, estudar-se-ão alguns desses problemas, onde aplicar-se-ão técnicas úteis para resolvê-los.

A Busca Exaustiva pode ser usada para encontrar o melhor caminho para uma rota de um ponto a outro através de otimizações empregando-se, para isso, de matemática combinatória. Os algoritmos que já foram testados em Busca Exaustiva se edificam em

soluções através de grafos. Os algoritmos ideais para a solução da Busca Exaustiva necessariamente devem ter geralmente uma complexidade próxima de ***n*** ou ***logn***. Um problema clássico de Busca Exaustiva é o Problema do Caixeiro Viajante. O algoritmo proposto para alocação de vários usuários a vários recursos, como já foi colocado, tem complexidade não polinomial o que, de certa forma, já o excluiria de ser um algoritmo eficiente de Busca Exaustiva.

Esse tipo de algoritmo, como citado, utiliza-se de matemática combinatória, pois ele se baseia na mesma para fazer as otimizações necessárias, visando a uma solução ótima. Esse método funciona para casos simples, dado que combinações e /ou arranjos que ultrapassam certo valor para se calcular um fatorial (20! por exemplo) já se tornam custosos demais, uma vez que são utilizadas árvores de decisão para o cálculo dos mesmos e em geral, boa parte dos algoritmos para cálculo de fatorial se utiliza de recursividade. Sabe-se que essa, sendo utilizada a um número de níveis torna-se penosa. É por isso que a Busca Exaustiva é contra-indicada para valores a partir de 20, 30. O algoritmo que gera a Alocação Multiusuário é de complexidade NP-Completo, ou seja, muito distante do ideal da complexidade da Busca Exaustiva (***n*** ou ***logn***). A Busca Exaustiva se calca na Teoria dos Grafos, onde a busca em um determinado vértice do grafo pode se tornar custosa quando se tem um número grande de vértices no mesmo. Logo, a Busca Exaustiva só funciona como esperado em um ambiente em que haja pouca complexidade, ou seja, em um grafo relativamente simples com poucos vértices e uma distância entre os mesmos que seja o mais curta possível. A partir desse ponto serão citadas algumas das técnicas inerentes à Busca Exaustiva.

2.3.1 - Geração De Permutações

Esta é a técnica utilizada neste trabalho, onde se usa um caso mais particular que é o das possíveis combinações entre usuários e recursos. Aqui se estudam algoritmos que possam gerar possíveis maneiras de se agrupar ***n*** ítems distintos. O problema de permutação de todo um conjunto de elementos é solucionado através da Busca Exaustiva, bem como a combinação de um certo número de elementos a uma certa quantidade de termos. Só o fato de usar a função de fatorial para se gerar o algoritmo de solução já evidencia a utilização de recursão. A Busca Exaustiva entra como um método perfeitamente adequado para a solução de problemas de análise combinatória.

Na figura a seguir, a técnica que objetiva listar todas as permutações de todos os elementos envolvidos. Nessa proposta é citada uma técnica similar, a da geração de

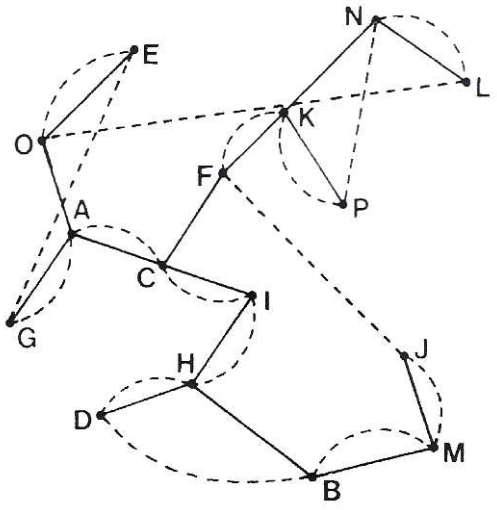
permutações, que é a geração de todas as combinações. Essa técnica gera um número menor de elementos.

1	***2***	***3***	***4***	***2***	***3***	1	4
1	2	4	3	***2***	***4***	1	3
1	***3***	***2***	***4***	***3***	***2***	1	4
1	4	2	3	***4***	***2***	1	3
1	3	4	2	***3***	***4***	1	2
1	4	3	2	***4***	***3***	1	2
2	1	3	4	***2***	***3***	***4***	1
2	***1***	***4***	***3***	***2***	***4***	***3***	1
3	1	2	4	***3***	***2***	***4***	1
4	1	2	3	***4***	***2***	***3***	***1***
3	***1***	***4***	***2***	***3***	***4***	***2***	1
4	1	3	2	***4***	***3***	***2***	1

Figura 5 – GERAÇÃO DE PERMUTAÇÕES

2.3.2 - Algoritmos De Aproximação

Ao procurar o caminho mais curto em um grafo com muitos nós, percebe-se o quão longo será o número de passos para se encontrar a solução ótima para esse problema. Tendo em vista que o tempo de processamento e memória é longo o suficiente para tal, começou-se o estudo de Algoritmos de Aproximação, de forma a se reduzir a quantidade de processamento para a solução da Busca Exaustiva. Esses algoritmos de Aproximação são mostrados nas figuras a seguir.



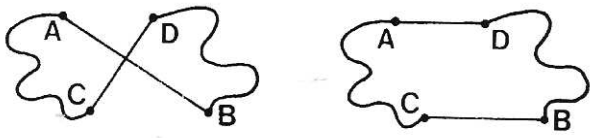


Figura 6 - EXEMPLOS DE ALGORITMOS DE APROXIMAÇÃO

2.4 - Comentários

Neste capítulo, foram apresentadas breves explanações sobre a Busca Exaustiva, pois esse método serve de comparação em relação à eficiência do Método *Fuzzy* sobre o mesmo. Também foi colocada uma explanação de toda a Teoria dos Conjuntos *Fuzzy*, base da proposta de solução *fuzzy*, apresentada neste trabalho.

CAPÍTULO 3 - MODELAGEM DO PROTÓTIPO

3.1 – Introdução

Neste capítulo será descrito o protótipo da solução do problema da alocação multiponto a multiponto, tratando-se de um Sistema de Tomada de Decisão que alocará os usuários em um dado número de recursos conforme a disponibilidade de vagas em cada recurso. Esse protótipo foi desenvolvido em Delphi 7.0 e a sua implementação contempla a Lógica *Fuzzy*, apresentada no capítulo anterior. Aqui será descrita a sua documentação, a saber: Diagrama de Contexto, DFD, Modelo de Dados (MER) e o Projeto Lógico.

3.2 - Arquitetura

Basicamente, o protótipo possui três funções: cadastrar os usuários, cadastrar os recursos e fazer a alocação dos mesmos. A ilustração da sua arquitetura é exposta através de um Diagrama de Contexto e de um Diagrama de Fluxo de Dados (DFD), mostrando como as funções do aplicativo interagem entre si. Optou-se por utilizar os diagramas da Análise Estruturada Moderna e não da Linguagem de Modelagem Unificada (UML), pois o sistema foi projetado com base nos moldes tradicionais da programação estruturada e a modelagem da base de dados se apoiou em banco de dados relacionais.

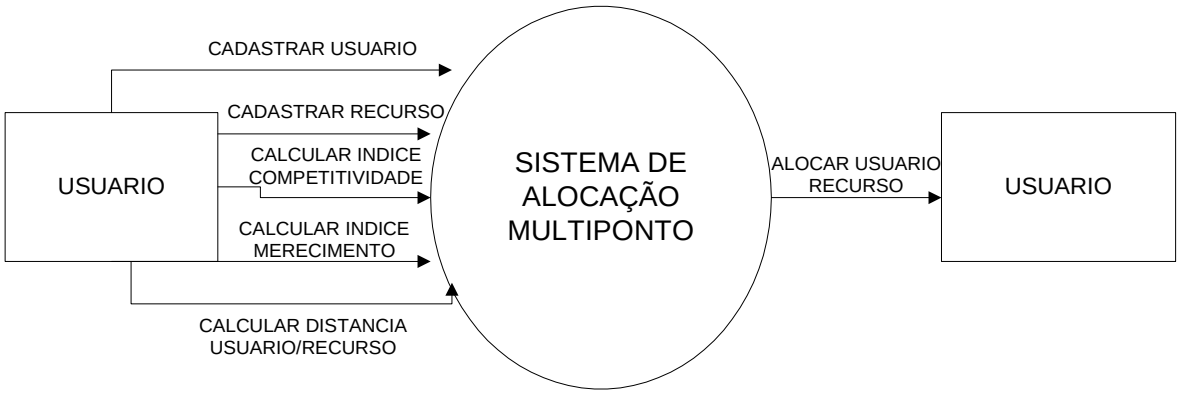


Figura 7 - DIAGRAMA DE CONTEXTO DO SISTEMA DE ALOCAÇÃO MULTIPONTO A MULTIPONTO

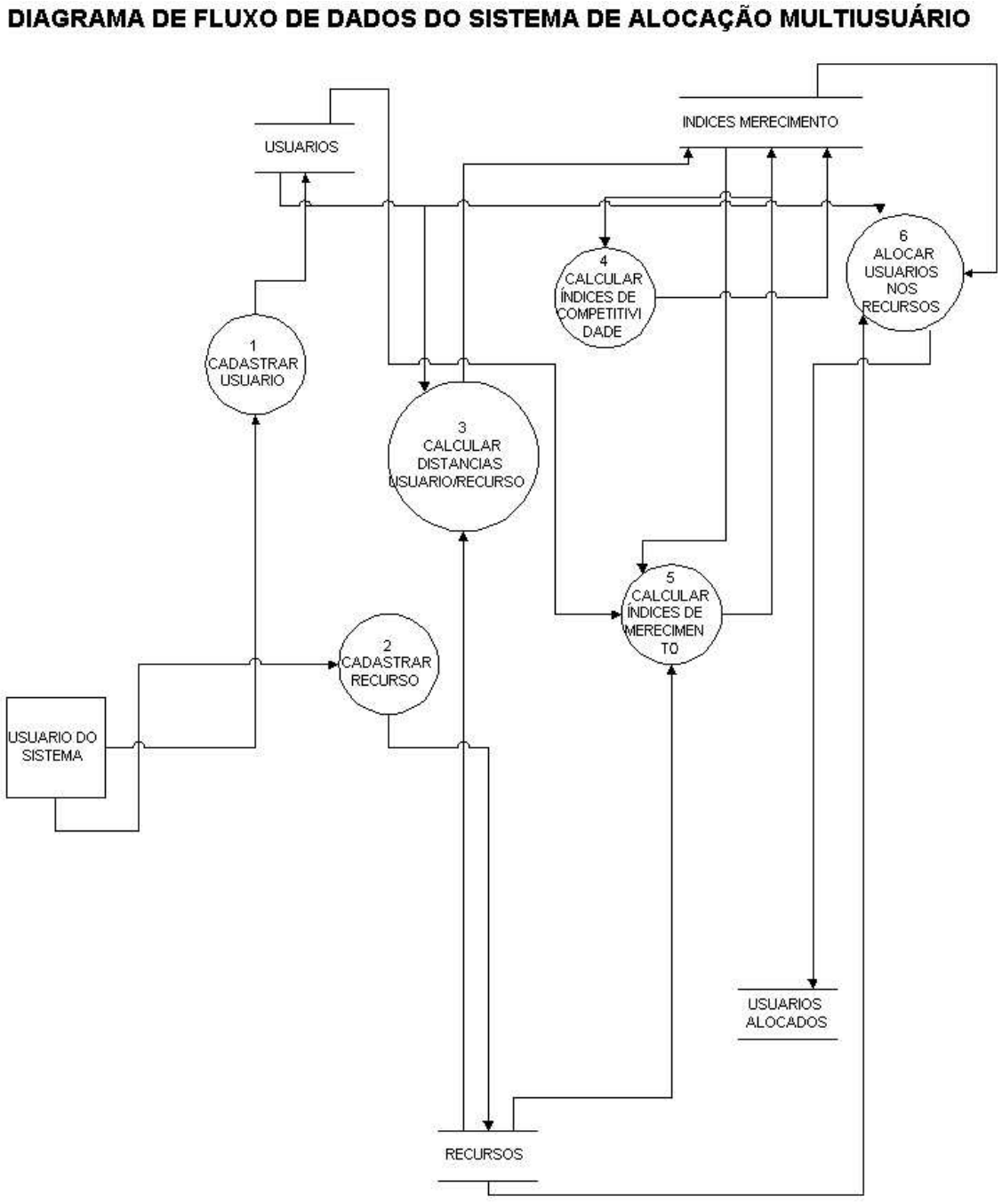


Figura 8 - DFD DO PROTÓTIPO DO SISTEMA MULTIPONTO A MULIPONTO

Os componentes do DFD são os seguintes:

USUÁRIO DO SISTEMA – Entidade externa que representa o operador do sistema.

CADASTRAR USUÁRIO – Processo que representa a função de cadastro dos usuários candidatos a recursos no sistema.

CADASTRAR RECURSO - Processo que cadastra os recursos do sistema.

CALCULAR DISTÂNCIAS USUÁRIO-RECURSO – Processo do cálculo das distâncias entre usuários e recursos.

CALCULAR ÍNDICES DE COMPETITIVIDADE - Processo que calcula os índices de competitividade entre os usuários e recursos

CALCULAR ÍNDICE DE MERECIMENTO – Processo que calcula os índices de merecimento.

ALOCAR USUÁRIOS NOS RECURSOS – Processo que aloca os usuários nos recursos

RECURSOS – Depósito de dados que representa os recursos cadastrados no sistema

USUARIOS – Depósito caracterizado pelo arquivo que armazena os usuários candidatos a recursos

ÍNDICES MERECIMENTO – Depósito de dados que armazena os índices de merecimento e as distâncias entre os usuários e os recursos.

USUÁRIOS ALOCADOS – Depósito que guarda as informações sobre em quais recursos os usuários foram alocados.

3.3 - Modelo Conceitual Do Banco De Dados

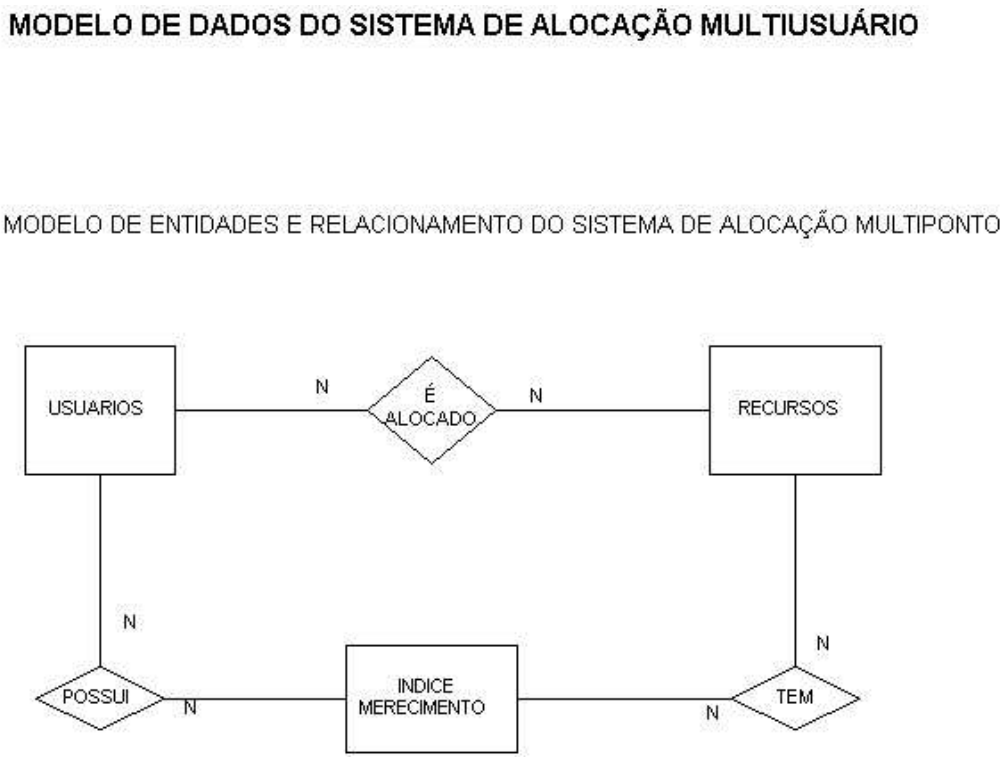


Figura 9 – MODELO DE ENTIDADES-RELACIONAMENTOS DO PROTÓTIPO

3.3.1 - Modelo De Entidades E Relacionamento Do Protótipo

Na modelagem de dados, conforme a proposta exaustivamente debatida até aqui, pode-se afirmar que a cardinalidade do relacionamento é N para N, uma vez que um usuário tem vários recursos em que ele pode ser alocado e um recurso pode receber vários usuários. Os componentes são os seguintes:

USUARIO – Entidade representativa dos usuários a serem incluídos nos recursos

RECURSOS – Entidade que vem a ser o conjunto de todos os recursos disponíveis no sistema.

INDICE MERECIMENTO – Entidade que representa o conjunto de todos os índices de merecimento calculados entre os recursos.

RELACIONAMENTO TEM – Relacionamento representativo do recurso ter ou não índices de merecimento

RELACIONAMENTO POSSUI – Relacionamento representativo do usuário ter ou não índices de merecimento

RELACIONAMENTO É ALOCADO – Esse é o relacionamento central do modelo, pois representa as alocações dos usuários dentro dos recursos.

3.4 - Modelo Lógico

O Modelo Lógico derivado do Modelo Conceitual, em que se tem o relacionamento Usuário/Recurso de N para N, onde um usuário pode ir para vários recursos e um recurso pode receber vários usuários. O Modelo Lógico da proposta é apresentado a seguir:

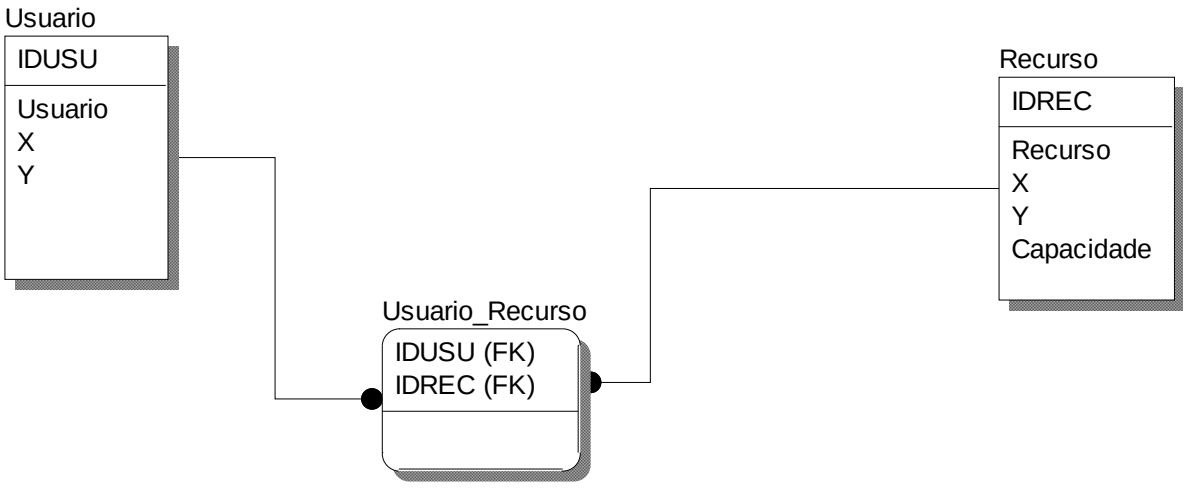


Figura 10 – MODELO LÓGICO DO PROTÓTIPO

3.5 - Modelo *Fuzzy*

Diante da inviabilidade demonstrada no algoritmo de solução do problema de alocação de vários usuários a vários recursos, pois está comprovado que a sua complexidade é em tempo fatorial, é preciso propor uma alternativa a essa solução que seja mais simples, ou seja, que tenha um tempo de execução bem mais rápido. Sendo assim, foi concebida uma proposta, aqui apresentada, que se trata de uma alternativa viável para a solução desse problema. A proposta deste trabalho encontra-se suportada pela Lógica *Fuzzy*, que foi apresentada no Capítulo II. Foi desenvolvido, portanto, um sistema de inferência *fuzzy* para compor o protótipo do sistema de alocação multiponto a multiponto.

Como essa proposta engloba dois índices *fuzzy*, um secundário e outro principal, houve a necessidade de se projetar dois SIF's: um para calcular o índice secundário e outro para se calcular o índice principal, ou seja, aquele índice que será determinante ou não para a alocação de certo usuário em um recurso. A saída do SIF1 constitui uma das entradas do SIF2.

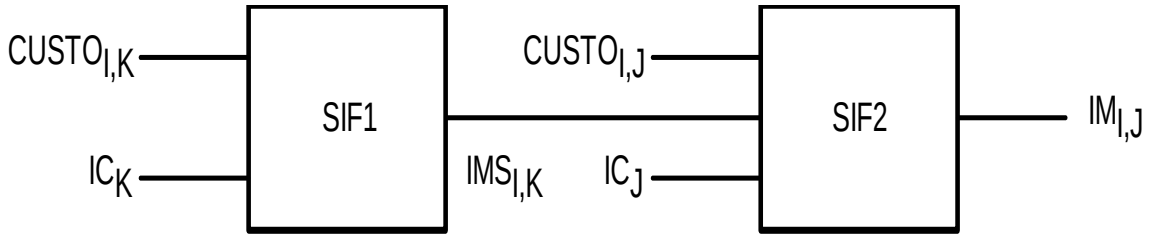


Figura 11 – O MODELO *FUZZY* PARA O SISTEMA MULTIPONTO A MULTIPONTO

No modelo proposto na figura anterior, o cerne da questão é tratar o problema considerando-se três variáveis: o custo (distância do recurso ao usuário), capacidade e demanda. Então, no primeiro SIF a saída é o índice de merecimento em relação ao segundo recurso (segunda melhor opção do usuário) que é utilizado como entrada no segundo SIF, além da distância do recurso ao usuário e o índice de competitividade do recurso que é a demanda dividida pela capacidade.

O modelo é composto de dois sistemas de inferência *fuzzy* sendo que o SIF2 é o sistema que determinará o recurso a ser alocado pelo usuário, ou seja, o SIF2 no modelo é o sistema principal que depende intrinsecamente do que ocorre no SIF1. O SIF1 escolhe a melhor opção

de alocação para o usuário diante de uma competição com outro por um recurso, colocando esse na melhor opção possível, diante da impossibilidade dele ser alocado na sua opção ótima.

3.5.1 - Variáveis Lingüísticas

No SIF1, têm-se as seguintes variáveis:

Na entrada: $CUSTO_{I,K}$ e IC_K

Na saída: $IMS_{I,K}$

No SIF2, as variáveis são:

Na entrada: $CUSTO_{I,J}$, IC_J e $IMS_{I,K}$

Na saída: $IM_{I,J}$

É bom frisar que a variável $IMS_{I,K}$ não possui a mesma definição das funções de pertinência de seus termos quando é usada como saída do SIF1 e entrada do SIF2. Em outras palavras, a variável $IMS_{I,K}$ possui os mesmos termos *fuzzy* (BAIXO, MÉDIO e ALTO) e o mesmo universo de discurso tanto na saída do SIF1 quanto na entrada do SIF2, todavia as funções de pertinência desses termos *fuzzy* são diferentes quando são utilizados como saída do SIF1 e entrada do SIF2.

A seguir, é apresentada a definição das funções de pertinência de todos os termos *fuzzy* de todas as variáveis lingüísticas do sistema protótipo e, para diferenciar a variável $IMS_{I,K}$ quando tratar-se de saída do SIF1 e entrada do SIF2, ela será mostrada como $IMS_{I,K}^{S}$ (saída do SIF1) e $IMS_{I,K}^{E}$ (entrada do SIF2):

$CUSTO_{I,K}$ – O custo é a distância do usuário a um recurso que é a segunda melhor opção de alocação, ou seja, é a menor distância encontrada depois da opção ótima. Os valores dessa variável (conjuntos *fuzzy*) são: ZERO, BAIXO, MÉDIO E ALTO e seu universo de discurso é de 0 até o Perímetro da Região do Mapa onde será analisada tal distância.

IC_K – O IC_K é o índice de competitividade do recurso que é a melhor opção do usuário depois da sua opção ótima. Ele é medido dividindo-se o total de usuários pela capacidade média deste recurso. Os valores (conjuntos *fuzzy*) desta variável são BAIXO, MÉDIO e ALTO e o seu universo de discurso é de 0 até o valor máximo desse índice que é o total de

usuários que estão disputando recursos dividido pela capacidade média dos recursos, ou seja, a média aritmética da capacidade de todos os recursos envolvidos.

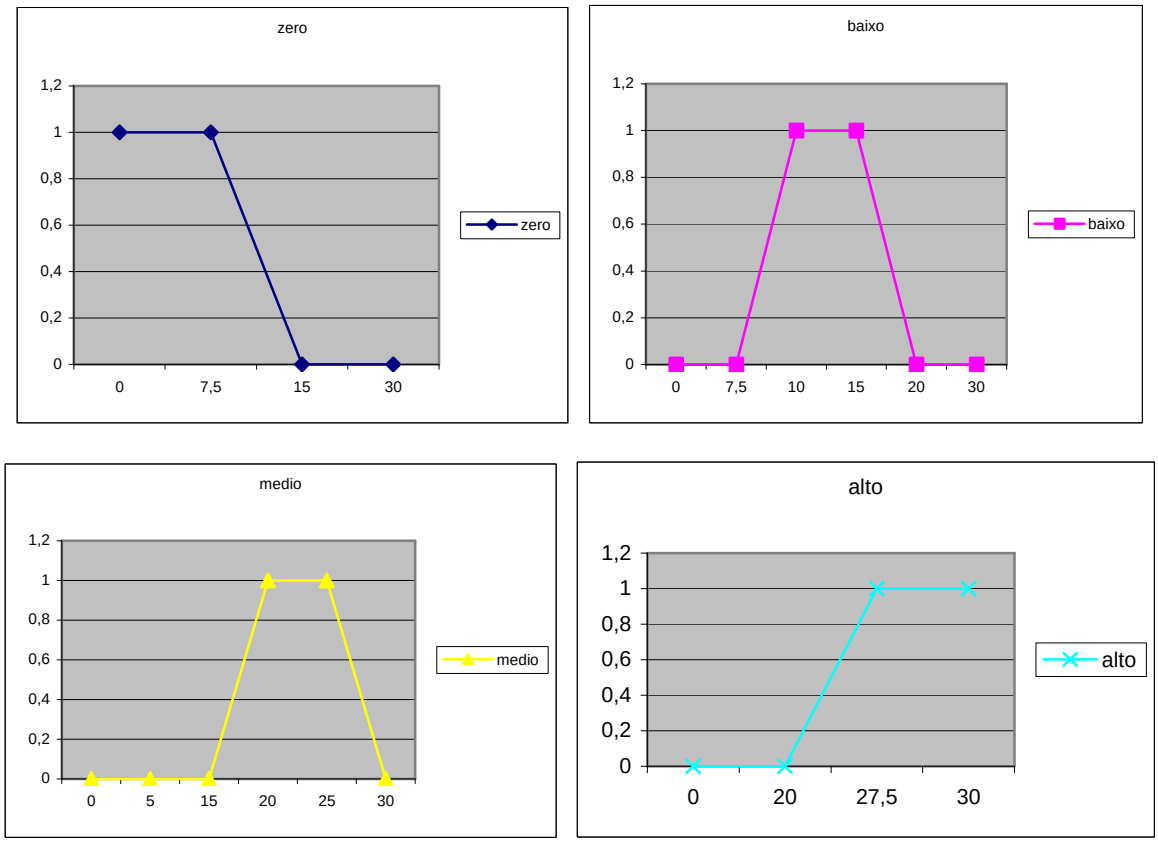
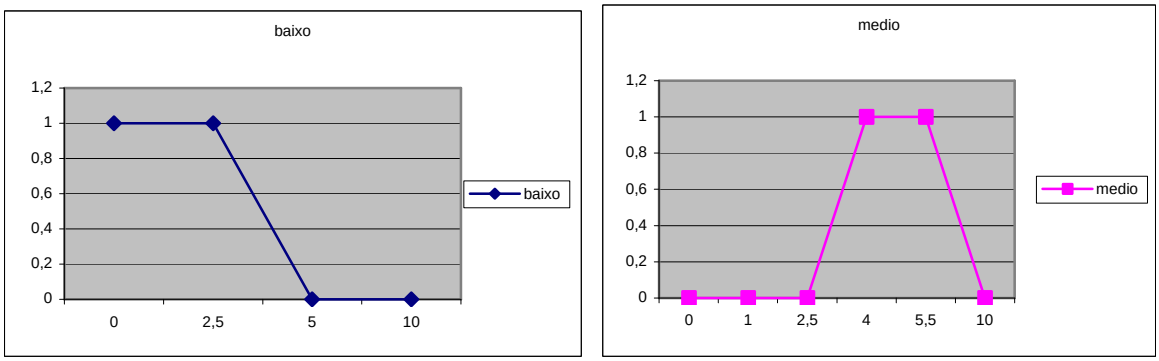


Figura 12 – A VARIÁVEL $CUSTO_{I,K}$ E SEUS TERMOS *FUZZY*



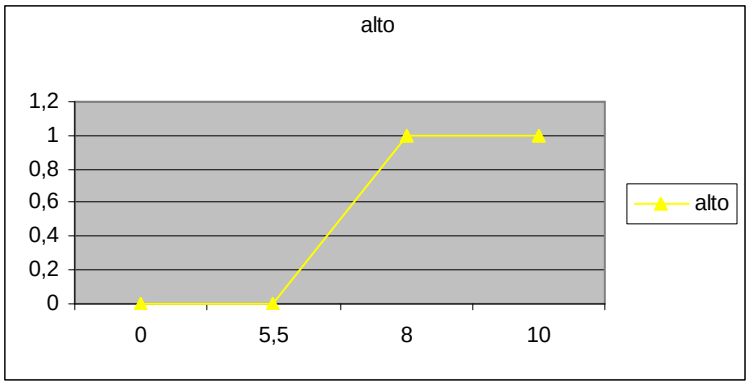


Figura 13 – A VARIÁVEL IC_K E SEUS TERMOS *FUZZY*

$IMS_{I,K}^{S}$ – Esta variável representa o índice de merecimento de um usuário em relação à sua melhor opção depois da principal. Os valores *fuzzy* são: BAIXO, MÉDIO e ALTO e o seu universo de discurso é de 0 até 1. Ele é a saída do SIF1 que servirá de entrada para o SIF2. Esta variável se divide em duas variáveis distintas quando se trata de resultado do SIF1 e quando se constitui de entrada do SIF2.

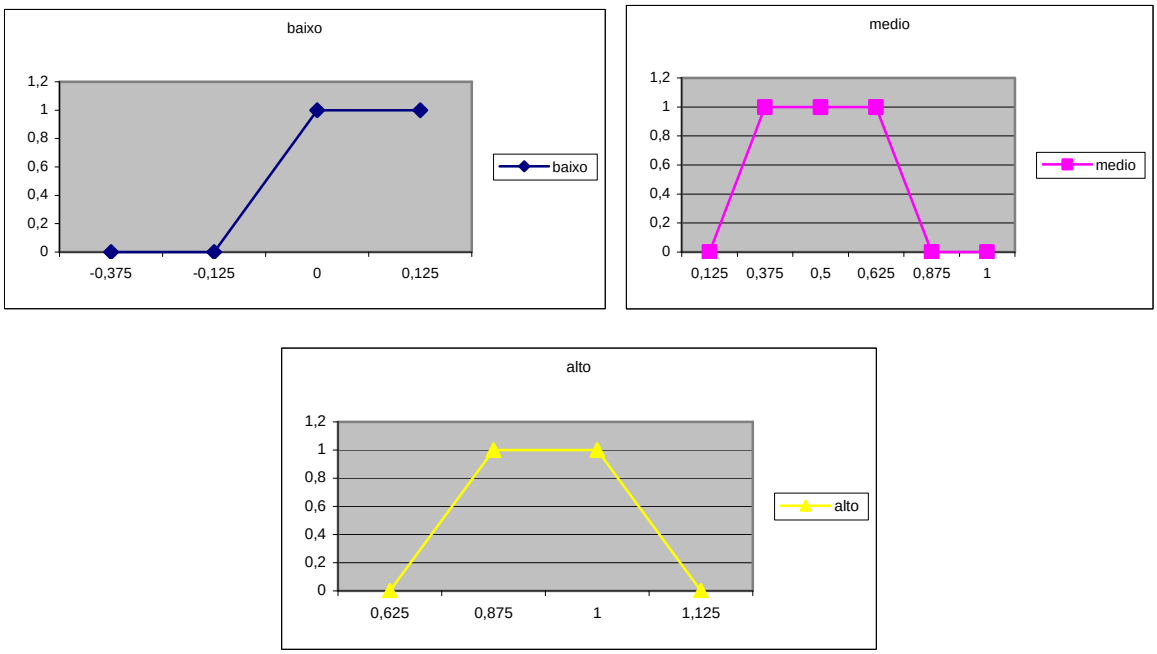


Figura 14 – A VARIÁVEL $IMS_{I,K}^{S}$ E SEUS TERMOS *FUZZY*

$IMS_{I,K}^{E}$ – Esta variável representa o índice de merecimento de um usuário em relação à sua melhor opção depois da principal. Os valores são: BAIXO, MÉDIO e ALTO e o seu universo de discurso é de 0 até 1. Conforme foi exposto em $IMS_{I,K}^{S}$, $IMS_{I,K}^{E}$ torna-se uma

outra variável quando for uma das entradas para o SIF2, muito embora tenha o mesmo universo de discurso do $IMS_{I,K}^{S}$.

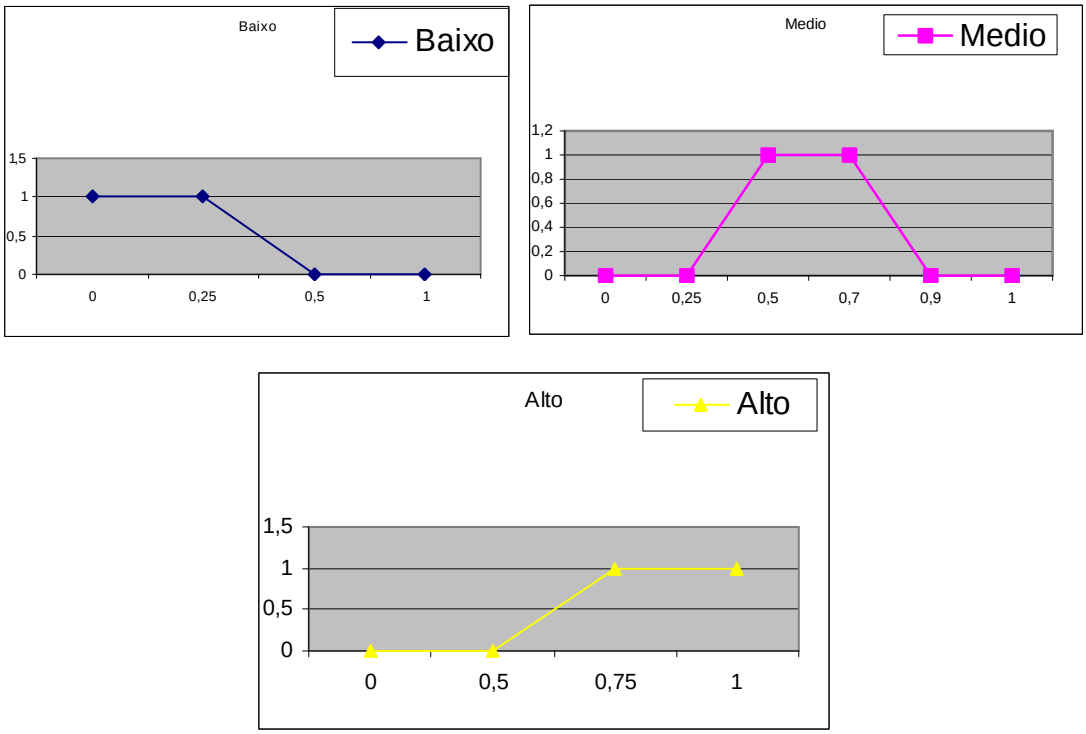
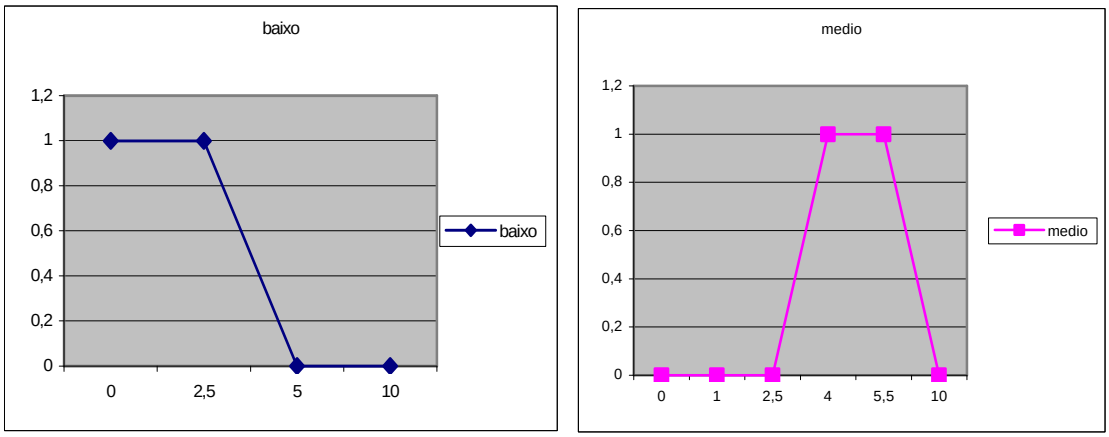


Figura 15 – A VARIÁVEL $IMS_{I,K}^{E}$ E SEUS TERMOS *FUZZY*

IC_J – O IC_J é o índice de competitividade do recurso que é a opção ótima do usuário em que se está fazendo as alocações. Ele é medido dividindo-se o total de usuários pela capacidade média desse recurso. Os valores (conjuntos *fuzzy*) desta variável são BAIXO, MÉDIO e ALTO e o seu universo de discurso é de 0 até o valor máximo desse índice que é o total de usuários que estão disputando recursos dividido pela capacidade média dos recursos, ou seja, a média aritmética da capacidade de todos os recursos envolvidos.



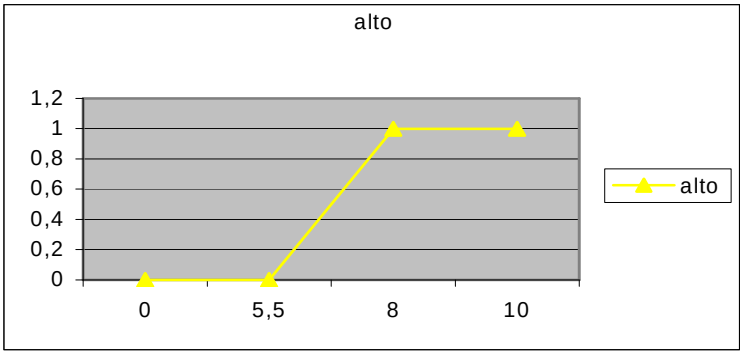


Figura 16 – A VARIÁVEL IC_J E SEUS TERMOS *FUZZY*

$IM_{I,J}$ – Esta variável representa o índice de merecimento de um usuário em relação à sua opção principal (ótima). Os seus valores *fuzzy* são BAIXO, MÉDIO e ALTO e o seu universo de discurso é de 0 até 1. Ele é a saída do SIF2, isto é, ele á a solução do problema de alocação do usuário ao seu melhor.

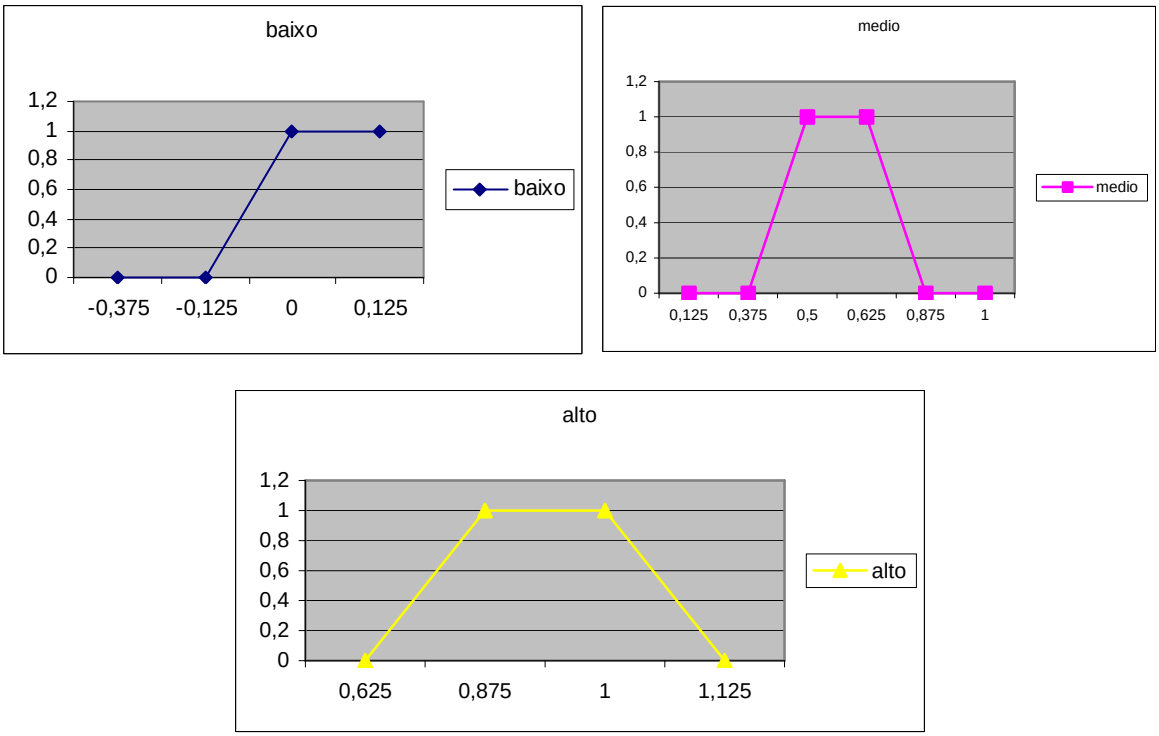


Figura 17 – A VARIÁVEL $IM_{I,J}$ E SEUS TERMOS *FUZZY*

$CUSTO_{I,J}$ – O custo é a distância do usuário a um recurso que é a melhor opção de alocação, ou seja, é a menor distância encontrada em relação à opção ótima (a que melhor resolve o problema da alocação para um dado usuário). Os valores dessa variável (conjuntos *fuzzy*) são ZERO, BAIXO, MÉDIO E ALTO e seu universo de discurso é de 0 até o Perímetro da Região do Mapa onde será analisada tal distância.

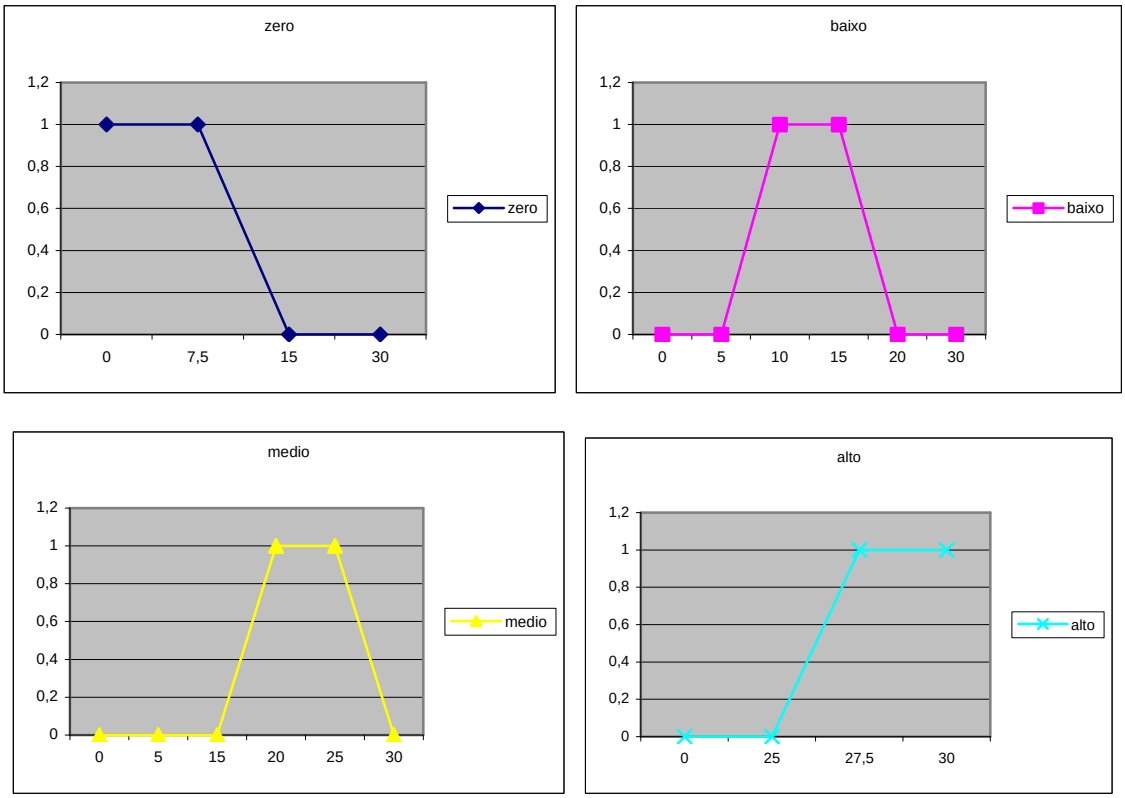


Figura 18 – A VARIÁVEL $CUSTO_{I,J}$ E SEUS TERMOS *FUZZY*

3.5.2 - Base De Regras Para O SIF Secundário Do Sistema Multiponto a Multiponto

No Sistema Secundário (SIF1) têm-se duas variáveis: Custo e Índice de Competitividade. O universo de discurso da variável lingüística de entrada Custo encontra-se dividido em 4 conjuntos *fuzzy*: zero, baixo, médio e alto. Já a entrada Índice de Competitividade está dividida em 3 conjuntos *fuzzy*: baixo, médio e alto. Então o número de regras é 12. As regras estão descritas na tabela FAM (*Fuzzy Associative Memory*) abaixo:

	CUSTO				ÍNDICE DE MERECIMENTO SECUNDÁRIO
	ZERO	BAIXO	MÉDIO	ALTO	
ÍNDICE DE COMPET	ALTO	ALTO	MÉDIO	MÉDIO	BAIXO
	MÉDIO	MÉDIO	MÉDIO	BAIXO	MÉDIO
	MÉDIO	MÉDIO	BAIXO	BAIXO	ALTO

BAIXO
MÉDIO
ALTO

Figura 19 – TABELA FAM DO SIF 1 (SISTEMA SECUNDÁRIO)

3.5.3 - Base De Regras Para O SIF Principal Do Sistema Multiponto A Multiponto

Em relação ao Sistema Principal (SIF2) têm-se três variáveis de entrada e uma de saída. A variável Custo possui 4 termos lingüísticos (conjuntos *fuzzy*), o Índice de Merecimento secundário, 3 termos e o Índice de Competitividade também 3 termos. Portanto o número de regras é 4 x 3 x 3 = 36 regras. Essas regras se encontram disponibilizadas em 3 tabelas FAM (*Fuzzy Associative Memory*), visto que esse SIF possui 3 entradas e, dessa forma, isola-se uma das variáveis de entrada para que dentro da mesma , para cada um de seus conjuntos *fuzzy* haja uma tabela FAM correspondente. A representação encontra-se na figura abaixo:

CUSTO

ÍNDICE DE COMPET	ZERO	BAIXO	MÉDIO	ALTO
BAIXO	ALTO	ALTO	ALTO	MÉDIO
MÉDIO	MÉDIO	ALTO	MÉDIO	MÉDIO
ALTO	MÉDIO	MÉDIO	MÉDIO	BAIXO

ÍNDICE DE MERECIMENTO SECUNDÁRIO: BAIXO

ÍNDICE DE MERECIMENTO PRINCIPAL: BAIXO, MÉDIO, ALTO

CUSTO

ÍNDICE DE COMPET	ZERO	BAIXO	MÉDIO	ALTO
BAIXO	ALTO	MÉDIO	MÉDIO	MÉDIO
MÉDIO	MÉDIO	MÉDIO	MÉDIO	BAIXO
ALTO	MÉDIO	MÉDIO	BAIXO	BAIXO

ÍNDICE DE MERECIMENTO SECUNDÁRIO: MÉDIO

ÍNDICE DE MERECIMENTO PRINCIPAL: BAIXO, MÉDIO, ALTO

CUSTO

ÍNDICE DE COMPET	ZERO	BAIXO	MÉDIO	ALTO
BAIXO	MÉDIO	MÉDIO	MÉDIO	MÉDIO
MÉDIO	BAIXO	MÉDIO	MÉDIO	BAIXO
ALTO	BAIXO	BAIXO	BAIXO	BAIXO

ÍNDICE DE MERECIMENTO SECUNDÁRIO: ALTO

ÍNDICE DE MERECIMENTO PRINCIPAL: BAIXO, MÉDIO, ALTO

Figura 20 – TABELAS FAM DO SIF 2 (SISTEMA PRINCIPAL)

3.6 - Comentários

Até aqui, procurou-se mostrar como um todo o que é a solução proposta para o Problema de Alocação Multiponto a Multiponto. Trata-se de um sistema de tomada de decisão que apresenta um componente da Inteligência Computacional, ou seja, a Lógica *Fuzzy*. No próximo capítulo será apresentada a implementação do protótipo.

CAPÍTULO 4 – IMPLEMENTAÇÃO DO PROTÓTIPO

4.1 – Introdução

Depois de apresentado o modelo lógico do protótipo (DFD, MER e o Modelo *Fuzzy*), neste capítulo será mostrado a sua implementação. Serão descritos o modelo físico do sistema protótipo de alocação multiponto a multiponto, além das suas interfaces componentes e os algoritmos de inferência *fuzzy* e de geração dos resultados da alocação *fuzzy*.

4.2 – Interfaces

O protótipo desenvolvido apresenta 6 módulos a saber: Cadastramento dos Usuários, Cadastramento dos Recursos, Cálculo das Distâncias entre Usuários e Recursos, Cálculo dos Índices de Merecimento, a Alocação Propriamente Dita e o Cálculo dos Índices de Competitividade. Há também a Tela Inicial.

4.2.1 – Tela Principal

É a apresentação inicial. Essa interface é composta de botões ao invés da organização por menus.

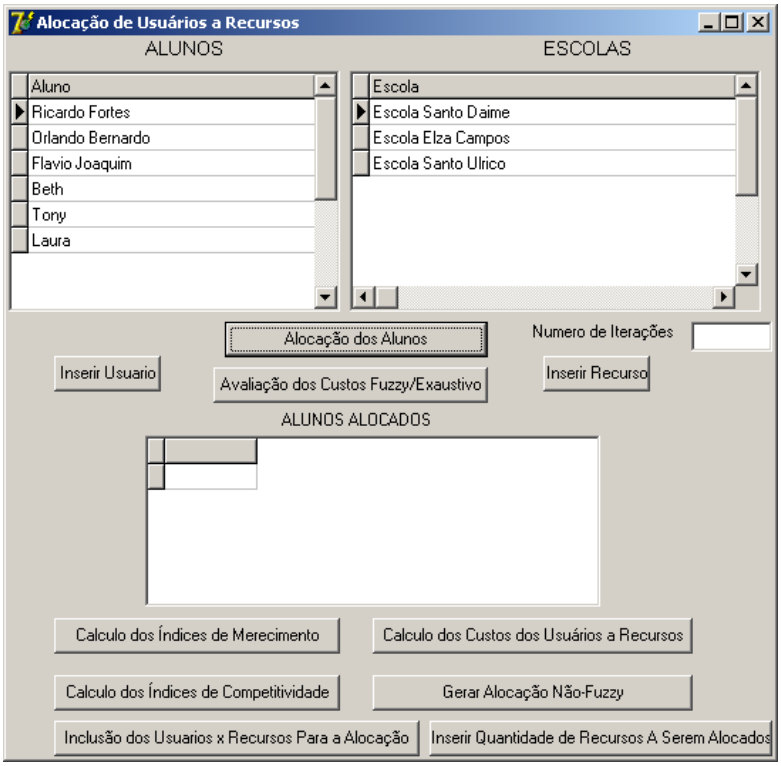


Figura 21 – TELA PRINCIPAL

Observa-se que aparecem todas as funções do protótipo nessa tela: os botões de Alocação de Alunos, Inserir Usuário, Inserir Recurso, Cálculo dos Índices de Merecimento, Cálculo dos Custos e Cálculo do Índice de Competitividade entre Usuários e Recursos.

4.2.2 - Cálculos Dos Custos Dos Usuários A Recursos

Este módulo calcula a distância entre cada usuário e cada recurso. Esse cálculo é feito com base em um sistema de coordenadas cartesianas que se supõe estar em um mapa de um determinado SIG. Cada usuário tem um par de coordenadas cartesianas nesse sistema, bem como o recurso.

4.2.3 - Cálculo Dos Índices De Competitividade

Aqui neste módulo, uma vez calculadas as distâncias entre os usuários e recursos, pega-se as distâncias calculadas e, dividindo-se pela capacidade de cada recurso, se encontram os índices de competitividade.

INSERIR USUÁRIO

É a tela de Cadastramento dos Usuários

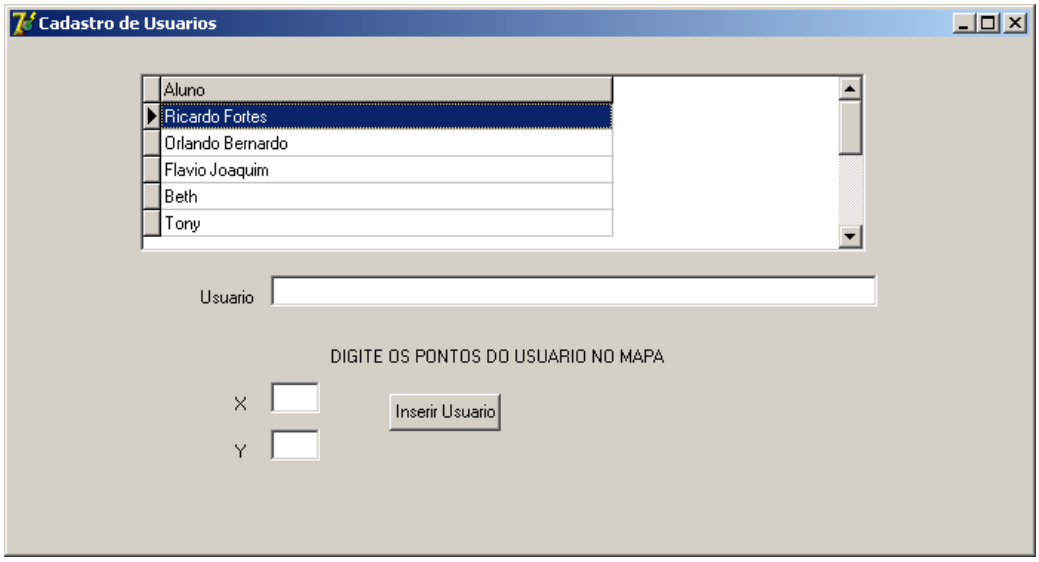


Figura 22 – CADASTRO DE USUÁRIOS

Observe que há os pontos de coordenadas X e Y a serem cadastrados.

INSERIR RECURSO

É a tela de Cadastramento dos Recursos. Observe que similarmente à tela de usuários, existem campos para preenchimento das coordenadas X e Y do Recurso.

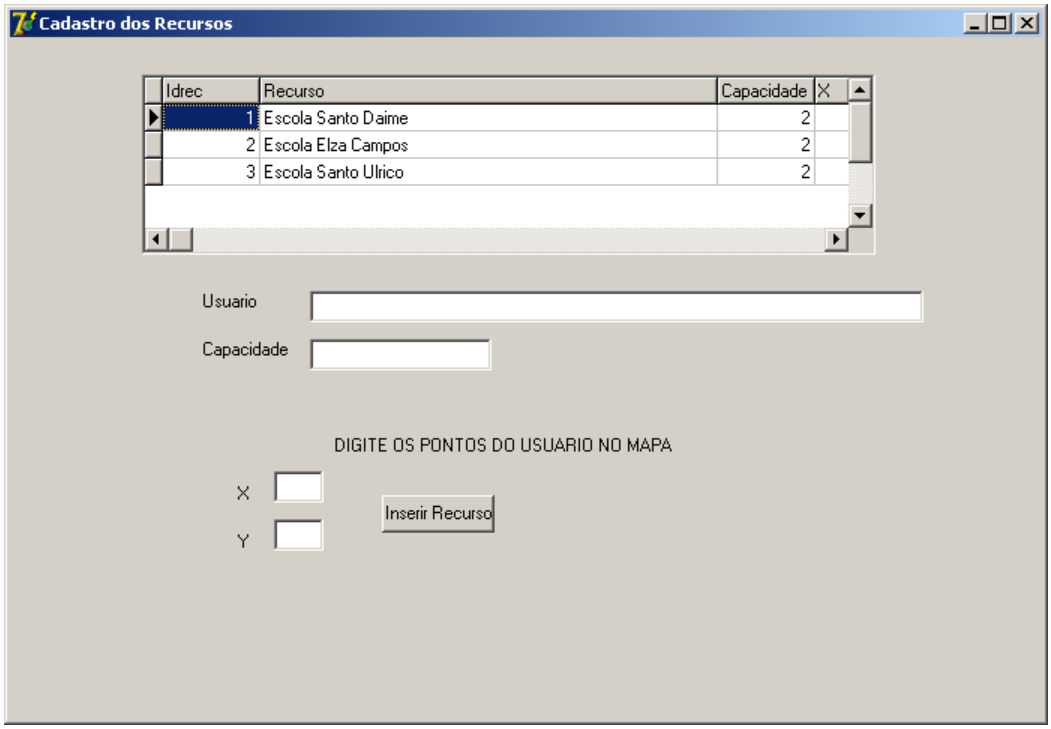


Figura 23– FUNÇÃO DE INSERÇÃO DE RECURSOS

4.2.4 - Cálculo Dos Índices De Merecimento

Neste módulo, está o coração do protótipo. É aqui neste módulo onde são calculados os índices com base no algortimo *fuzzy* para a alocação. Além disso, se calculam os índices que serão determinantes na alocação dos usuários dentro dos recursos disponíveis. Só depois do cálculo dos mesmos é que se inicia a rotina principal do protótipo que é a alocação propriamente. Na figura a seguir, o módulo desse cálculo que pode ser feito por um par usuário/recurso ou de todos de uma só vez.

Figura 24 – TELA DO CÁLCULO DOS ÍNDICES *FUZZY*

4.2.5 - Alocação Dos Alunos

Neste módulo, é executada a alocação dos alunos (usuários) nas escolas (recursos), pois, como exemplo de alocação multiponto a multiponto, foi adotado o caso de matrícula de alunos em escolas. É conveniente perceber que, se não for utilizado o Módulo do Sistema de Inferência *Fuzzy* antes disso, a geração dessa alocação torna-se totalmente imprecisa e sem sentido algum. Este módulo dependendo do número de usuários a serem alocados e recursos disponíveis, pode ser extremamente rápido, na ordem de poucos segundos.

4.3 – Projeto Físico

Neste tópico, será mostrada a estrutura de cada uma das tabelas que foram criadas para o protótipo a partir do modelo lógico explanado no Capítulo III. As tabelas que compõem essa estrutura de dados utilizada no protótipo criado são:

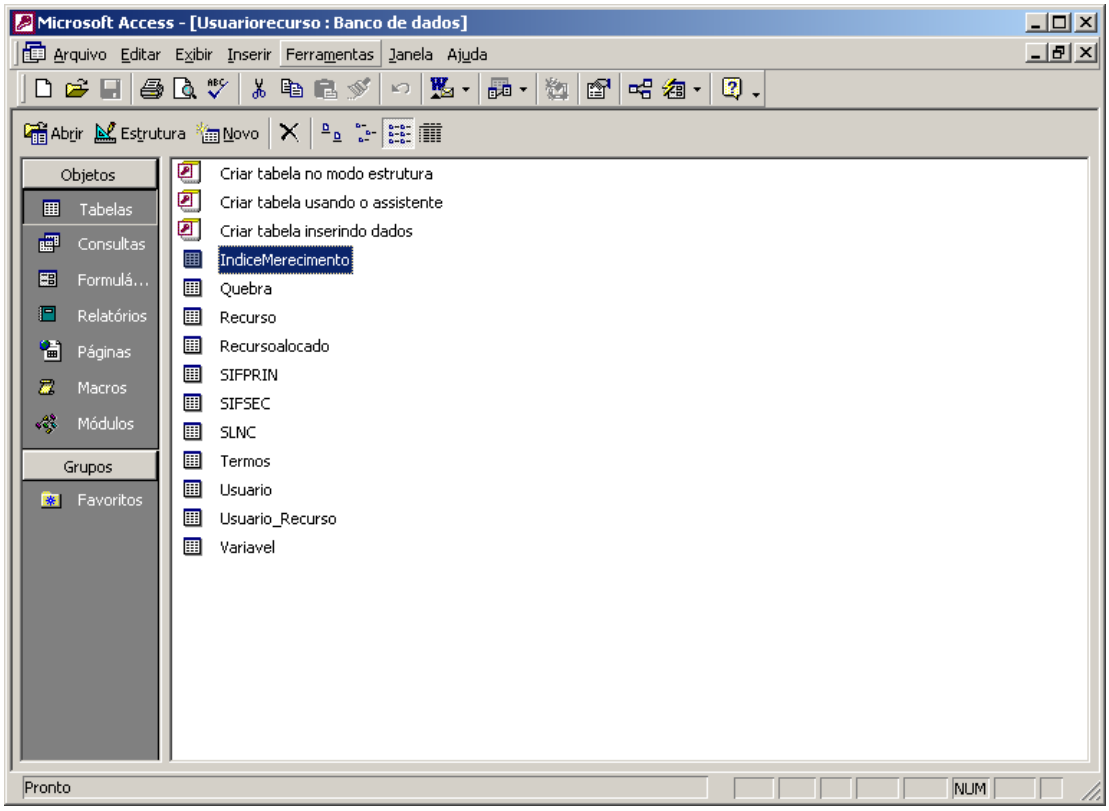


Figura 25 – TABELAS DO PROTÓTIPO

As tabelas do protótipo foram transformadas em um número maior por conta do sistema de inferência *fuzzy*, uma vez que para cálculo desses índices, necessita-se da base de regras e de todo o ferramental da Lógica *Fuzzy* como os pontos de quebra, variáveis lingüísticas e os termos (conjuntos *fuzzy*). Conseqüentemente foram montadas as tabelas Quebra, SIFSEC, SIFPRIN, Termos e Variáveis. A tabela IndiceMerecimento é uma das centrais do sistema. É nela que são armazenadas as informações dos Índices de Merecimento Primário e Secundário, além das Distâncias e Índices de Competitividade. Em outras palavras, em relação ao modelo lógico projetado, cuja preocupação era retratar a relação usuário/recurso, no projeto físico, aparecem as tabelas relativas ao SIF (Sistema de Inferência *Fuzzy* além das três já existentes no modelo lógico (Usuario, Recurso e Usuario_Recurso). A seguir a estrutura das tabelas.

Tabela USUARIO

(Idusu, Usuario, X, Y, Alocado)

É a tabela que armazena os usuários a serem alocados. O campo Idusu é o identificador do Usuário, o campo usuario é o usuário propriamente dito, o X e o Y são as coordenadas do

Usuário no espaço geográfico e Alocado é um campo *booleano* que indica se o usuário foi alocado ou não.

Tabela RECURSO

(IdRec, Recurso, Capacidade, X, Y)

É a tabela de Recursos que alocam os usuários no sistema. O campo IdRec é o identificador do recurso, o campo Recurso é o recurso propriamente dito, o X e o Y são as coordenadas do recurso no espaço geográfico e Capacidade é um campo que armazena a quantidade de usuários que podem ser alocados.

Tabela USUARIO_RECURSO

(Idusu,Idrec)

É a tabela onde se registram em quais recursos os usuários foram alocados. Naturalmente, Idusu é o identificador do usuário que está sendo colocado no recurso e o Idrec é o recurso onde o usuário está sendo incluído.

Tabela ÍNDICEMERECIMENTO

(Idusu,Idrec,indmerec,indmerecaux,icomp,custo)

Essa tabela é a mais importante do sistema. É nela que se armazenam as distâncias entre os usuários e os recursos, os índices de competitividade e os Índices *Fuzzy*, que são o principal fundamento de suporte para esse trabalho.

Idusu é o identificador do usuário, idrec é o identificador do recurso, indmerec é o índice de merecimento principal, indmerecaux é o índice de merecimento secundário, icomp é o índice de competitividade do recurso e o custo é a distância entre o usuário e o recurso.

Tabela SIFPRIN

(custoprin,indmerecaux,indcompprin,indmerecprin)

Essa tabela armazena o Sistema *Fuzzy* Principal. Custoprin é o custo principal, indmerecaux é o índice de merecimento secundário, indcompprin é o índice de competitividade e indmerecprin é o índice de merecimento principal.

Tabela SIFSEC

(custosec,indcompet,indmerecsec)

Essa tabela armazena o Sistema *Fuzzy* Secundário. Custosec é o custo secundário, indcompet é o índice de competitividade e indmerecsec é o índice de merecimento secundário.

Tabela QUEBRA

(CodVar, Codtermo,X,Y)

Tabela que armazena os pontos de quebra dos Termos *Fuzzy* inerentes a esse Protótipo

CodVar é o código da Variável linguística, Codtermo é o código do termo referente ao conjunto *fuzzy* e X e Y são os pontos de quebra.

Tabela VARIAVEL

(Codigo, Nome, Inicio, Fim, Antecedente)

Tabela que armazena as variáveis lingüísticas do Sistema *Fuzzy*. Codigo é o código da variável, Nome é o nome da variável, Inicio é o valor de X aonde ela começa e Fim é o limite dela, ou seja, onde ela termina e Antecedente é o campo que determina se ela é tem antecedente ou não.

Tabela TERMOS

(Codigo,CodVar,Nome,Npontos)

Tabela que armazena os conjuntos *fuzzy* que agem no Sistema *Fuzzy*. Codigo é o código do termo *Fuzzy*, CodVar é o código da variável lingüística a qual o termo pertence, Nome é o nome do conjunto *Fuzzy* e npontos é o número de pontos do Conjuntos *Fuzzy*.

Tabela RECURSOALOCADO

(Idrec, Alocacao)

Tabela auxiliar que faz a contagem de quantos usuários foram alocados em um determinado recurso. Auxilia no teste de quando o mesmo se encontra com a capacidade esgotada, a fim de se desalocar ou não um usuário deste recurso. O Idrec é o id do recurso e alocação é o campo que aponta quantos usuários estão alocados em um dado instante.

4.4 – O Algoritmo De Inferência *Fuzzy*

O algoritmo de Inferência *Fuzzy* utilizado no Protótipo de Sistema de Alocação Multiusuário envolve tanto o cálculo do par usuário/recurso quanto o cálculo de todos os

índices de todos os pares usuário/recurso. Da forma que esse aplicativo foi estruturado como mostrado no tópico de interface do sistema, todos os dois cálculos citados chamam um procedimento de nome calcula_inferencia, onde são ativados todos os processos de inferência tais como qual o conjunto *fuzzy* em que se inserem os valores passados de acordo com a base de regras projetada, a partir daí se calcula de ativação da regra, e depois se parte param a rotina de *defuzzificação* que aqui nesse aplicativo é utilizado o Método do Centro de Área (Centróide).

Na verdade, o funcionamento desse algoritmo pode ser descrito da seguinte forma: primeiro ele calcula o índice de merecimento secundário onde para o seu cálculo são utilizados o índice de competitividade e o custo (distância entre o usuário e o recurso onde aquele será alocado). Então, nesse primeiro momento, utiliza-se um primeiro sistema de inferência *fuzzy* para inferir o Índice de Merecimento Secundário, isto é, o Índice para a segunda melhor opção para o Usuário.

Uma vez feita a inferência para o índice secundário, o algoritmo parte para a segunda fase que é o Índice de Merecimento para o recurso que determinará para que recurso irá o usuário. Então, tem-se um segundo SIF para se fazer o cálculo desse índice. Agora, nessa segunda fase o algoritmo levará em conta três parâmetros: a distância entre usuário e recurso, o índice secundário e o índice de competitividade do recurso. Ao se cruzar esses valores, o algoritmo chama esse segundo SIF e faz a inferência, efetuando dessa forma o cálculo do Índice de Merecimento Principal objetivo fundamental para o funcionamento do protótipo proposto.

Esse algoritmo está implementado nos botões **Inferir Índice Inicial** e **Gerar Todos os Índices de Merecimento** respectivamente. Evidentemente que nesse protótipo, o usuário pode gerar os índices *fuzzy* das duas maneiras; em geral de acordo com essa proposta de dissertação. A idéia é que o usuário calcule esses índices eventualmente e não a todo o instante, uma vez que não será a todo o momento que um usuário e/ou recurso será cadastrado. Então se for apenas a um usuário novo, pode-se apenas fazer um único cálculo e nada além. Em outras palavras, o algoritmo de inferência *fuzzy* na verdade engloba dois sub-algoritmos que são interdependentes, ou seja, existe um parâmetro aplicado em um que depende intrinsecamente do outro algoritmo, para ser calculado. Os detalhes da implementação do algoritmo de inferência encontram-se no Apêndice B e logo a seguir ele é apresentado na forma de pseudocódigo.

```
inicio
para  I = 0 até indicemaisalto(imerec) do
inicio
     Abre SifSecundario
     valor = InfereSLN('SifSecundario',imerec[i].custo, imerec[i].icomp, 0)
     s= Converterealparacaracter(valor)
     imerec[i].indmerecaux= Convertercaracterparareal (s)
     Abre Sifprincipal
     valor := ('Sifprincipal',imerec[i].custo, imerec[i].icomp,
imerec[i].indmerecaux)
     s = Converterealparacaracter(valor)
     imerec[i].indmerec:= Convertercaracterparareal (s)
fim
fim
```

4.5 – Algoritmo De Geração Dos Resultados

Aqui nesta seção, tratar-se-á do algoritmo que gera a alocação dos usuários dentro dos recursos disponíveis. Ao se descrever este algoritmo, antes de mais, nada é preciso levar em conta que inicialmente dois laços obrigatoriamente são criados. Os dois laços em questão vão percorrer toda a tabela usuário e a tabela de índices de merecimento, uma vez que os usuários deverão ser alocados e, dentro do laço da tabela de usuários, fazer o laço que percorre a tabela de índices de merecimento para se verificar em qual recurso o algoritmo irá alocar esse usuário (ver o algoritmo em pseudocódigo a seguir).

```
Procedimento Alocação
Inicio
 Zeraalocacao
 Primeirousuario
 Para i:= 1 até fim {Rotina de Rodar os Usuarios a serem alocados}
  inicio
   Fecha Consultaaosindices
   Consultaaosindices.Parametro('usu').valor = identusuario
   Abre Consultaaosindices
   Primeiroindice
   Fecha recursosalocados
   recursosalocados.parametero('rec').value:= identrecursosalocados;
   abre recursosalocados
   aloc:= recursosalocadoscalocacao
   Se ( aloc < capacidaderecurso) entao     {Aqui procede-se à Alocação}
   inicio
       Insereusuario
       aloc := aloc + 1;
       alteraalocacao.Parametro ('idrec').Value := identrecursosalocados
       abre alteraalocacao
   fim
   senao inicio      {Aqui é a rotina de Desalocação caso a capacidade do
recurso esteja esgotada}
      Fecha listausuariosrecursos
      Listausuariosrecursos..Parametro('rec') = identrecursosalocados
      abre listausuariosrecursos
```

```
      Primeiro listausuariosrecursos
      maior:= Consultaaosindicesindmerecimento;
      valor:= listausuariosrecursosindmerecimento
      Se  maior > valor  entao
        Remanejausuario.Parametro('usu'):= identusuario
        Remanejausuario.Parametro ('rec') := identrecurso
        Remanejausuario.Parametro ('id').Value := identrecursosalocados
        Remanejausuario
        Desalocausuario.Parametro('usu').Value:= identrecursosalocados
        Desalocausuario
      fim
       senao    {Caso nao aconteça a Desalocação, tenta-se a segunda opção
de Alocação em Caso de Recurso já esgotado}
       proximoindice
       Para j de 1 to totalregistrosConsultaindices faca
        inicio

          Fecha RecursosAlocados
          RecursosAlocados.Parametero('rec').value:=
Consultaindicesidrecurso
          Abre RecursosAlocados
          aloc:= RecursosAlocados.numalocados
          Se  ( aloc < CapacidadeRecurso) entao
          inicio
               Insereusuario
              aloc := aloc + 1;
              alteraalocacao.Parametro ('idrec').Value  =
identrecursosalocados
              abre alteraalocacao
              atualizaalocacao.Parametro('aloc') = aloc + 1
              atualizaalocacao.Parametro('idrec')= Consultaindicesidrecurso
              abre atualizaalocacao
              alteranumeroalocados.Parametro('usu').Value:=
listausuariosrecursosindusu
              abre alteranumeroalocados
          fim se
          senao
            proximoindice

          fim para
        fim se

      fim se
   fim se
   atualizaalocacao.Parametro('aloc') = aloc + 1
   atualizaalocacao.Parametro('idrec')= Consultaindicesidrecurso
   abre atualizaalocacao
   alteranumeroalocados.Parametro('usu').Value:=
listausuariosrecursosindusu
   proximo usuario
  Fim para

  {Aqui é a rotina de Realocação de Usuarios na segunda ou última melhor
opção,que ficaram desalocados}
 Fecha Realocacao
 Abre Realocacao
 PrimeiroRealocacao
 Para I = 1 to TotalRegistrosRealocacao do
   inicio
    Fecha Consultaaosindices
    Consultaaosindices.Parametro('usu').valor = identusuario
```

```
    Abre Consultaaosindices
    Para j = 1 ate TotalRegistrosConsultaaosindices faca
        inicio
          Fecha recursosalocados
          recursosalocados.parametero('rec').value:= identrecursosalocados
         abre recursosalocados
         aloc:= recursosalocadoscalocacao
         Se ( aloc < capacidade) entao
          inicio
              Se ( aloc < capacidaderecurso) entao     {Aqui procede-se à
Alocação}
               inicio
                 Insereusuario
                 aloc := aloc + 1
                 alteraalocacao.Parametro ('idrec').Value  =
identrecursosalocados
                 abre alteraalocacao
                 atualizaalocacao.Parametro('aloc') = aloc + 1
                 atualizaalocacao.Parametro('idrec')=
Consultaindicesidrecurso
                 abre atualizaalocacao
                 alteranumeroalocados.Parametro('usu').Value:=
listausuariosrecursosindusu
                 abre alteranumeroalocados
              fim se
          fim se
          senao
             proximoindice
         fim para
   proximo atualizaalocacao
   fim para
  Abre usuariosalocados
fim
```

A questão é saber qual dos laços será o principal. Nesta proposta, é utilizado o laço usuário como o laço principal, pois como os mesmos serão alocados, então são percorridos todos os usuários para se averiguar em que recurso ele ficará. Para isso, é necessário fazer a varredura nos índices para se validar qual deles é o de melhor alocação para aquele usuário. Porém, não é apenas pura e simplesmente aferir o melhor índice e alocar o usuário em um determinado recurso.

Além de verificar o melhor índice, este algoritmo deve também testar se um dado recurso encontra-se ocupado ou não; então ele faz o teste de verificação para constar se o recurso encontra-se com a sua capacidade preenchida ou não. Se o mesmo ainda estiver com vagas, o algoritmo o aloca normalmente ali, senão caso a melhor opção desse usuário seja mesmo esse recurso, ele busca o índice de merecimento desse usuário em relação a esse recurso, compara com o menor índice de merecimento encontrado nos usuários já alocados. Então, o algoritmo testa se o índice daquele usuário é maior do que este já alocado. Se for maior, aquele usuário é alocado neste recurso e o usuário já alocado é desalojado desse

recurso e realocado na segunda melhor opção de recurso; senão o mesmo é alocado na sua segunda melhor opção.

Observe que antes do laço que percorre a tabela usuário começar, sempre se limpa a tabela de alocação antes e o protótipo ao iniciar sua execução, são zeradas as tabelas auxiliares que fazem o teste de verificação se o recurso encontra-se ocupado ou não.

4.6 – Algoritmo Exaustivo

Como será descrito a seguir, este algoritmo é do tipo NP (Ordem Fatorial), não tendo, portanto, uma solução em tempo polinomial. O problema consiste em se alocar ***n*** usuários a ***n*** recursos. Esse algoritmo envolve Matemática Combinatória, pois inclui árvore de possibilidades, uma vez que, teoricamente, um usuário pode ser alocado em qualquer recurso. Logo, abre-se uma árvore de possibilidades para cada usuário. O raciocínio desse algoritmo parte da premissa que todos os usuários sejam alocados. Assim, se terá um sem número de combinações que resultarão em um processamento longo, o que torna o algoritmo de difícil execução.

Exemplos:

1) Considere a possibilidade de tentativa de alocação de 150 alunos em 5 escolas. Essas escolas têm respectivamente 30, 40, 20, 50 e 10 vagas para se fazer as respectivas alocações.

Nesse caso então:

Para E1, na primeira escola, tem-se 150 alunos e 30 vagas. O número de alocações possíveis seria $C_{150,30}$.

Para E2, na segunda escola, tem-se 120 alunos para 40 vagas. O número de alocações possíveis $C_{120,40}$

Para E3, sobram 80 alunos a serem alocados em 20 vagas, ou seja, ter-se-ia $C_{80,20}$.

Para E4, sobram 60 alunos para serem alocados em 50 vagas. Então tem-se $C_{60,50}$.

Para E5, ficam 10 alunos para serem alocados nas 10 vagas restantes. As possibilidades se resumem a 1, ou seja, tem-se $C_{10,10} = 1$

Como essas alocações são eventos independentes uns dos outros, o número total de possibilidades de alocação dos 150 alunos nas cinco escolas é:

$N = C_{150,30} \times C_{120,40} \times C_{80,20} \times C_{60,50} \times 1$

No caso $C_{10,10} = 1$.

Esse número estaria na casa de aproximadamente 120!, uma vez que se essa expressão fosse desenvolvida, ter-se-ia algo desse tipo:

150!/120! 30! x 120!/ 80! 40! X 80!/20!60! x 60!/50! 10!

2) Agora, considere a possibilidade de tentativa de alocação de 10 alunos em 4 escolas. Essas escolas têm respectivamente 3, 3, 2 e 2 vagas para se fazer as respectivas alocações.

Nesse caso então:

Para E1, na primeira escola tem-se 10 alunos e 3 vagas. O número de alocações possíveis seria $C_{10,3}$.

Para E2, na segunda escola tem-se 7 alunos para 3 vagas. O número de alocações possíveis $C_{7,3}$

Para E3, sobram 4 alunos a serem alocados em 2 vagas, ou seja seria $C_{4,2}$.

Para E4, sobram 2 alunos para serem alocados nas 2 vagas restantes. Então tem-se $C_{2,2}$. As possibilidades se resumem a 1, ou seja, tem-se $C_{2,2} = 1$.

Como essas alocações são eventos independentes uns dos outros, o número total de possibilidades de alocação dos 10 alunos nas quatro escolas é:

$N = C_{10,3} \times C_{7,3} \times C_{4,2} \times C_{2,2}$

No caso $C_{2,2} = 1$.

Esse número estaria na casa de aproximadamente 7!, 8!, pois:

10!/7! 3! x 7!/ 3! 4! X 4!/2!2! x 1

Aqui será estipulado o caso geral.

Considere k_1, k_2,..., k_n recursos em que k_1 possua v_1 vagas, k_2 possua v_2 vagas até k_m que possui v_m vagas. O número de usuários é ***n***. Então, partindo do raciocínio da árvore de possibilidades a fórmula de cálculo do número dessas possibilidades fica da seguinte maneira:

$N = C_{n,v1} \times C_{n-v1,v2} \times C_{n-v1-v2,v3} \times .... C_{n-v1-v2 -...-vn,vn}$

Sendo que $n - v_1 - v_2 - \ldots - v_n$ seria o número v_m das últimas vagas a serem alocadas.

Tendo em vista a fórmula encontrada, percebe-se que mesmo usando grafos e tentando se achar o caminho mínimo dos mesmos nota-se que haverá um número muito alto de iterações, o que torna o processamento desse algoritmo muito custoso.

Então, conclui-se pelo exposto, que a complexidade desse algoritmo é do tipo NP, onde não existe solução em tempo polinomial para o mesmo.

4.7 – Algoritmo Ingênuo

Este algoritmo consiste em fazer alocação com base no menor custo (distância). O procedimento adotado consiste em se percorrer a lista dos usuários a serem alocados e a cada iteração verifica-se qual o recurso mais próximo àquele usuário. Feito isto, o usuário é alocado nesse recurso. Em outras palavras, nesse método podem acontecer injustiças preocupantes como alocar um usuário a um recurso muito mais distante do que aquele mais próximo. Neste método o usuário é alocado ao recurso mais próximo e, pelo fato de não haver uma ordem que otimize a alocação, acontecem resultados muito díspares em relação a esse algoritmo e, em função dos mesmos, percebe-se que no Algoritmo Ingênuo há uma dependência na ordem de entrada, ao se percorrer a lista dos usuários candidatos aos recursos; logo, para se reduzir a disparidade e, por conseguinte, obter-se um desempenho melhor do

algoritmo, é preciso se encontrar a melhor ordem possível para se percorrer a lista dos Usuários. Esse algoritmo teve utilizações em trabalhos anteriores, como (FADEL, 2003).

4.8 – Comentários

Neste tópico foram apresentados os Algoritmos *Fuzzy* e Ingênuo e de que forma foi feita a implementação do protótipo do Sistema de Alocação Multiponto a Multiponto. Foi apresentado o algoritmo de alocação propriamente dito, além do projeto físico das tabelas inerentes ao protótipo e as interfaces que são fundamentais para o funcionamento do mesmo. Mais adiante, no Capítulo de Resultados, poder-se-á perceber o quanto esses algoritmos foram satisfatórios nos testes em termos dessa proposta usando a Lógica *Fuzzy*.

CAPÍTULO 5 – ESTUDOS DE CASOS

5.1 – Introdução

Neste trabalho realizado que culminou com a criação do protótipo do Sistema de Alocação Multiponto a Multiponto, utilizou-se como estudo de caso a dissertação de (FADEL, 2003) que trata de um sistema que auxilia a matrícula de alunos em escolas públicas, no Município de Nova Iguaçu. Este sistema foi o adotado por ser um caso que se encaixa no ensaio proposto, pois existe uma demanda x de alunos para serem matriculados em um número de escolas públicas. O trabalho de (FADEL, 2003) tratava apenas da matrícula do aluno em si, não levando em conta as opções que todos os alunos tinham; em outras palavras, o Sistema de Gestão de Matrícula de Escolas Públicas matriculava o primeiro aluno da fila, não importando se os outros alunos seguintes tinham aquela escola como a melhor opção por conta da distância; os mesmos ficavam de fora.

O protótipo de alocação do presente trabalho já leva em conta esse critério, tanto é que se a melhor opção para um dado aluno for mesmo a escola X, ele tenta alocar aquele aluno, mesmo que precise desalojar outro já matriculado e realocar esse aluno em outra escola, ou seja, esta proposta procura ser mais justa. O critério utilizado é o da menor distância do aluno à escola. Além disso, serão apresentados os resultados encontrados pelo protótipo e a comparação do Modelo *Fuzzy* proposto com o Método Exaustivo.

5.2 – Resultados Do Sistema De Inferência *Fuzzy*

O estudo de caso adotado neste capítulo inspirou seis situações em que foram feitos os testes para essa Proposta utilizar-se-á como citado no início, um Sistema baseado no Problema de alocar alunos em Escolas no Municipio de Nova Iguaçu. Esses testes serão descritos a seguir:

1ª Situação: 35 alunos, 9 escolas com 5 alunos, capacidade: 45 alunos
2ª Situação: 45 alunos, 9 escolas com 5 alunos, capacidade: 45 alunos
3ª Situação: 50 alunos, 9 escolas com 5 vagas, capacidade: 45 alunos
4ª Situação: 70 alunos,11 escolas com 5 vagas, capacidade: 55 alunos
5ª Situação: 90 alunos, 12 escolas com 5 vagas, capacidade: 60 alunos

6ª Situação: 100 alunos,12 escolas com 5 vagas, capacidade: 60 alunos

No primeiro caso, o sistema levou um tempo de 2 s e 46 centésimos para alocar os 35 alunos; no segundo caso 04 s 58 centésimos; no terceiro 04 s e 76 centésimos; no quarto caso já levou um pouco mais de tempo 08 s e 97 centésimos ainda assim abaixo de 10 s; no quinto caso 14 s e 19 centésimos e no último caso 16s e 09 centésimos.

Como pode-se depreender dos ensaios feitos o tempo levado para alocação dos alunos nas escolas foi um tempo bem pequeno. Estes ensaios foram efetuados *stand alone* em um PC Athlon 1.2 Mhz com 512MB de Memória RAM.

Todavia, é preciso lembrar que o problema de Alocação Multiusuário teoricamente tem como solução óbvia, pela abordagem da Força Bruta (Busca Exaustiva), gerar todas as combinações possíveis para cada alocação de uma escola por vários alunos. Só que, primeiro se gera todas as combinações possíveis para em seguida, de acordo com o menor custo total qual é a combinação mais adequada à situação proposta. Então foi feito um outro ensaio com uma base menor, onde temos 4 alunos e 2 escolas, onde cada escola tem 2 vagas para matrícula. Foram efetuados os testes com as duas abordagens, tanto a abordagem pela Busca Exaustiva (Combinações) como pela abordagem proposta que utiliza a Lógica *Fuzzy* por detrás da solução.

O resultado alcançado pelas duas abordagens foi praticamente o mesmo. O tempo que cada solução teve em seu desempenho foi da ordem de menos de 2 segundos. Logo todas as duas soluções têm a mesma eficiência quando testadas em bases que possuem uma pequena quantidade de alunos e escolas. Mas e quando a abordagem precisa ser testada com uma quantidade de usuários maior, como por exemplo, com 35 usuários? Tem-se, então, a seguinte constatação: em termos de Busca Exaustiva, considerando a população mundial em torno de 6 bilhões de pessoas com computadores processando informações a 1 microsegundo em um século (100 anos), a capacidade máxima de processamento de informações seria calculada da seguinte forma:

$CAPMAXIMA=100x365x24x60x60x10^{6}x6x10^{9} = 189216000000000000000000000$, ou na notação científica seria 1.89216×10^{25}

Logo, este resultado é da ordem de grandeza 10^{25}. Esse número já seria inferior a 30!. Em outras palavras, isso significaria dizer que para um número de usuários superior ou igual a

30, esse procedimento superaria a capacidade máxima suportada por todos os computadores, levando-se em conta que cada pessoa no mundo possuísse o seu próprio equipamento.

Diante do exposto, pode-se concluir que a Busca Exaustiva a partir de um Número de Usuários superior a 30 inviabiliza a Solução da Alocação Multiusuário através da mesma. Para a comprovação do fato, foi feita a tentativa de se alocar 50 alunos em 9 escolas cada uma com 5 vagas para cada aluno. Foi tentada a Busca Exaustiva através da Geração de Todas as Combinações possíveis. Sabe-se que o número total no final do processamento será igual a algo como:

$$\text{Total} = C_{50,5} \times C_{45,5} \times C_{40,5} \times C_{35,5} \times C_{30,5} \times C_{25,5} \times C_{20,5} \times C_{15,5} \times C_{10,5} \times C_{5,5} .$$

Só na primeira combinação, $C_{50,5}$, temos o total de 2118760 combinações possíveis. Só para selecionar os alunos que entrariam na primeira escola o tempo estimado foi em torno de 4 horas e 10 minutos e, ainda assim não foram geradas todas as 2118760 combinações possíveis. E foi criado um arquivo de 256 MB com as combinações até então geradas! Imagine se fosse feita a tentativa de listar as combinações seguintes. Levar-se-iam dias para a geração de todas essas combinações. E surgiria um arquivo gigantesco da ordem de , no mínimo uns 14 GB. Na verdade seria preciso um supercomputador bastante eficiente e com grande capacidade de processamento e memória para a busca em cima desse arquivo para selecionar o menor custo total e a Combinação que atendesse a cada escola.

Na alternativa que aqui se propõe utilizando Lógica *Fuzzy*, ao se fazer a tentativa de alocação dos mesmos 50 alunos nas 9 escolas disponíveis, ainda que se passe um intervalo de 23 minutos e 25 segundos para a geração dos Índices *Fuzzy*, o processamento da alocação leva 04 segundos e 44 centésimos de segundo. E com um detalhe: o arquivo gerado não chega sequer a 1 MB de tamanho para gerir tal alocação.

Diante desses testes, pode-se concluir que a abordagem através da Lógica *Fuzzy*, além de um processamento bem enxuto, leva a uma solução dessa alocação próxima da solução ótima. Tanto é que no experimento com aquela base simples com 4 alunos e 2 escolas, a solução foi exatamente a mesma.

A Busca Exaustiva depende de vários fatores, porém o principal é a velocidade de processamento, pois ele recorre invariavelmente à memória uma vez que ele utiliza fortemente a recursividade como solução e isso demanda intensivo processamento. Só para se processar para os 50 alunos, somente se existisse um computador 1000 vezes mais rápido do que os supercomputadores atuais talvez o processamento só para a primeira escola poderia

terminar em uma hora. Só para se ter uma idéia do quão é inviável a Busca Exaustiva nos dias atuais se fosse necessário alocar 59 alunos nas 9 escolas com 5 vagas cada uma, poder-se-ia levar o período de um ano (365 dias) ao se listar todas as combinações possiveis e selecionar as mesmas na tentativa de matricular os alunos nas respectivas escolas.

Dessa forma, a Proposta de Alocação dos Alunos nas Escolas utilizando Lógica *Fuzzy* pelos testes feitos , mostrou ser uma estratégia mais enxuta e porquê não dizer, apresenta uma Solução que se não é muito próxima do ideal, se aproxima do mesmo. O único problema que detectado é quando é preciso calcular todos os Índices *Fuzzy* para um par Aluno/Escola, mas o usuário do Sistema pode optar de, ao invés de calcular todos os índices de uma vez só, calcular os que julgar necessários em um dado contexto que a situação exija, como por exemplo calcular os índices de Merecimento de três escolas para um par de 4 usuários, aí o cálculo dos Índices fica bem mais rápido e não se precisa levar 23 minutos ou até mais de 1 hora calculando-se estes índices. No Método Exaustivo não se tem essa opção, é fundamental o cálculo e a apresentação de todas as Combinações possíveis.

A abordagem do problema multiusuário através da Inteligência Computacional é uma estratégia que se tornou algo bastante palpável no sentido de que o conceito *Fuzzy* de grau de pertinência se ajusta ao modo de como se trata essa alocação de recursos, tendo em vista que quando o ser humano tenta adotar um modelo de alocar algum aluno em alguma escola, geralmente é utilizado um raciocínio aproximado para poder ajustar a situação de acordo com o sistema, ou seja, de acordo com o Modelo de Matrícula projetado por essa escola, por exemplo.

5.3 - Comparação Do Método *Fuzzy* Com O Método Exaustivo

Aqui nesta seção serão mostrados inúmeros ensaios feitos com os Métodos *Fuzzy* e Exaustivo a fim de se medir a viabilidade do Sistema Especialista que é a proposta desse trabalho. Ter-se-ão gráficos visando a comparar os resultados obtidos pelos dois métodos através dos Custos encontrados nos dois métodos.

Em todos os ensaios que serão descritos a seguir cabe citar dois aspectos relevantes para se compreender o funcionamento dos gráficos obtidos nestes:

1º) todos os ensaios foram realizados dentro de três espaços geográficos com dimensões 10 x 10, 100 x 100 e 1000 x 1000. A justificativa desse procedimento foi abranger vários testes em espaços geográficos distintos para se tentar ilustrar de forma objetiva e clara o desempenho do método apresentado nesse trabalho em relação a outras técnicas conhecidas.

Nesses gráficos o eixo das abscissas corresponde ao número de iterações realizadas e o eixo das ordenadas corresponde ao custo, sendo que é importante ressaltar que esse custo é o **custo total aferido de todos os usuários a todos os recursos em cada iteração**. Além disso, para cada iteração corresponde uma distribuição geográfica diferente no espaço geográfico compreendido, atentando para o detalhe de que tais distribuições geográficas são sorteadas, ou seja, são aleatórias.

2º) Para que os ensaios fossem viáveis do ponto de vista da Busca Exaustiva era necessário escolher um problema pequeno, pois o cálculo da solução ótima pela Busca Exaustiva seria muito extenso e impraticável com um problema um pouco maior do que o escolhido para os ensaios feitos. Sendo assim, os experimentos criados foram realizados em um conjunto de 6 usuários e 3 recursos, onde cada recurso possui 2 vagas. Ao empregar o cálculo da alocação multiponto a multiponto em um problema de pequeno porte, torna-se possível executar a Busca Exaustiva para ele em diversas situações de posicionamento de usuários e recursos diferentes para obter a solução ótima e compará-la com a solução obtida pelo Método *Fuzzy*.

Em relação ao segundo gráfico comparativo entre os custos obtidos pela Busca Exaustiva e os custos obtidos pelo Método *Fuzzy*, a fórmula que obtém a razão entre os dois custos é expressa da seguinte forma:

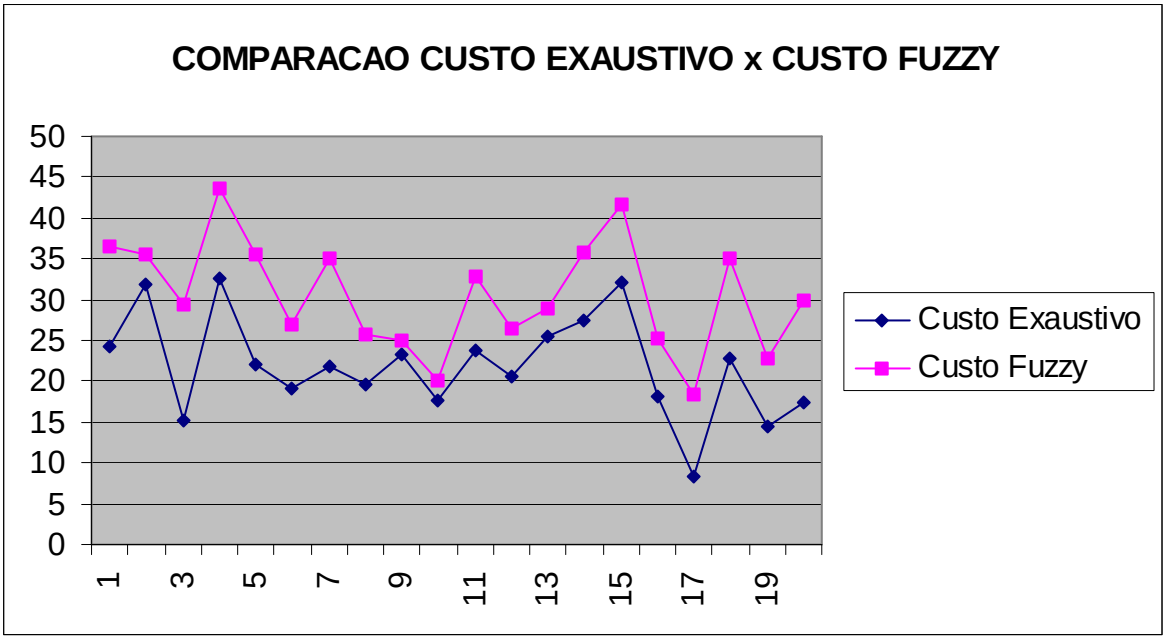
Indice = Custo Exaustivo /Custo *Fuzzy*

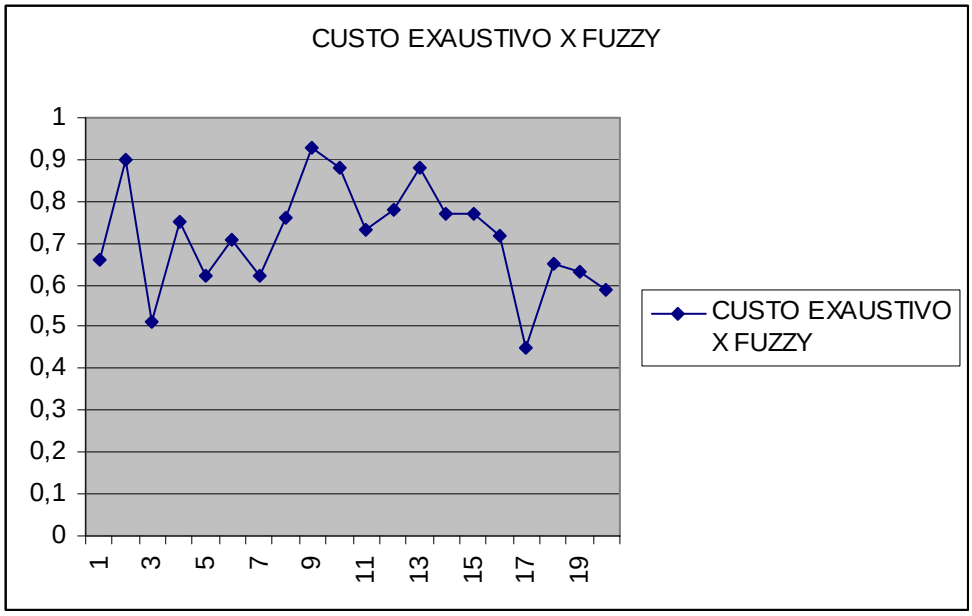
Serão apresentados a seguir alguns ensaios que foram realizados para mensurar a viablidade da Proposta de Alocação *Fuzzy*, ou seja, pelo exposto até este momento, percebe-se que a Solução *Fuzzy* proposta para Solução do Problema Multiponto a Multiponto é eficiente. Mas agora é preciso se apresentar comprovações para a eficácia dessa solução; em outras palavras, precisa-se agora apresentar alguns ensaios como argumentação necessária para a comprovação do quão a Solução *Fuzzy* se apresenta próxima da Solução Ótima do problema Multiponto que seria a Busca Exaustiva. É sabido pelo apresentado em itens anteriores que até certo limite é possível utilizarmos a Busca Exaustiva como a solução ótima. Entretanto, sabe-se que este limite é um limite é muito pequeno, então pode-se tentar utilizar a Busca Exaustiva para até 30 usuários. Ao se ultrapassar este valor, o método de Busca Exaustiva perde-se por completo e pode-se levar até dias, semanas e até anos para gerar todas as combinações possíveis entre usuários e recursos. Assim sendo, foi utilizada neste ensaio uma quantidade pequena de usuários e recursos para que se pudesse utilizar os dois métodos,

tanto o Exaustivo quanto a *Fuzzy*, a fim de se obter os resultados necessários para as comparações entre os mesmos. Tem-se então o modelo com 6 usuários e 3 recursos, onde cada recurso possui 2 vagas a serem preenchidas. Foram feitos 6 ensaios, onde no primeiro ensaio geramos 20 iterações , onde em cada iteração são calculados os custos mínimos para o Modelo Exaustivo e para o Modelo *Fuzzy*. A partir da obtenção desses valores têm-se os gráficos fazendo o comparativo para comprovar se os custos se aproximam na maioria das iterações realizadas. O mesmo procedimento foi feito nos 5 ensaios seguintes, sendo que no segundo ensaio foram 25 iterações, no terceiro 30 iterações, no quarto 40 iterações, no quinto 45 iterações e no sexto e último ensaio 50 iterações. Eis os ensaios.

5.3.1 - Primeiro Ensaio

Neste ensaio foram realizadas 20 iterações.





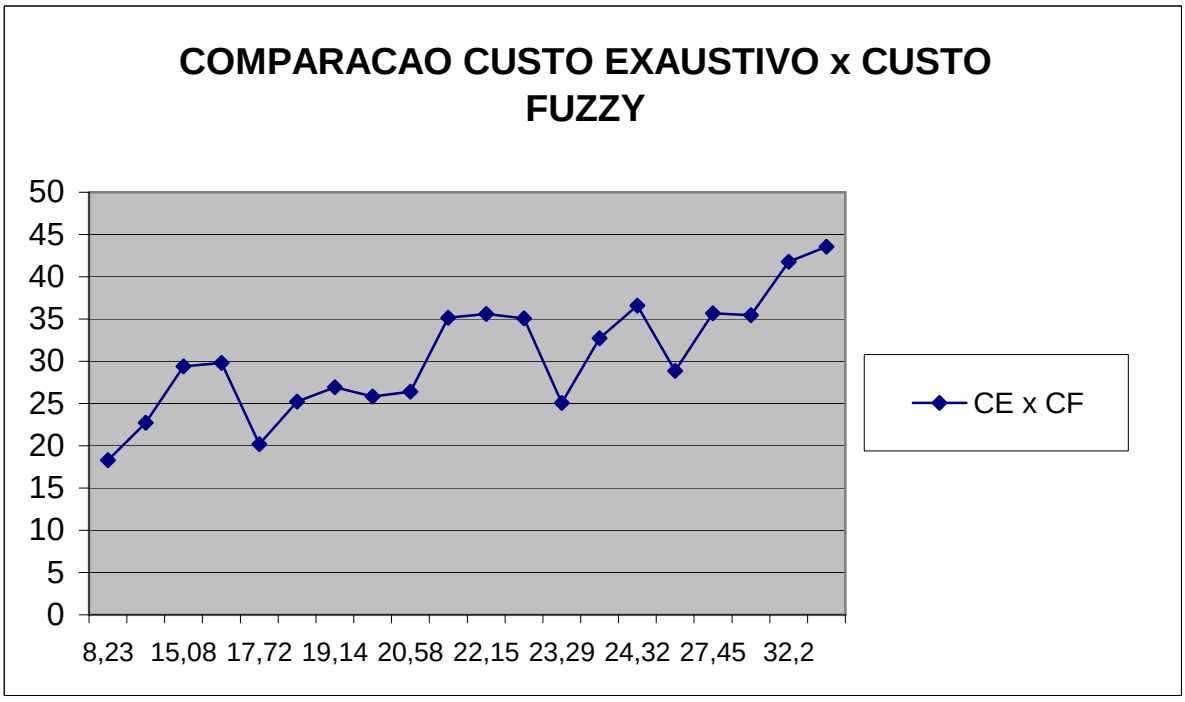


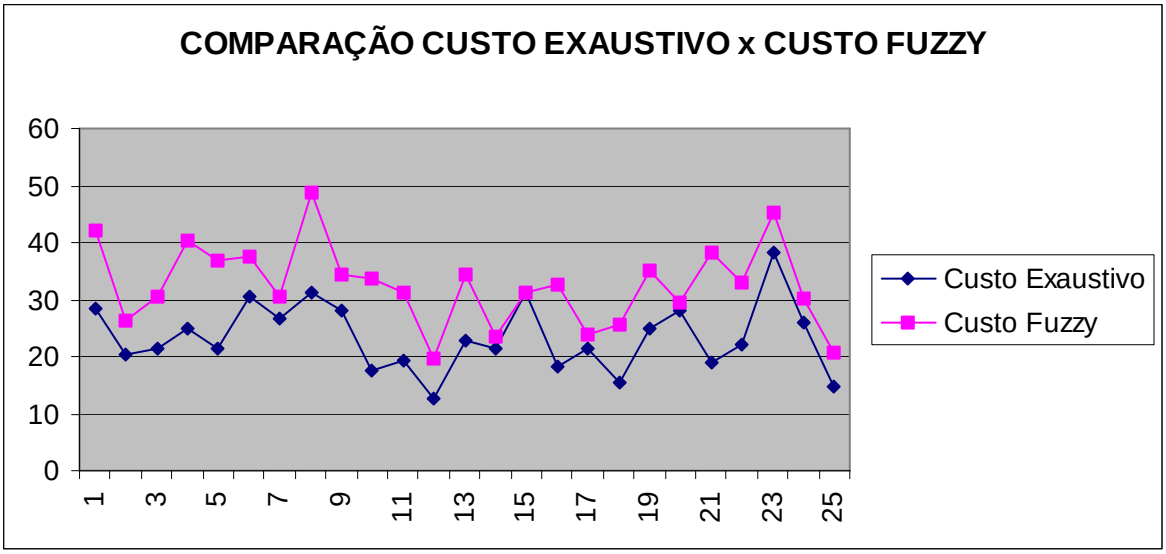
Figura 26 – GRÁFICOS COM AS EXPERIÊNCIAS COM 20 ITERAÇÕES

Em 20 iterações o maior custo *fuzzy* encontrado foi próximo de 50 e o maior custo exaustivo foi um pouco maior do que 30. O que se observa no gráfico acima é que na maior parte das iterações o custo exaustivo é em média 72% menor do que o Custo *Fuzzy*, o que significa dizer que a solução *Fuzzy* está próxima da solução ótima que é encontrada na Busca Exaustiva. Em algumas iterações tivemos até uma diferença de 92% entre o Custo Exaustivo

e o Custo *Fuzzy*, ou seja, a Solução *Fuzzy* poderia ser considerada a solução ótima em alguns casos. A média encontrada de todas as iterações foi de 72 %.

5.3.2. Segundo Ensaio

Neste ensaio foram realizadas 25 iterações. Os gráficos abaixo ilustram a experiência feita.



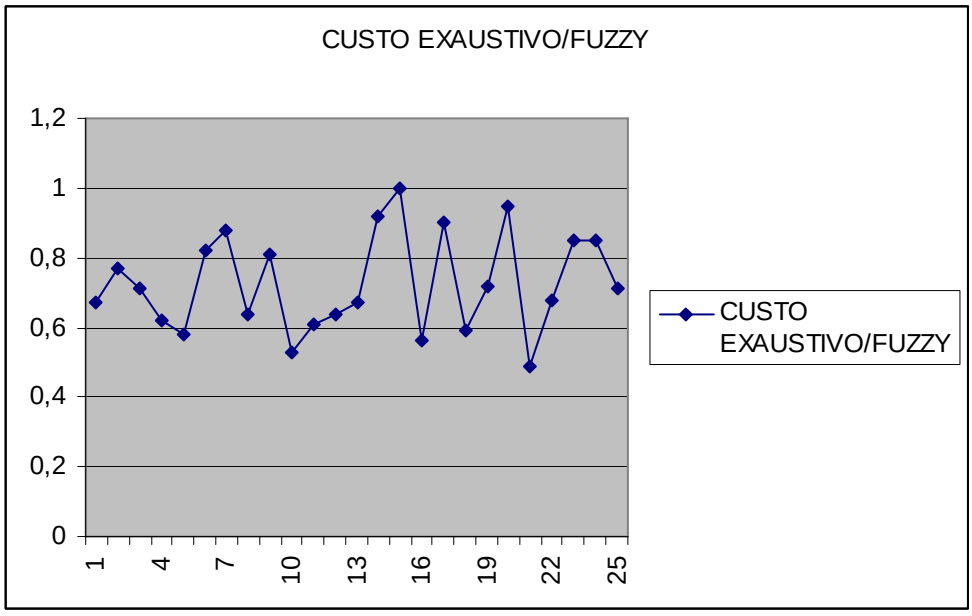
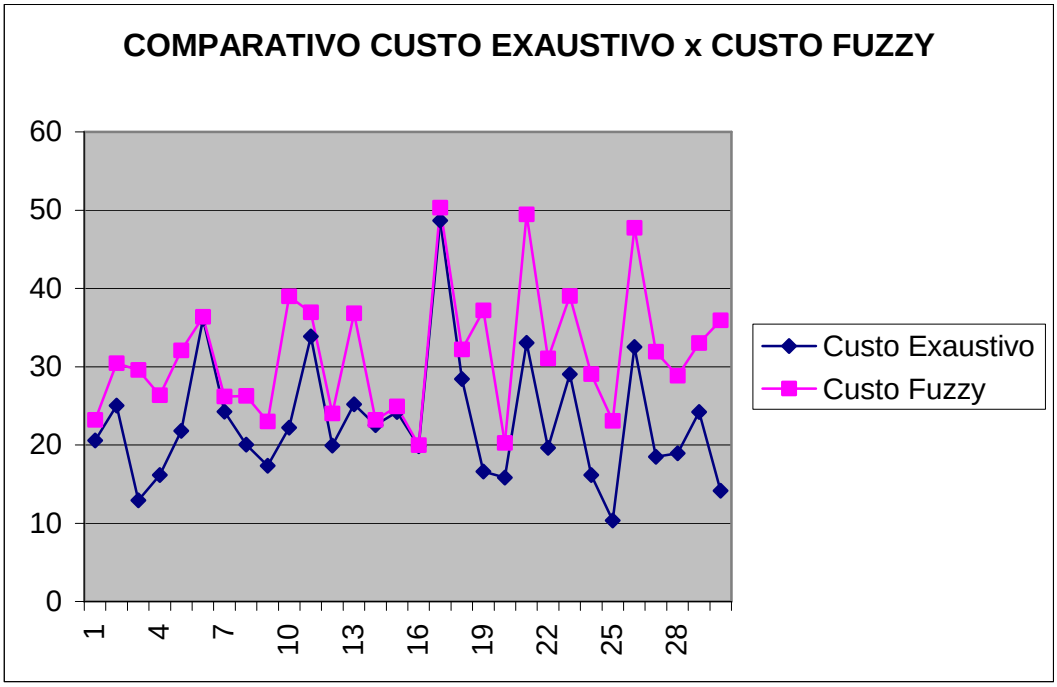


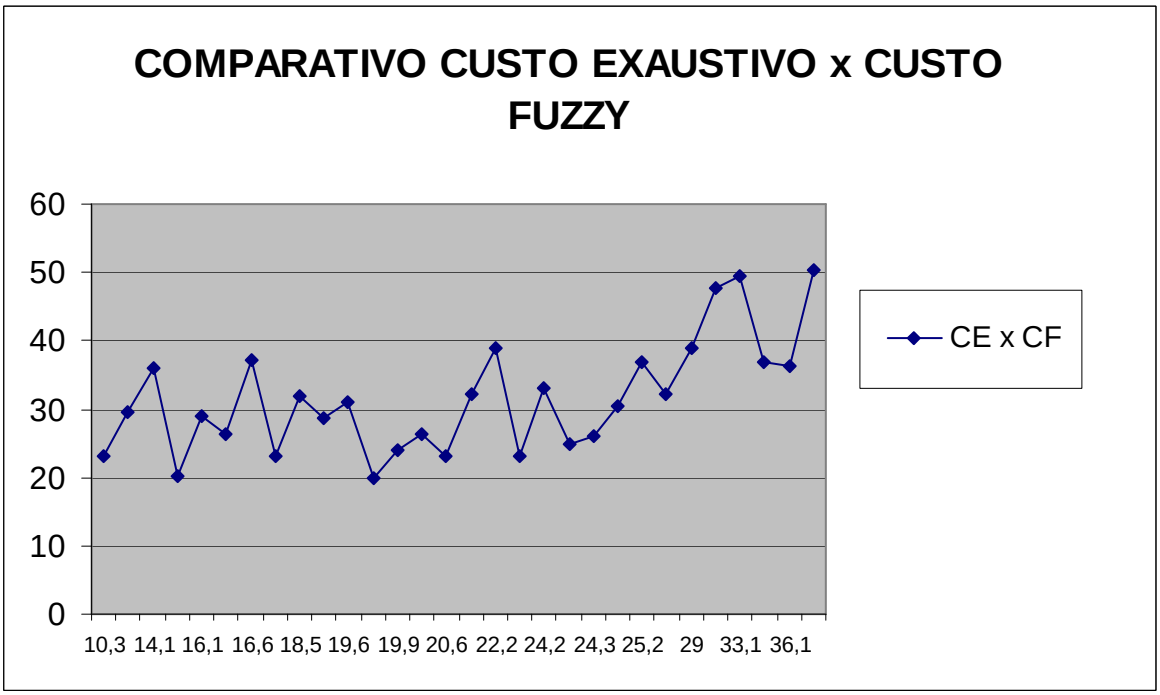
Figura 27 – GRÁFICOS COM AS EXPERIÊNCIAS COM 25 ITERAÇÕES

Neste segundo ensaio tivemos até algumas iterações em que a diferença entre o Custo *Fuzzy* e o Custo Exaustivo foi de ZERO, ou seja, a solução *Fuzzy* coincide com a solução ótima. Isso é verificado em 3 iterações. Em outras 5 iterações o Custo Exaustivo fica muito próximo do Custo *Fuzzy*. Neste ensaio a Técnica *Fuzzy* se apresentou comprovadamente eficaz. Pode-se então concluir que a solução *Fuzzy* é uma alternativa muito boa à solução Ótima. A média dos índices encontrados nas Iterações foi de 72 %.

5.3.3 - Terceiro Ensaio

Neste ensaio utilizou-se 30 iterações e os resultados encontrados foram bem satisfatórios, tendo em vista que a maior parte dos índices encontrados foram maiores do 50% e boa parte deles acima dos 70 %.


COMPARATIVO CUSTO EXAUSTIVO x CUSTO FUZZY
60
50
40
30
20
10
0
1
4
7
10
13
16
19
22
25
28
Custo Exaustivo
Custo Fuzzy


COMPARATIVO CUSTO EXAUSTIVO x CUSTO FUZZY
60
50
40
30
20
10
0
10,3
14,1
16,1
16,6
18,5
19,6
19,9
20,6
22,2
24,2
24,3
25,2
29
33,1
36,1
CE x CF

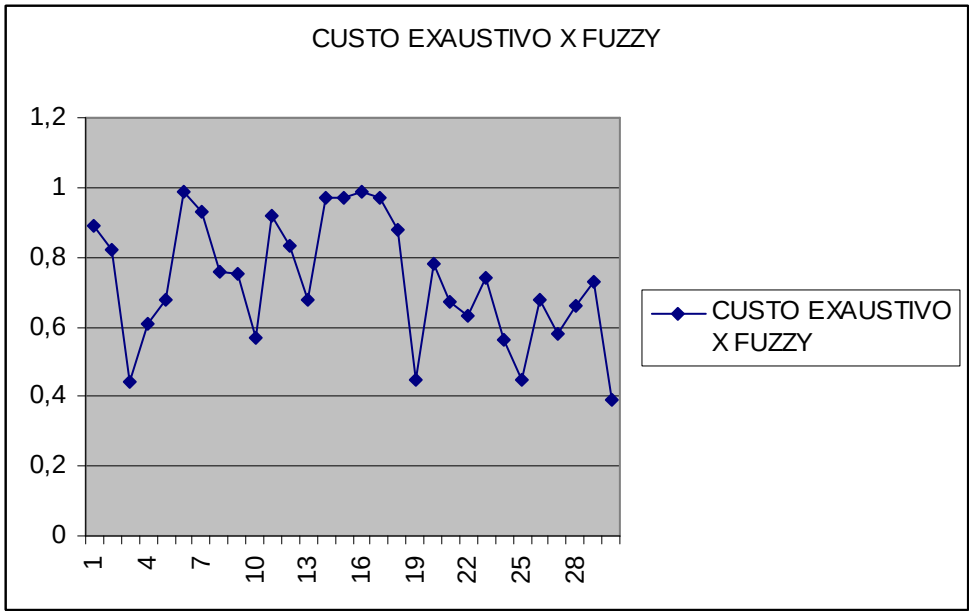
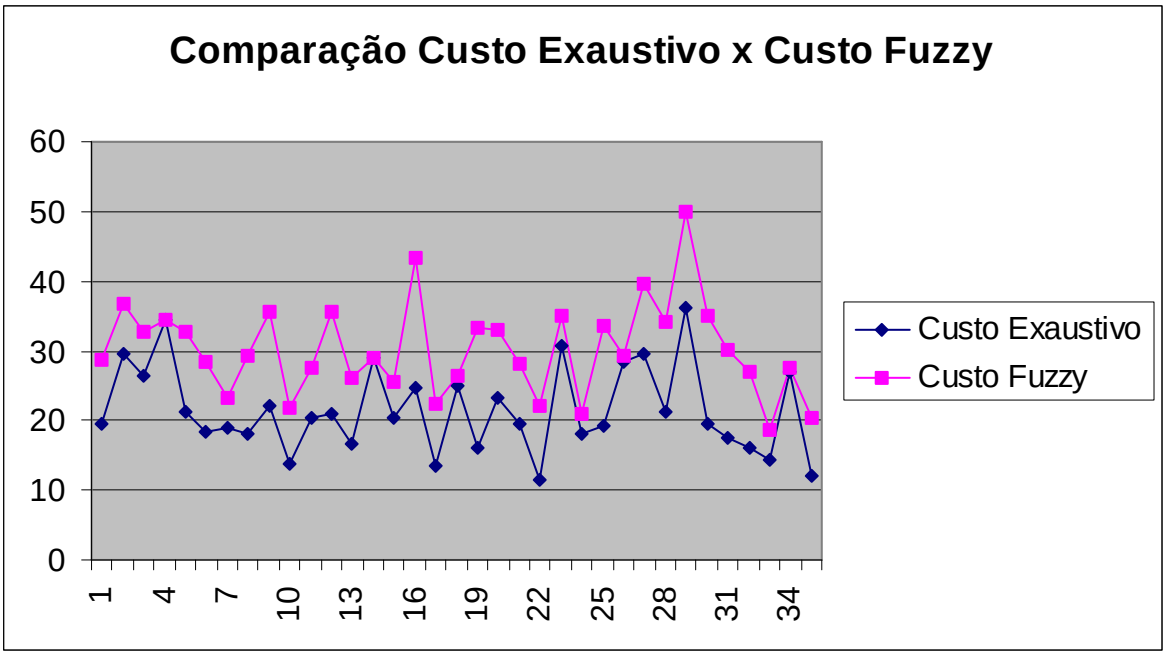


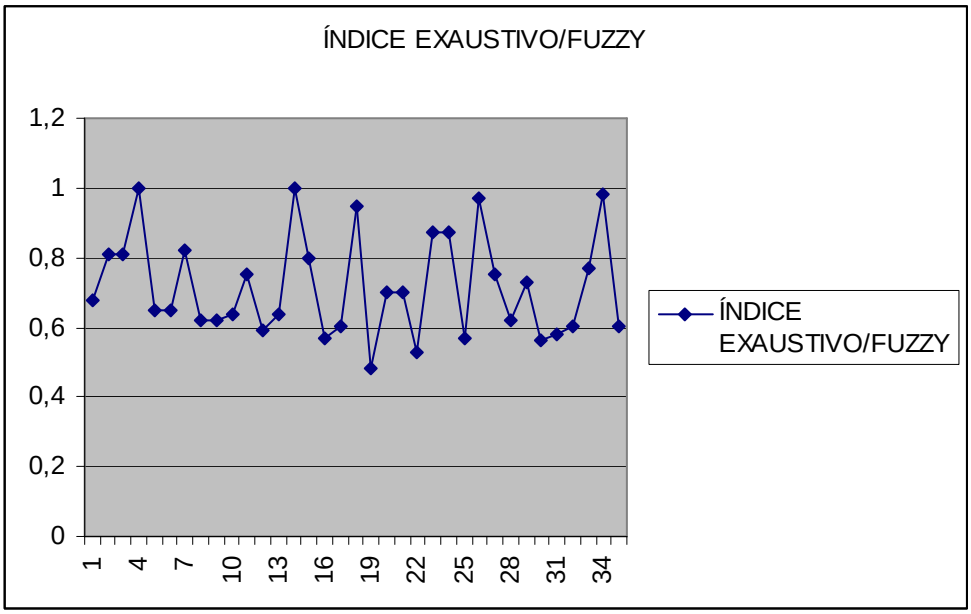
Figura 28 – GRÁFICOS COM AS EXPERIÊNCIAS COM 30 ITERAÇÕES

O que se pode concluir dos resultados encontrados no gráfico é que nesse ensaio com 30 iterações, principalmente nas iterações intermediárias, notam-se inúmeros pontos onde há uma aproximação muito grande entre os valores do Custo Exaustivo e do Custo *Fuzzy*. Na iteração em que temos os valores máximos para os dois custos verifica-se a aproximação da ordem de 97 % na diferença entre os mesmos, isto é, a solução *Fuzzy* pode ser considerada como solução Ótima nessa iteração em particular. Nessas iterações intermediárias a diferença de valor entre os Custos varia entre 97 e 99%. Mas a diferença média dos custos medida nas iterações ficou em 72 %.

5.3.4 - Quarto Ensaio

Neste ensaio foram realizadas 35 iterações. Os gráficos abaixo mostra os resultados obtidos:


Comparação Custo Exaustivo x Custo Fuzzy
60
50
40
30
20
10
0
1
4
7
10
13
16
19
22
25
28
31
34
Custo Exaustivo
Custo Fuzzy


ÍNDICE EXAUSTIVO/FUZZY
1,2
1
0,8
0,6
0,4
0,2
0
1
4
7
10
13
16
19
22
25
28
31
34
ÍNDICE EXAUSTIVO/FUZZY

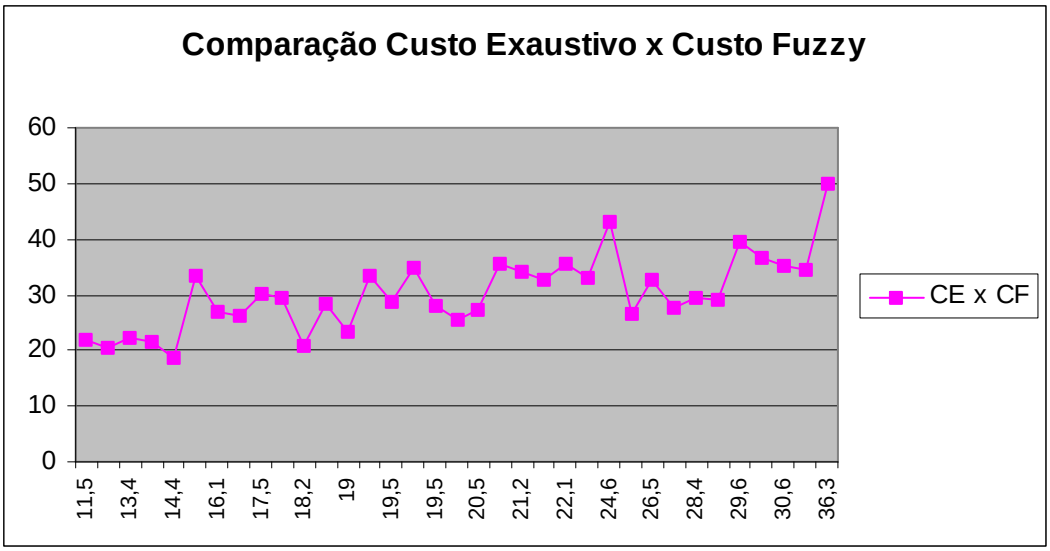
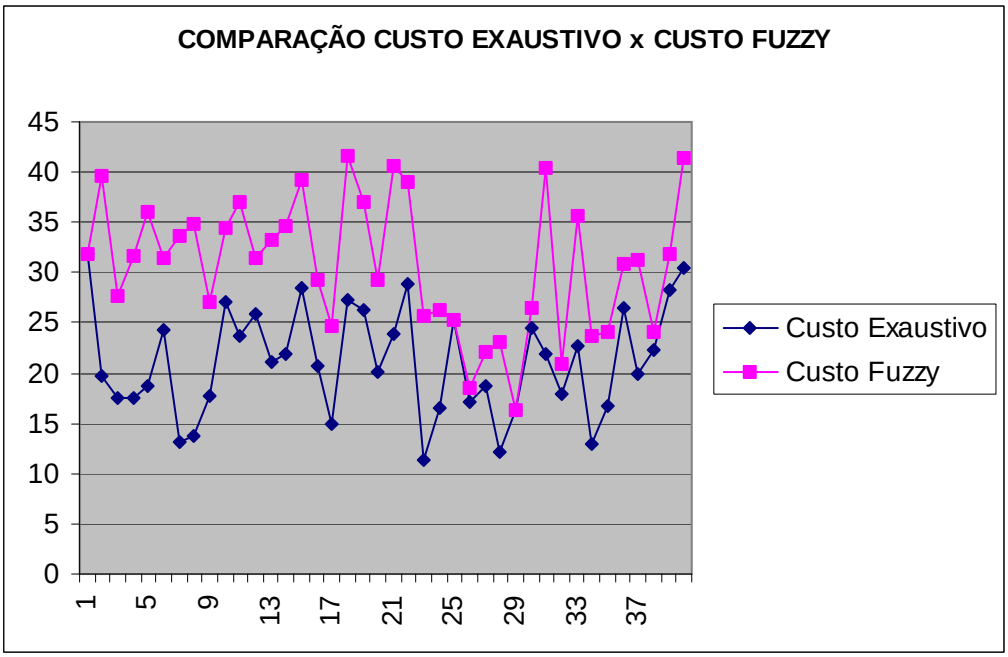


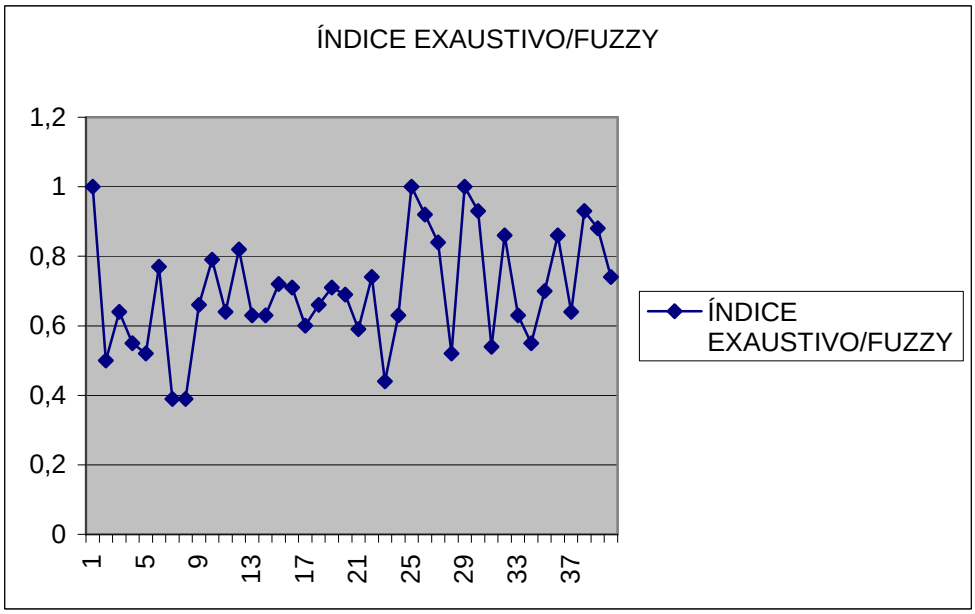
Figura 29 – GRÁFICOS COM AS EXPERIÊNCIAS COM 35 ITERAÇÕES

Neste gráfico há que se constatar dois fatos importantes: em três iterações realizadas encontramos o mesmo valor dos Custos em duas delas e o valor de 97% em uma; o que significa que a solução *Fuzzy* poderia ser considerada a ótima nesses casos tanto quanto a pela Busca Exaustiva. E deve-se ressaltar que na maioria das iterações a diferença entre os custos ficou acima de 70 % , perfazendo a média geral de 72% de todas as diferenças encontradas. E se observa cada vez mais que a solução *Fuzzy* pode ser considerada uma solução totalmente viável e uma ótima alternativa à Alocação Multiusuário.

5.3.5 - Quinto Ensaio

Neste ensaio decidiu-se fazer os testes dos custos com 40 iterações. Na figura abaixo, estão os gráficos com os resultados encontrados.


COMPARAÇÃO CUSTO EXAUSTIVO x CUSTO FUZZY
45
40
35
30
25
20
15
10
5
0
1
5
9
13
17
21
25
29
33
37
Custo Exaustivo
Custo Fuzzy


ÍNDICE EXAUSTIVO/FUZZY
1,2
1
0,8
0,6
0,4
0,2
0
1
5
9
13
17
21
25
29
33
37
ÍNDICE EXAUSTIVO/FUZZY

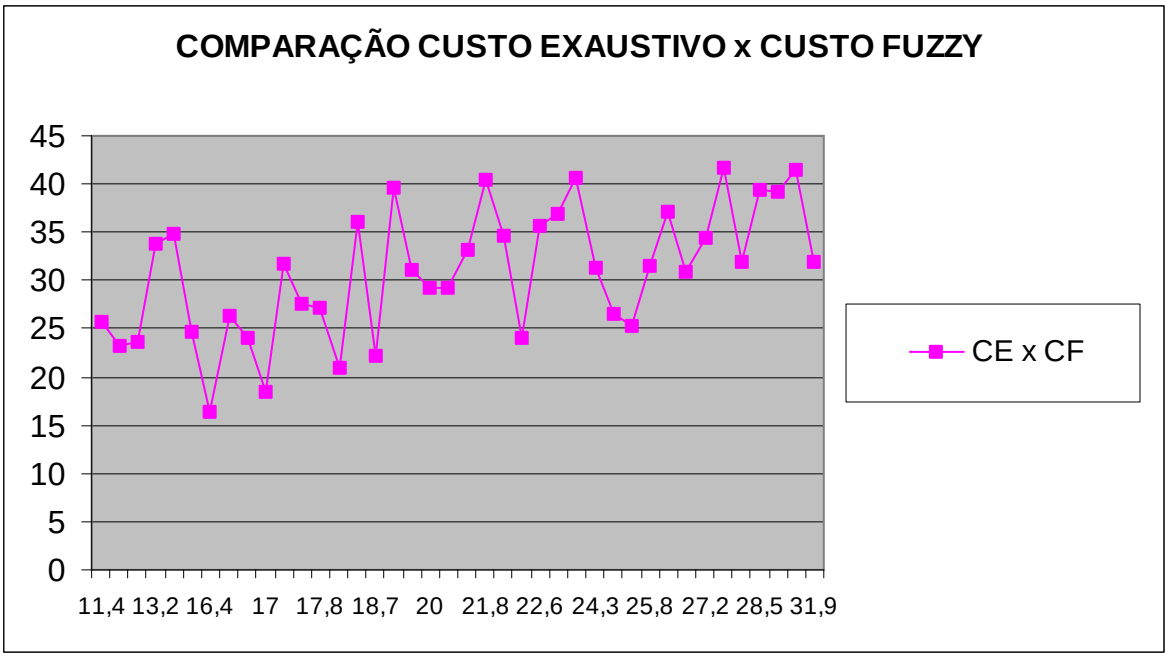
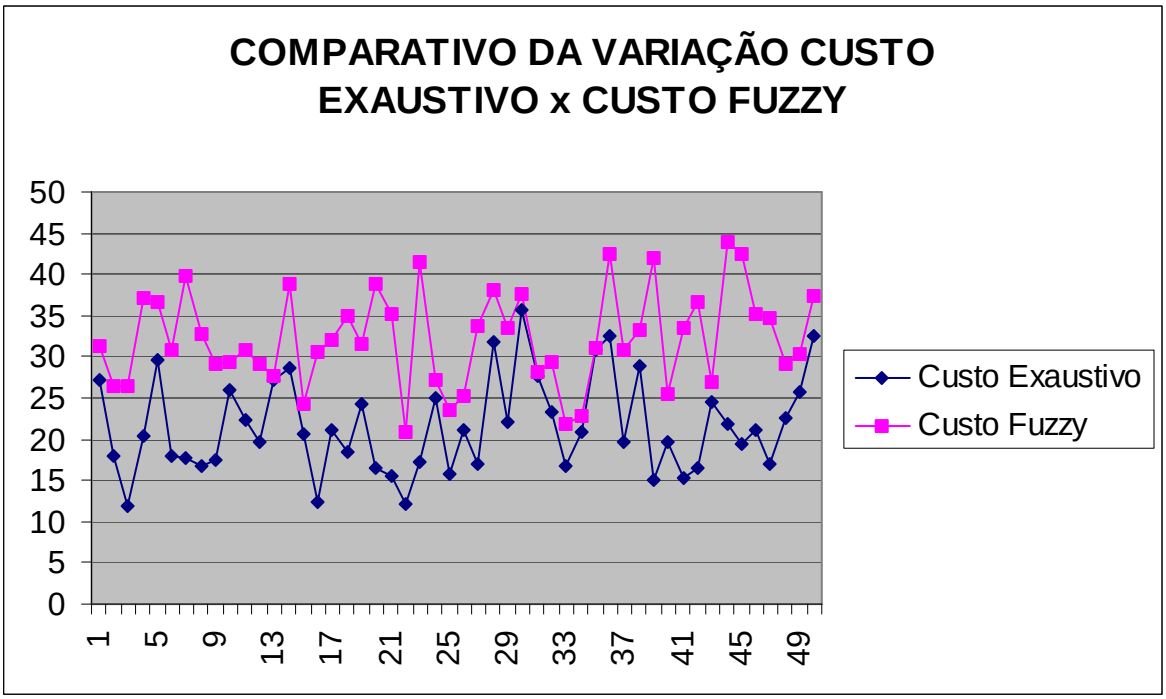


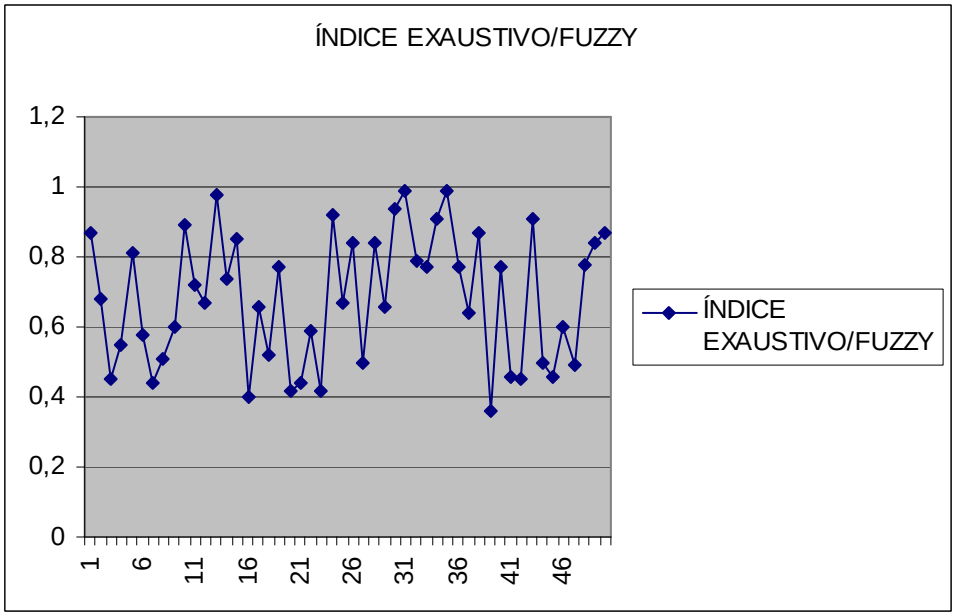
Figura 30 – GRÁFICOS COM AS EXPERIÊNCIAS COM 40 ITERAÇÕES

Neste ensaio, diferentemente dos Ensaios anteriores em que se tiveram iterações apresentando valores iguais nos Custos, não há valores que apresentem um quadro de 100% de eficiência. A distribuição das diferenças entre os valores é mais homogênea. Por outro lado, na maioria das iterações os valores entre os custos apresentam-se relativamente próximos na casa de 71 % em média. Nas últimas iterações observa-se uma proximidade maior entre os mesmos. No final de tudo, a proposta *Fuzzy* teve um desempenho satisfatório, tendo em vista uma boa proximidade nos valores em relação ao Custo Exaustivo.

5.3.6 – Sexto Ensaio

Neste item o Ensaio foi feito com 50 iterações e os resultados apresentados foram mais homogêneos do que nos ensaios anteriores. Eis os gráficos com os resultados abaixo.


COMPARATIVO DA VARIAÇÃO CUSTO EXAUSTIVO x CUSTO FUZZY
50
45
40
35
30
25
20
15
10
5
0
1
5
9
13
17
21
25
29
33
37
41
45
49
Custo Exaustivo
Custo Fuzzy


ÍNDICE EXAUSTIVO/FUZZY
1,2
1
0,8
0,6
0,4
0,2
0
1
6
11
16
21
26
31
36
41
46
ÍNDICE EXAUSTIVO/FUZZY

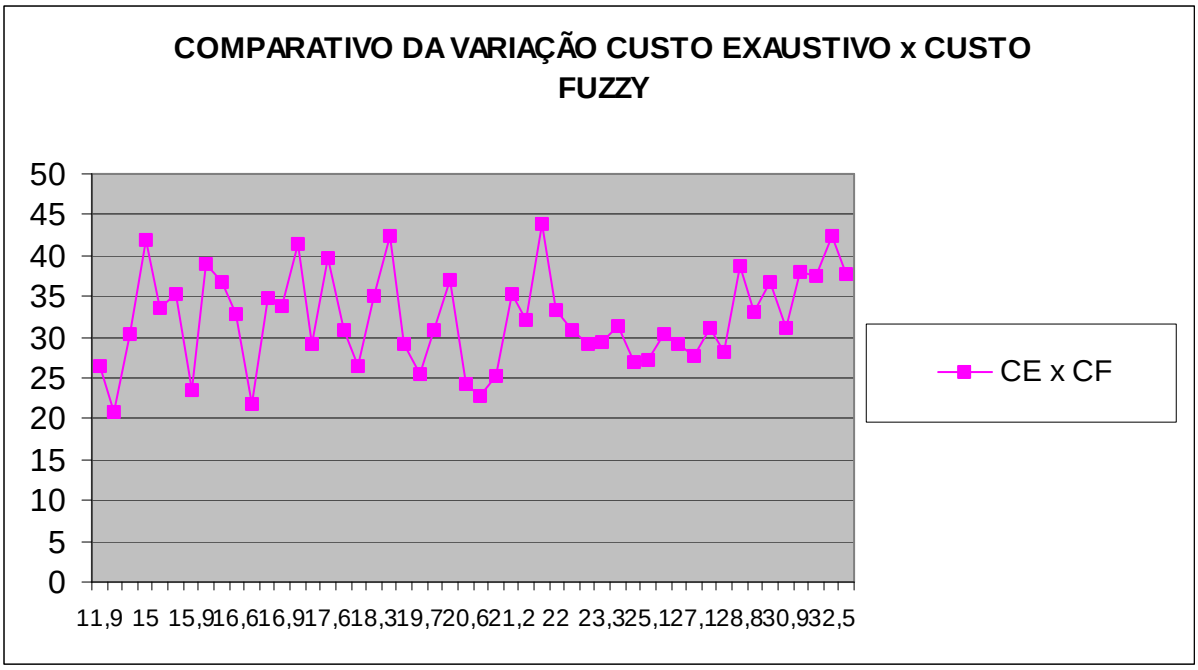
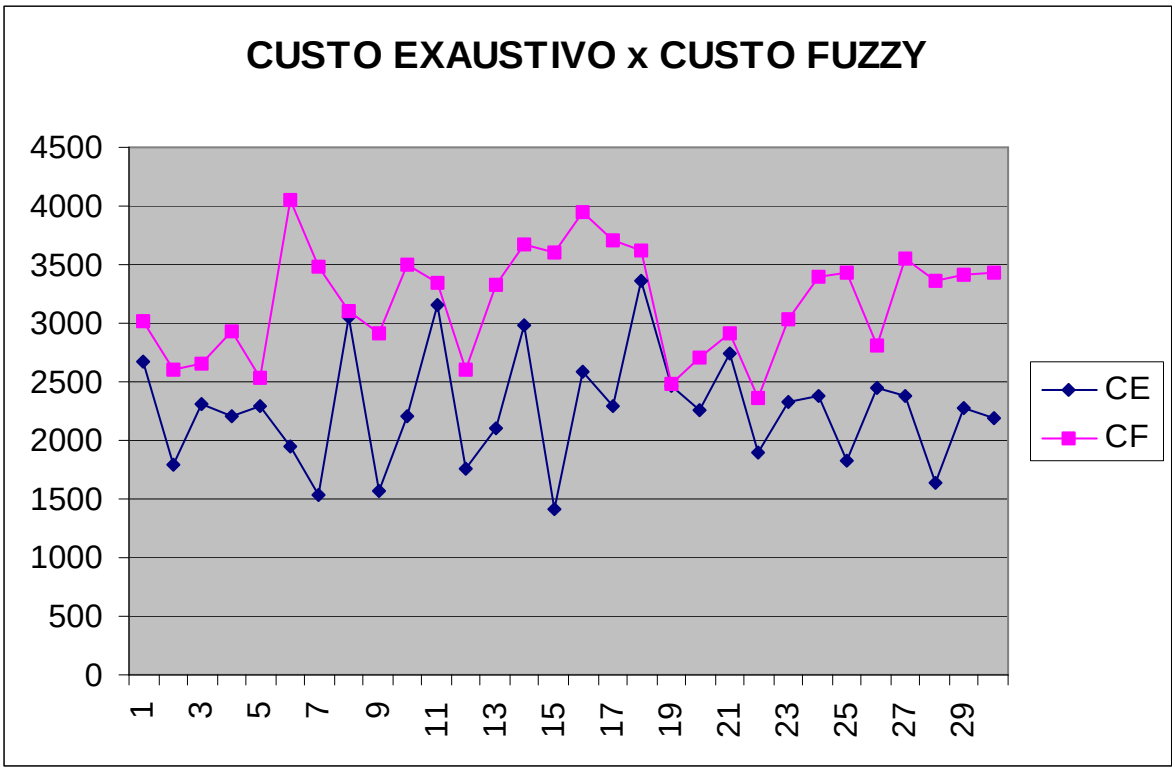


Figura 31 – GRÁFICOS COM AS EXPERIÊNCIAS COM 50 ITERAÇÕES

O que foi observado neste ensaio com 50 iterações é que a diferença entre os valores dos Custos *Fuzzy* e Exaustivo beirou entre 70 e 90 %. A média geral encontrada foi em torno de 72%. Ou seja, pode-se afirmar que a diferença entre os custos é maior do que 70 % , o que denota a boa eficácia da aplicação *fuzzy* como uma boa alternativa à solução da Alocação Multiusuário.

Todos estes ensaios foram realizados tendo em vista a utilização de números aleatórios nas coordenadas cartesianas, tendo em vista que, para cada iteração era necessário um conjunto de dados diferente, a fim de se observar o comportamento de ambos os custos a cada alteração. Para se efetuar uma precisão maior nos ensaios, foi realizado um outro ensaio com 30 iterações, onde os números aleatórios que determinavam os valores das coordenadas para cada usuário ou recurso poderia ser de, no máximo 1000. Eis o gráfico:



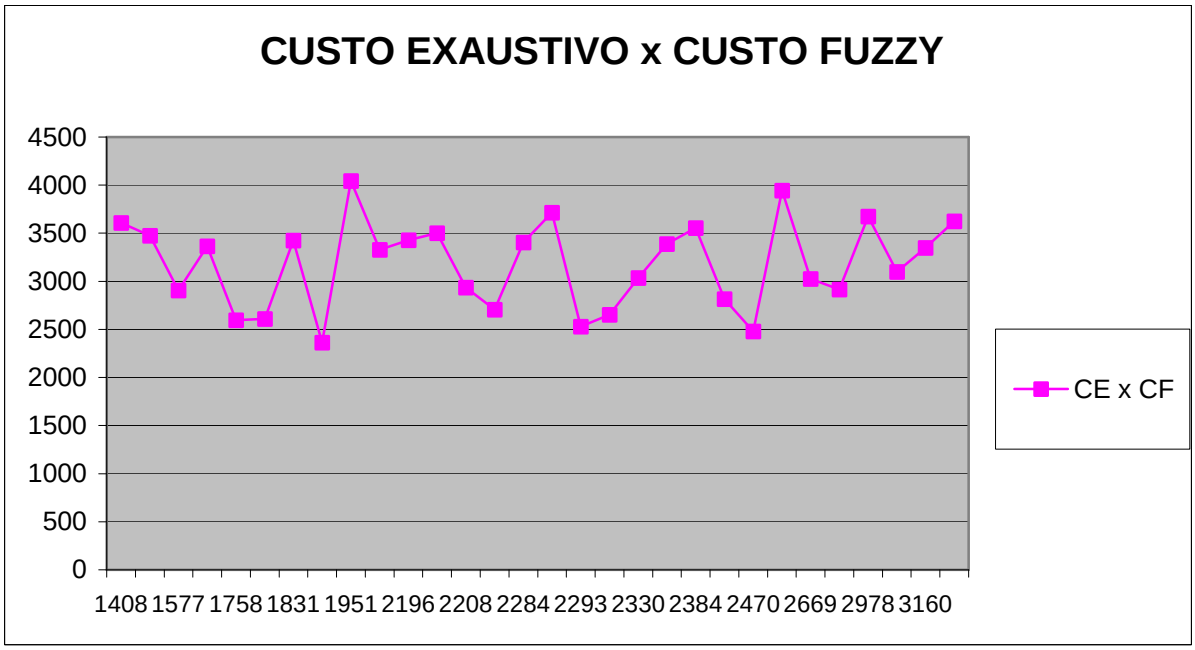


Figura 32 – GRÁFICOS COM AS EXPERIÊNCIAS COM 30 ITERAÇÕES EM UM PLANO CARTESIANO COM CUSTOS DE ATÉ 4500 U.C

Pelo gráfico gerado verifica-se que os resultados foram semelhantes ao dos gráficos com iterações com os números de coordenadas menores do que 100 como nos ensaios anteriores. A média encontrada foi de 73% e os resultados dos custos ficaram bem próximos de um modo geral.

Basicamente, o objetivo com a realização desses ensaios foi mostrar que, através da diferença entre os custos mínimos da Busca Exaustiva e o Custo Mínimo da Solução *Fuzzy*, que a Proposta da Solução *Fuzzy* pode ser uma boa alternativa à Solução tradicional do problema de Alocação Multiusuário, uma vez que esta solução por Busca Exaustiva que é considerada a solução ótima dessa alocação torna-se completamente inviável a partir do momento em que se aumenta significativamente o número de usuários. E pelos resultados apresentados nos ensaios, a Solução *Fuzzy* pode ser considerada muito próxima da Ótima, sendo que inclusive em alguns desses ensaios, em algumas das iterações realizadas neles, a diferença entre os Custos Mínimo Exaustivo e o Custo Mínimo *Fuzzy* chegou a ZERO. Em outras palavras, a Solução *Fuzzy* possui uma boa eficiência nos casos maiores e pode ter uma boa capacidade de aplicabilidade em muitos casos envolvendo a Alocação Multiusuário.

5.4 - Comparação Do Método *Fuzzy* Com O Método Exaustivo E o Método Ingênuo

Aqui nesta seção serão relatados alguns ensaios feitos com os Métodos *Fuzzy*, Exaustivo e Ingênuo e a Comparação dos ensaios se faz ao se gerar duas comparações, com vistas a Comparação final que é a comparação do desempenho da proposta *Fuzzy* com o Algortimo Ingênuo. A implementação dessa comparação gera primeiramente dois Índices de Comparação, o do método *fuzzy* em relação ao método exaustivo e o do algoritmo ingênuo em relação ao método exaustivo. A exemplo do que foi feito no Tópico anterior foram criados vários conjuntos de iterações.

Os resultados encontrados nestes ensaios demonstraram que o Método *Fuzzy* apresentou na média um índice de 92 % enquanto que o Algoritmo Ingênuo apresentou 89 %. A um leitor leigo, aparentemente a grosso modo ele diria que os dois métodos apresentariam o mesmo desempenho, porém ao longo das iterações feitas, notou-se que em algumas delas, o Método *Fuzzy* superava o Ingênuo de 10 a até 15 pontos percentuais, tanto que na média a diferença entre os dois chega a 3 pontos percentuais.

Foi feito também um ensaio específico onde os recursos foram colocados, dentro dos sistemas de coordenadas cartesianas em uma mesma ordenada e os usuários espalhados ao longo do sistema. A figura, ilustrando esse ensaio, encontra-se a seguir. O resultado da comparação denotou uma eficiência de 100% para o método *Fuzzy* e em torno de 60 % para o método Ingênuo, ou seja, uma diferença de 40 pontos percentuais em relação ao Método *Fuzzy*, uma diferença considerável.

Conforme visto no Capítulo anterior, o Algoritmo Ingênuo é dependente da ordem de entrada dos usuários, quando do início do “loop” que é criado para percorrer todos os usuários para a alocação dos mesmos. E neste ensaio foi notado que, quando se modifica a ordem de entrada dos usuários no algoritmo o desempenho do mesmo tem uma razoável melhora. Ao se modificar essa ordem o desempenho fica estacionado entre 66 e 75 pontos percentuais. E nessas mudanças, o desempenho do Método *Fuzzy* não se altera em momento algum.

A partir desse ensaio feito, pode-se concluir que o Método *Fuzzy*, fazendo uma ou outra modificação é um método perfeitamente válido como uma boa alternativa à solução para a Alocação Multiusuário. E essa comparação com o Método Ingênuo foi de fundamental relevância, uma vez que o Algortimo Ingênuo funciona de forma similar ao Método *Fuzzy* só que era preciso verificar que o Algortimo *Fuzzy* fosse mais efetivo que o Algoritmo Ingênuo.

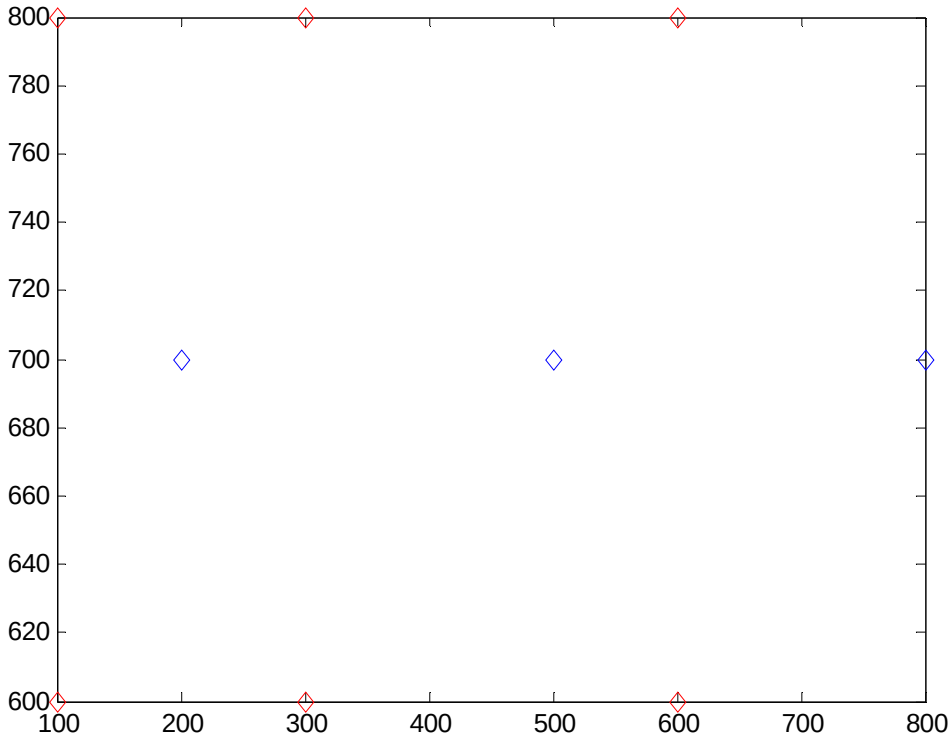


Figura 33 – PLANO CARTESIANO 800 POR 800 ONDE OS RECURSOS E USUÁRIOS ESTÃO DISPOSTOS. OS USUÁRIOS SÃO OS PONTOS EM VERMELHO E OS RECURSOS EM AZUL.

5.5 – Comparação Do Método *Fuzzy* Com O Método Ingênuo

Nesse tópico será apresentada uma comparação entre o Método do algoritmo ingênuo e o método *fuzzy*. Na seção anterior foram apresentadas duas comparações: a comparação Exaustivo/Ingênuo e a comparação Exaustivo/*Fuzzy* e uma vez efetuado os Índices havia o confronto final entre as duas comparações. Aqui será apresentado um índice para comparar os dois métodos. Esse índice possui a seguinte fórmula:

Indice = Custo *Fuzzy*/Custo Ingênuo

Foram realizados dois ensaios com 20 e 30 iterações.

5.5.1 - Primeiro Ensaio

Neste primeiro ensaio ocorreram 20 iterações. A média foi de 70%, o que significa dizer que nesta comparação o Método *Fuzzy* é mais eficaz tendo em vista que o índice é menor do que 100%. No gráfico abaixo, pode ser verificado como foram as iterações e a comprovação da eficácia do método *Fuzzy* em relação ao algoritmo ingênuo percebendo-se a disparidade nos gráficos dos dois métodos.

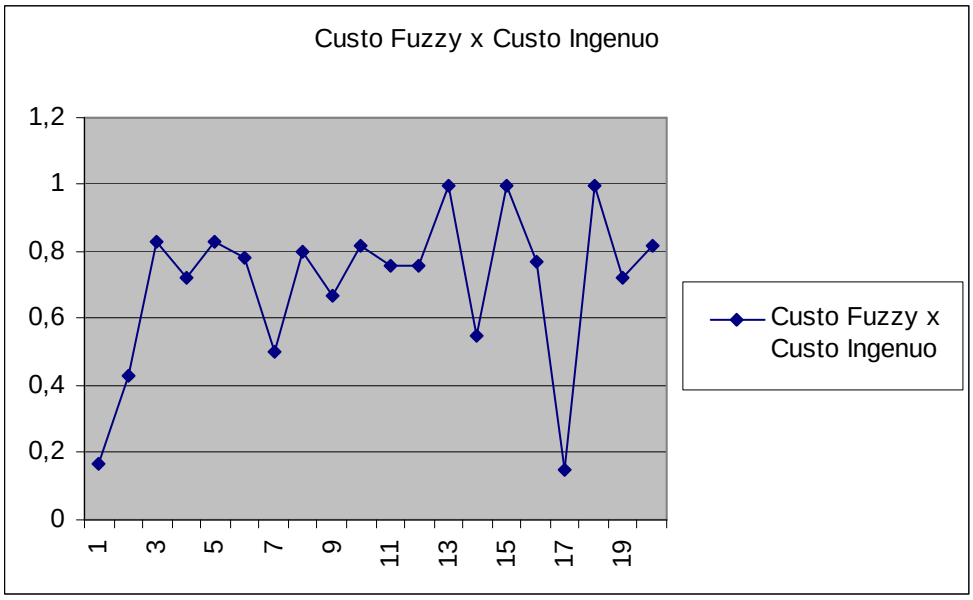


Figura 34 – GRÁFICO *FUZZY*/INGÊNUO COM AS EXPERIÊNCIAS COM 20 ITERAÇÕES

5.5.2 - Segundo Ensaio

No segundo ensaio realizado teve-se na média um índice de 72 %. Considerando-se que o índice entre os dois, de acordo com a formulação proposta, deve ser menor do que 1 (100%) , pode-se afirmar que o objetivo foi alcançado com êxito. Portanto nesse ensaio, o Método *Fuzzy* provou ser mais eficaz também. O gráfico encontra-se abaixo.

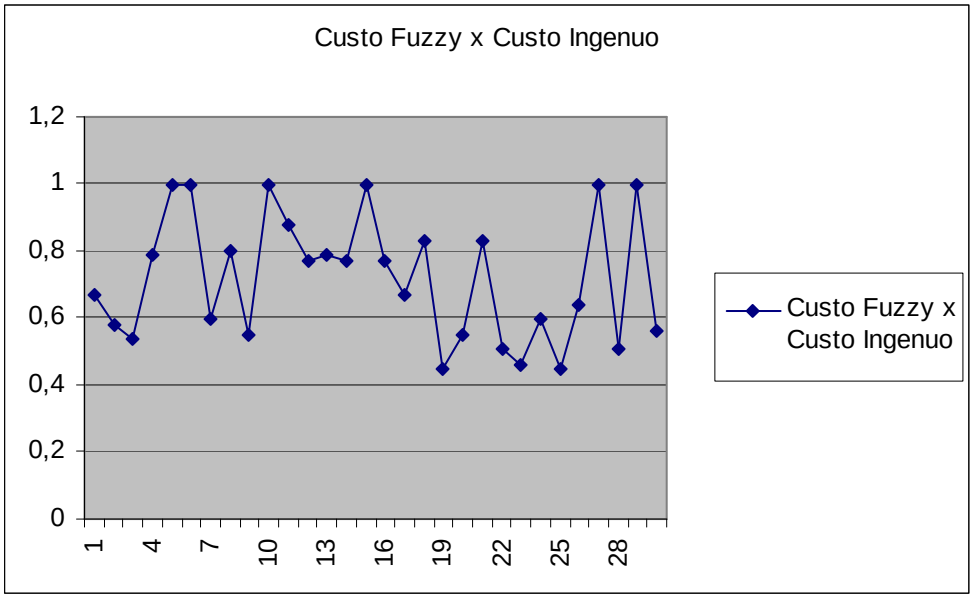


Figura 35 – GRÁFICO *FUZZY*/INGÊNUO COM AS EXPERIÊNCIAS COM 30 ITERAÇÕES

5.6 - Comentários

Neste capítulo foram descritos os resultados e alguns ensaios realizados para se fazer as comparações entre o Método Exaustivo, o Algoritmo Ingênuo e o Método *Fuzzy*. Esses ensaios objetivam mostrar apenas se o Método *Fuzzy* é um método próximo da Solução Ótima (Exaustivo). Em todos os ensaios, na maioria das iterações, a diferença entre os Custos do Método Exaustivo e o Custo do Método *Fuzzy* foi de mais de 70 % em média, o que denota que o Método *Fuzzy* é perfeitamente viável uma vez que se encontra bem próximo da solução ótima. Então, diante dos ensaios realizados e dos resultados dos testes feitos em cima do protótipo, onde tivemos as alocações sendo feitas com bastante rapidez de processamento, pode-se concluir que um Sistema Especialista *Fuzzy* é um sistema bastante eficiente e pode ser considerado uma boa alternativa ao Método Exaustivo.

CAPÍTULO 6 – CONCLUSÕES

A partir dessa proposta, o que se espera é uma investigação mais profunda na metodologia apresentada. O que se pode depreender do que foi colocado até agora é que o uso da Lógica *Fuzzy* aqui ilustrado é uma alternativa de boa viabilidade no tocante ao problema da Alocação Multiponto a Multiponto, visto que a Busca Exaustiva a partir de um certo número de usuário torna-se custosa e até pode-se afirmar, complexa também, uma vez que as visitas em árvores, a cada aumento do número de usuários no sistema aumenta o número de níveis da árvore de busca e, conseqüentemente mais nós a se visitar, o que torna o processamento de busca cada vez maior e mais penoso.

Assim sendo, essa proposta visa a uma solução do Problema de Alocação Multiponto a Multiponto de forma mais rápida e porque não dizer, mais simples, pois além de um processamento mais enxuto, a solução *fuzzy* é uma simulação do pensamento, no sentido de que a forma como o algoritmo foi projetado faz-se uma tentativa de se aproximar o pensamento do que um funcionário de escola ou Zona Eleitoral faria ao fazer a tentativa de alocação de um aluno e/ou um eleitor. Portanto, a solução aqui mostrada também mostra-se válida, no sentido de se montar um sistema mais amigável a quem for proceder a alocação. Acredita-se que assim, essa solução possa ser uma opção interessante de implementação de sistema para escolas, zonas eleitorais, funcionários públicos etc.

Quando foi citado que uma pesquisa maior nesse tema deve ser feita é porque ainda que a solução *fuzzy* tenha se apresentado bem viável, mais enxuta e com uma boa eficiência, ela necessita de alguns ajustes nos casos em que a diferença do Custo dela para o Custo Exaustivo ficou em menos de 50%. É preciso investigar mais profundamente isso, a fim de se aprimorar ainda mais essa solução.

Outro item também a ser aprimorado é o algoritmo de geração dos índices, no sentido apenas de se diminuir ainda mais o tempo de processamento dos índices. Não que o tempo tenha sido enorme, não é absolutamente o caso aqui, mas esse tempo pode ser diminuído no caso em que se faz a geração de todos os índices de todos os usuários e todos os recursos. Quando se faz apenas para um ou dois casos o tempo é da ordem de milésimos de segundo. Tudo são meramente melhorias para a proposta aqui exposta ser mais eficiente e viável.

O que se espera, além das melhorias dessa proposta aqui apresentada, é que essa proposta possa contribuir para um melhor suporte para tomadas de decisão mais eficientes em

problemas que envolvam alocações de várias pessoas para vários lugares nos exemplos supracitados no início desse tópico.

A proposta de solução apresentada neste trabalho objetivou buscar uma alternativa com um bom grau de viabilidade, tal que a solução encontrada fosse próxima da solução ótima que é a Busca Exaustiva. O trabalho que foi proposto nesta dissertação é apenas o início de uma longa pesquisa em torno de qual método se adequará melhor ao problema Multiponto a Multiponto. Esse problema trata-se de um problema de otimização e existem outras formas de solução para propostas envolvendo otimização tais como Algoritmo Guloso e Programação Dinâmica. Assim sendo, fica aqui a sugestão de trabalhos envolvendo a comparação da Proposta através da Lógica *Fuzzy* aqui apresentada com tais métodos, analisando a eficácia do Método *Fuzzy* frente a eles.

Essa dissertação aqui apresentada objetiva propor uma alternativa viável e quiçá mais eficiente para a questão da Alocação Multiponto a Multiponto que é um sistema que emprega Lógica *fuzzy*. Essa abordagem é diferente de algumas soluções já apresentadas, uma vez que, ao contrário das soluções que utilizaram *Dijkstra*, Busca Exaustiva e Algortimos Genéticos, ela trata de um método que simula a aproximação do raciocínio humano para resolver a questão.

Cabe-se ressaltar também que, apesar do caráter científico, esse trabalho visa a mostrar que a ciência não procura ficar atinente apenas à teoria, existe um objetivo maior de se colocar o quão prático será esse trabalho em futuras implantações de políticas com finalidades que incluam alocação de vários usuários a vários recursos.

Por fim, o autor desse trabalho espera que, com isso, ele possa ter dado uma boa contribuição aos trabalhos na área de Inteligência Computacional, em nível não só científico como também no aspecto prático, a fim de que a implementação desse sistema possa melhorar os procedimentos administrativos nas áreas em que a Alocação Multiponto a Multiponto se faz presente.

REFERÊNCIAS BIBLIOGRÁFICAS

AGUIAR, H e OLIVEIRA, JR., Lógica Difusa Aspectos práticos e aplicações, 1999.

COLOMBO, L.R. e RANGEL, M.C., *Um Algoritmo Construtivo Baseado Em Uma Abordagem Algébrica Do Problema Quadrático De Alocação*, <http://www.scielo.br/pdf/pope/v26n1/29478.pdf>.

CORMEN, T. H., LEISERSON, C. H., RIVEST, R. L., e STEIN, C., 2002, *Algoritmos Teoria e Prática*, Editora Campus.

COX, E., The fuzzy systems handbook: a practitioner's guide to building, using, and maintaining fuzzy systems, 1994

DUBOIS, D. e PRADE, H., *Fuzzy Sets and Systems, Theory and Applications*, Academic Press, Orlando, 1980.

FADEL, J. E., 2003, *Aplicação da Geomática no Auxílio da Matrícula em Escolas da Rede Pública*, Dissertação de Mestrado, Faculdade de Engenharia – UERJ.

FILHO, O. B, PEDROZA, A. C. e LEÃO J. L. S. Uma ferramenta para verificação de sistemas distribuídos com lógica nebulosa: Implementação e experiências, Artigo do Departamento de Enga. De Sistemas e Computação/FEN-UERJ, 2001.

KANDEL, A., *Fuzzy Mathematical Techniques with Applications*, Addison-Wesley Publishing Co., 1986.

KLIR, G. J. e FOLGER, 1988, *Fuzzy Sets, Uncertainty, ans Informations*, Prentice Hall, Englewood Cliffs Exhaustive Search pg 513.

KOERICH, A., 2004, *Força Bruta - Técnicas de Projetos de Algoritmos* — PUC/PR, <www.ppgia.pucpr.br/~alekoe/PAA/20042>.

LEE, C. C., *Fuzzy Logic in Control Systems: Fuzzy Logic Controller*, parts I & II, IEEE Transactions on Systems, Man and Cybernetics, vol. 20, nº 2, 1990.

MAMDANI, E. H., 1976, *Advances in the linguistic synthesis of fuzzy controllers*, 8:669-678

MENDEL, J. M., *Fuzzy Logic Systems for Engineering: A Tutorial*, Proceedings of the IEEE, vol. 83, nº 3, março, 1995.

MARIA, J. e COUTINHO-RODRIGUES, J., 2004, *Um Sistema de Apoio ao Planejamento de Rotas Turísticas: Incorporação de Técnicas de Optimização e Heurísticas com Tecnologia* Sig – Universidade de Coimbra.

MIRANDA, M. N., *Algoritmos Genéticos: Fundamentos e Aplicações*, http:/www.gta.ufrj.br /~marcio/genetic.html.

MURGEL FILHO, W., 2005, *Sistema Nebuloso de Apoio à Produção de Plantas de Valores*, Faculdade de Engenharia – UERJ.

PIRES, L. A., 2002, *Algoritmo Genético Aplicado na Localização de Escolas no Município de Coronel Vivida* – PR, Tese de Mestrado, Universidade Federal do Paraná. www.cpgmne.uf**pr**.br/dissertacoes.

SANDRI, S. e CORREA, C., 1999, *Lógica Nebulosa* – INPE, http://www.deti.ufc.br /~guilherme/PAPERS.

SAWADA, J., 1997, Combinatorial Generation, http://theory.cs.uvic.ca/inf/comb /CombinationsInfo.html.

SEDGEWICK, R., 1983, *Algorithms*, Brown University, Addison-Wesley Pub. Company, Exhaustive Search pg 513.

SILVA, A, P. da, Escalamento Heurístico Baseado em Lógica Difusa, Tese de Mestrado, Programa de Engenharia Elétrica, COPPE/UFRJ, 1995

SOUZA, F. J., *Lógica Nebulosa*, Notas de Aula.

SZWARCFITTER, J. L., 1988, *Grafos e Algoritmos Computacionais*, Editora Campus.

TANSCHEIT, Ricardo. Controle Nebuloso, 9o CBA, UFES, Vitória, ES, Brasil, pp. 82-95.

TANSCHEIT, R., *Controle Nebuloso*, Anais do 9º Congresso Brasileiro de Automática, setembro, 1992.

TERANO, T., ASAI, K. e SUGENO, M., *Fuzzy Theory and its Applications*, Academic Press, Boston, 1987.

ZADEH, L. A., 1965, *Fuzzy sets. Fuzzy Sets, Information and Control* 8: 338 - 353.

ZADEH, L. A., 1978, *Fuzzy sets as a basis for a theory of possibility, Fuzzy Sets and Systems*

ZADEH, L. A., *The Role of Fuzzy Logic in the Management of Uncertainty in Expert Systems*, Fuzzy Sets and Systems 11, 1983.

ZADEH, L. A., *Fuzzy Logic*, Computer, 1988.

ZIMMERMANN, H.J., (1985). Fuzzy Mathematical Tecniques with Applications, Addison-Wesley Publishing Co, 1985.

ZIMMERMANN, H. J., (1985). *Fuzzy Set Theory - and Its Applications*, Kluwer-Nijhoff Publishing, 1985.

APÊNDICE A - CÓDIGO-FONTE DO ALGORITMO DE ALOCAÇÃO

```
procedure TFrmAlocRecurso.BtnAlocClick(Sender: TObject);
var i,j,aloc: integer;
maior,ind,valor: real;
begin
 QryDelRec.ExecSQL;
 QryUsuario.first;
 For i:= 1 to QryUsuario.recordcount do {Rotina de Rodar os Usuarios a serem alocados}
  begin
   QryIndmerec.Close;
   QryIndmerec.parameters.ParamByname('usu').value := Qryusuarioidusu.AsInteger;
   Qryindmerec.Open;
   Qryindmerec.first;
   QryAlocRec.close;
   QryAlocRec.parameters.ParamByname('rec').value:= QryIndMerecidrec.AsInteger;
   QryAlocRec.open;
   aloc:= QryAlocRecalocacao.AsInteger;

   If ( aloc < QryAlocRecCapacidade.AsInteger) then begin     {Aqui procede-se à Alocação}
      QryInsusrec.Parameters.ParambyName('idusu').Value := QryIndMerecidusu.AsInteger;
      QryInsusrec.Parameters.ParambyName('idrec').Value := QryIndMerecidrec.AsInteger;
      QryInsusrec.ExecSQL;
      AltRecAloc.Parameters.ParambyName('aloc').Value := aloc + 1;
      AltRecAloc.Parameters.ParambyName('idrec').Value := QryIndMerecidrec.AsInteger;
      AltRecAloc.ExecSQL;
   end
   else begin      {Aqui é a rotina de Desalocação caso a capacidade do recurso esteja esgotada}
     QryUsuRec.close;
     QryUsuRec.Parameters.ParambyName('rec').Value := QryIndMerecidrec.AsInteger;
     QryUsuRec.open;
     QryUsuRec.first;
     maior:= Qryindmerecindmerec.AsFloat;
     valor:= QryusuRecindmerec.AsFloat;
     if maior > valor  then begin
       Qryaltusurec.Parameters.ParambyName('usu').Value := QryIndMerecidusu.AsInteger;
       Qryaltusurec.Parameters.ParambyName('rec').Value := QryIndMerecidrec.AsInteger;
       Qryaltusurec.Parameters.ParambyName('id').Value := Qryusurecidusu.AsInteger;
       Qryaltusurec.ExecSQL;
       QryAltDesaloc.Parameters.Parambyname('usu').Value:= Qryusurecidusu.value;
       QryAltDesaloc.ExecSQL;
     end
     else begin     {Caso nao aconteça a Desalocação, tenta-se a segunda opção de Alocação em Caso de Recurso já esgotado}
      QryIndMerec.Next;
      For j:= 1 to Qryindmerec.RecordCount do
       begin
         QryAlocRec.close;
         QryAlocRec.parameters.ParamByname('rec').value:= QryIndMerecidrec.AsInteger;
```

```
        QryAlocRec.open;
        aloc:= QryAlocRecalocacao.AsInteger;
        If ( aloc < QryAlocRecCapacidade.AsInteger) then begin
         QryInsusrec.Parameters.ParambyName('idusu').Value := QryIndMerecidusu.AsInteger;
         QryInsusrec.Parameters.ParambyName('idrec').Value := QryIndMerecidrec.AsInteger;
         QryInsusrec.ExecSQL;
         AltRecAloc.Parameters.ParambyName('aloc').Value := aloc + 1;
         AltRecAloc.Parameters.ParambyName('idrec').Value := QryIndMerecidrec.AsInteger;
         AltRecAloc.ExecSQL;
         QryAltAlocRec.Parameters.Parambyname('usu').Value:= Qryusurecidusu.value;
         QryAltAlocRec.ExecSQL;
         break;
        end
        else begin
          Qryindmerec.Next;
        end;
       end;

     end;
  end;
  QryAltAlocRec.Parameters.Parambyname('usu').Value:= QryUsuarioidusu.value;
  QryAltAlocRec.ExecSQL;
  QryUsuario.next;
 end;

 {Aqui é a rotina de Realocação de Usuarios na segunda ou última melhor opção,que ficaram desalocados}
QryRealoc.Close;
QryRealoc.Open;
QryRealoc.First;
 For i:= 1 to QryRealoc.RecordCount do
  begin
   QryIndmerec.Close;
   QryIndmerec.parameters.ParamByname('usu').value :=QryRealocidusu.AsInteger;
   Qryindmerec.Open;
   QryIndmerec.First;
   For j:= 1 to Qryindmerec.RecordCount do
      begin
        QryAlocRec.close;
        QryAlocRec.parameters.ParamByname('rec').value:= QryIndMerecidrec.AsInteger;
        QryAlocRec.open;
        aloc:= QryAlocRecalocacao.AsInteger;
        If ( aloc < QryAlocRecCapacidade.AsInteger) then begin
         QryInsusrec.Parameters.ParambyName('idusu').Value := QryIndMerecidusu.AsInteger;
         QryInsusrec.Parameters.ParambyName('idrec').Value := QryIndMerecidrec.AsInteger;
         QryInsusrec.ExecSQL;
         AltRecAloc.Parameters.ParambyName('aloc').Value := aloc + 1;
         AltRecAloc.Parameters.ParambyName('idrec').Value := QryIndMerecidrec.AsInteger;
         AltRecAloc.ExecSQL;
         QryAltAlocRec.Parameters.Parambyname('usu').Value:= QryRealocidusu.AsInteger;
         QryAltAlocRec.ExecSQL;
         break;
```

```
          end
          else begin
            Qryindmerec.Next;
          end;
        end;
    QryRealoc.Next;
   end;
  QryUsurecal.open;

end;
```

APÊNDICE B - CÓDIGO-FONTE DO ALGORITMO DE GERAÇÃO DOS ÍNDICES *FUZZY*

```
procedure TFrmInfer.BtngerClick(Sender: TObject);
var i,j: integer;
 s: string;
begin
 Screen.Cursor := crhourglass;
 For i:= 0 to High(imerec) do
  begin
    SisSec.Open;
    valor := InfereSLN('A',imerec[i].custo,
         imerec[i].icomp, 0);
    s:= trim(Format('%4.2f', [valor]));
    imerec[i].indmerecaux:= strtofloat(s);
    SisPrin.Open;
    valor := InfereSLN('B',imerec[i].custo,
         imerec[i].icomp, imerec[i].indmerecaux);
    s:= trim(Format('%4.2f', [valor]));
    imerec[i].indmerec:= strtofloat(s);
  end;

  Screen.Cursor := crdefault;
  ShowMessage ('Indices de Merecimento Calculados');

end;


procedure BuscaVariavel(ant : string);
var nvar,j : integer;
Begin
  Form1.Variavel.Open;
  Form1.Variavel.First;
  nvar := Form1.Variavel.RecordCount;
  For j := 1 to nvar do
    If Form1.Variavel.FieldValues['Nome'] <> ant
       then Form1.Variavel.Next
       else Begin codvar := Form1.Variavel.FieldValues['Codigo'];break;end;
  fim := Form1.Variavel.FieldValues['Fim'];
  Form1.Variavel.Close;
End;

procedure BuscaTermo(termo : string);
var ntermos,k : integer;
Begin
  Form1.Termos.Open; Form1.Termos.First;
  ntermos := Form1.Termos.RecordCount;
  For k := 1 to ntermos do
   Begin
```

```
   If (Form1.Termos.FieldValues['Codvar'] <> codvar) or
      (Form1.Termos.FieldValues['Nome'] <> termo)
      then Form1.Termos.Next
      else Begin codtermo := Form1.Termos.FieldValues['Codigo'];
              npontos := Form1.Termos.FieldValues['npontos'] -1;break;
           End;
   End;
  Form1.Termos.Close;
End;

function Ativacao(entrada : real) : real;
var X0,Y0,X1,Y1,cod : integer;
Begin
  Form1.Pontos.Open; Form1.Pontos.First;
  While (Form1.Pontos.FieldValues['codvar'] <> codvar) or
       (Form1.Pontos.FieldValues['codtermo'] <> codtermo) do Form1.Pontos.Next;
  While entrada >= Form1.Pontos.FieldValues['X'] do
    Begin
      X0 := Form1.Pontos.FieldValues['X']; Y0 := Form1.Pontos.FieldValues['Y'];
      If Form1.Pontos.FieldValues['codtermo'] = codtermo
        then Form1.Pontos.Next else break;
    End;
  If entrada >= Fim then ativacao := Form1.Pontos.FieldValues['Y'] else
    Begin
      X1 := Form1.Pontos.FieldValues['X']; Y1 := Form1.Pontos.FieldValues['Y'];
      ativacao := (entrada - X0)* (Y1 - Y0)/(X1 - X0) + Y0;
    End;
  Form1.Pontos.Close;
End;

procedure CalculaInferencia(g : real);
var i : integer;
   a,b,X0,Y0,X1,Y1 : real;
Begin
  a := 0; b := 0; area := 0; xarea := 0;
  Form1.Pontos.Open; Form1.Pontos.First;
  While (Form1.Pontos.FieldValues['codvar'] <> codvar) or
       (Form1.Pontos.FieldValues['codtermo'] <> codtermo) do Form1.Pontos.Next;
  X0 := Form1.Pontos.FieldValues['X']; Y0 := Form1.Pontos.FieldValues['Y'];
  If g <= Y0 then Y0 := g;
  Form1.Pontos.Next;
  For i :=1 to npontos do
    Begin
      X1 := Form1.Pontos.FieldValues['X']; Y1 := Form1.Pontos.FieldValues['Y'];
      If g <= Y1 then Y1 := g;
      a := (Y1-Y0)/(X1-X0); b := Y1 - a * X1;
      area := area +  a/2*(X1*X1-X0*X0) + b*(X1-X0);
      xarea := xarea + a/3*(X1*X1*X1-X0*X0*X0) + b/2*(X1*X1-X0*X0);
      X0 := X1; Y0 := Y1;
      Form1.Pontos.Next;
    End;
```

```
End;
function InfereSLN(sistema : char; entrada1, entrada2,entrada3: real) : real;
var i,j,totreg : integer;
   var1,var2,var3,grau,areacumul,xareacumul : real;
   termo, ant1,ant2,ant3,cons : string;

Begin
 SetLength( conseq,3);

 totreg:= 0;
 Case sistema of
  'A': Begin
        ant1 := 'custosec';
        ant2 := 'indcompet';
        cons := 'indmerecsec';
        totreg:= FrmInfer.SisSec.RecordCount;
        conseq[0].term:= 126;
        conseq[0].grat:= 0;
        conseq[1].term:= 127;
        conseq[1].grat:= 0;
        conseq[2].term:= 128;
        conseq[2].grat:= 0;
        SetLength( auxcons,12);
        FrmInfer.SisSec.First;
       End;
  'B': Begin
        ant1 := 'custoprin';
        ant2 := 'indcompprin';
        ant3 := 'indmerecaux';
        cons := 'indmerecprin';
        totreg:= FrmInfer.SisPrin.RecordCount;
        conseq[0].term:= 120;
        conseq[0].grat:= 0;
        conseq[1].term:= 121;
        conseq[1].grat:= 0;
        conseq[2].term:= 122;
        conseq[2].grat:= 0;
        FrmInfer.SisPrin.First;
        SetLength( auxcons,36);
       End;

 End;
 areacumul := 0; xareacumul := 0;
 For i := 1 to totreg do
  Begin
   BuscaVariavel(ant1);
   Case sistema of
    'A': termo := Frminfer.SisSec.FieldValues['custosec'];
    'B': termo := Frminfer.SisPrin.FieldValues['custoprin'];
    {'C': termo := Frminfer.RegrasSLNC.FieldValues['valor'];}
   End;
```

```
BuscaTermo(termo);
var1 := Ativacao(entrada1);

BuscaVariavel(ant2);
Case sistema of
  'A': termo := Frminfer.SisSec.FieldValues['indcompet'];
  'B': termo := Frminfer.SisPrin.FieldValues['indcompprin'];
  {'C': termo := Form2.RegrasSLNC.FieldValues['depreciacao'];}
End;
BuscaTermo(termo);
var2 := Ativacao(entrada2);
Case sistema of
  'A': begin
         if var1 <= var2 then grau := var1 else grau := var2;
       end;

  'B': begin
         termo := Frminfer.SisPrin.FieldValues['indmerecaux'];
         BuscaTermo(termo);
         var3:= Ativacao(entrada3);
         BuscaVariavel(ant3);
         if ((var1 <= var2) and (var1 <= var3)) then grau := var1
         else if ((var2 <= var1) and (var2 <= var3)) then grau := var2
         else if ((var3 <= var1) and (var3 <= var2)) then grau := var3;
       end;
end;

{if var1 <= var2 then grau := var1 else grau := var2; }
BuscaVariavel(cons);
Case sistema of
  'A': termo := Frminfer.SisSec.FieldValues['indmerecsec'];
  'B': termo := Frminfer.SisPrin.FieldValues['indmerecprin'];

 { 'C': termo := Form2.RegrasSLNC.FieldValues['valor_final'];}
End;
BuscaTermo(termo);
{ CalculaInferencia(grau);
areacumul := areacumul + area; xareacumul := xareacumul + xarea;
if areacumul = 0 then InfereSLN := 0 else InfereSLN :=xareacumul/areacumul; }
Case sistema of
  'A': begin
         FrmInfer.Sissec.Next;
         auxcons[i-1].term:= codtermo;
         auxcons[i-1].grat:= grau;
       end;
  'B': begin
         FrmInfer.SisPrin.Next;
         auxcons[i-1].term:= codtermo;
         auxcons[i-1].grat:= grau;
       end;
  {'C': Form2.RegrasSLNC.Next;}
```

```
    End;
  End;  {Fim Loop totregras}

  for i:= 0 to High(conseq) do
   begin
    for j:= 0 to High(auxcons) do
     begin
      if (conseq[i].term = auxcons[j].term) and (auxcons[j].grat > conseq[i].grat)
        then begin
          conseq[i].grat:= auxcons[j].grat;
        end;
     end;
   end;

  for i:= 0 to High(conseq)  do
   begin
    if (conseq[i].grat >= 0.01) then begin
     codtermo:= conseq[i].term;
     CalculaInferencia(conseq[i].grat);
    end;
     areacumul := areacumul + area; xareacumul := xareacumul + xarea;
     if areacumul = 0 then InfereSLN := 0 else InfereSLN :=xareacumul/areacumul;
   end;
End;
```