UNIVERSIDADE FEDERAL DE SANTA CATARINA

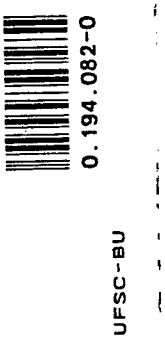
DEPARTAMENTO DE ENGENHARIA DE PRODUÇÃO E SISTEMAS

PROGRAMAÇÃO DINAMICA DIFUSA

DISSERTAÇÃO SUBMETIDA À UNIVERSIDADE FEDERAL DE SANTA CATARINA

PARA OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA

SAMUEL VIEIRA CONCEIÇÃO



FLORIANÓPOLIS

SANTA CATARINA - BRASIL

MAIO DE 1989

PROGRAMAÇÃO DINAMICA DIFUSA

SAMUEL VIEIRA CONCEIÇÃO

ESTA DISSERTAÇÃO FOI JULGADA ADEQUADA PARA OBTENÇÃO DO TITULO DE:

"MESTRE EM ENGENHARIA"

ESPECIALIDADE ENGENHARIA DE PRODUÇÃO E APROVADA EM SUA FORMA FINAL PELO PROGRAMA DE PÓS-GRADUAÇÃO

RICARDO MIRANDA BARCIA, Ph. D
COORDENADOR DO PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA DE PRODUÇÃO

BANCA EXAMINADORA:

RICARDO MIRANDA BARCIA , Ph. D. - Presidente

SÉRGIO FERNANDO MAYERLE, M. Eng.

ANTONIO SÉRGIO COELHO, M. Eng.

Aos meus queridos pais e irmãos

À Rose

AGRADECIMENTOS

Agradeço a todos aqueles que, de forma direta ou indireta, colaboraram para a realização deste trabalho, e em especial:

Ao Prof. Sérgio Fernando Mayerle, pela eficiente orientação fornecida no decorrer deste trabalho.

Aos Profs. Ricardo Miranda Barcia e Antonio Sérgio Coelho pela motivação e pelas sugestões fornecidas.

Ao apoio financeiro do CNPq.

Aos colegas, professores e funcionários do Departamento de Engenharia de Produção e Sistemas da UFSC, pelo apoio, convívio e aprendizado.

A programação dinâmica é uma técnica de pesquisa operacional muito utilizada para resolver problemas de economia, engenharia, planejamento e controle da produção, entre outros, que sejam modelados de forma precisa.

Com o surgimento da teoria dos conjuntos difusos, foi possível proporcionar ferramentas matemáticas mais adequadas ao tratamento dos problemas que envolvessem incertezas e/ou conceitos imprecisos.

O presente trabalho apresenta uma técnica que concatena a teoria dos conjuntos difusos e programação dinâmica, com o objetivo de resolver problemas de programação dinâmica definidos a partir de parâmetros imprecisos e/ou subjetivos.

Após o desenvolvimento da técnica, faz-se uma aplicação da mesma, para um problema de expansão de linhas de produção, e para um problema de marketing, caracterizando os casos de programação dinâmica determinística difusa e programação dinâmica estocástica difusa, respectivamente.

ABSTRACT

vi

Dynamic programming is a technique of operations research often used to solve appropriately formulated problemas in, among other areas, Economics, Engineering and Productions Planning and Control.

With the appearance of theory of Fuzzy Sets, it became possible to provide mathematical tools more suitable for problems involving uncertainty and inexact concepts.

The present work presents a mathematical too which links the theory of Fuzzy Sets and Dynamic programming . The objective of thé work is to solve dynamic programming problems defined or modeled through fuzzy concepts or values.

After the development of the tool, an applications is performed for a problem of expansion of production lines and a marketing problem. The former characterizes the case of Deterministic fuzzy Dynamic Programming and the latter fuzzy Stochastic Dynamic Programming.

SUMÁRIO

pág

LISTA DE TABELAS

pág

CAPÍTULO 1

1. - Introdução

1.1 - Origem do Trabalho

Bellman [3] desenvolveu, em 1956, um algoritmo recursivo para resolver problemas de decisão seqüêncial. Esse algoritmo, tido como uma das principais ferramentas para resolver problemas de engenharia, economia, e planejamento e controle da produção, que se enquadram como problemas de programação dinâmica, requer que os dados sejam bem definidos e não contenham informações e critérios de decisão subjetivos.

Por outro lado, Zadeh [25], em 1965, propos uma nova abordagem para a teoria dos conjuntos, a qual deu-se o nome de teoria dos conjuntos difusos. Baseados nessa nova teoria, vários autores propuseram revisões dos conceitos matemáticos clássicos, com o objetivo de proporcionarem ferramentas adequadas ao tratamento dos problemas que envolvem incertezas ou conceitos imprecisos.

Muitos problemas de planejamento e controle da produção, de engenharia e economia, entretanto, trazem em sua estrutura, informações e/ou parâmetros medidos, definidos ou obtidos de forma não muito clara e precisa, que impedem a utilização da matemática clássica como instrumento eficiente de análise.

O presente trabalho origina-se, portanto, da necessidade de se adaptar a técnica proposta por Bellmann, considerando informações e /ou parâmetros definidos ou medidos de forma não muito precisa, com o objetivo de melhor modelar o mundo real.

1.2 - Objetivo do Trabalho

O objetivo principal deste trabalho é desenvolver uma técnica para ser utilizada como suporte no auxílio às decisões, em problemas de programação dinâmica determinística e estocástica, que envolvam incertezas e/ou conceitos imprecisos.

1.3 - Importância do Trabalho

A utilização de programação dinâmica para resolver problemas de engenharia, planejamento e controle da produção, economia, entre outros, data de 1956, contemplando tanto os problemas determinísticos, quanto os problemas estocásticos.

Entretanto, nos casos onde os problemas têm uma estrutura que permite enquadrá-los como um problema de programação dinâmica, mas dentro dessas estruturas existem aspectos subjetivos, a resolução dos mesmos não pode ser feita pelos algoritmos clássicos.

Nesse sentido, a técnica de programação dinâmica proposta avalia problemas de programação dinâmica determinísticos e estocásticos, definidos e/ou medidos de forma subjetiva e não muito precisa, descritos sob a forma de um texto em linguagem natural. Tal descrição do problema é decorrente do julgamento humano de ações, fatos ou previsões. A técnica proposta permite, ainda, considerar a qualidade técnica das soluções obtidas, também sob forma de linguagem natural.

Nesse contexto a técnica proposta tem a vantagem de oferecer ao tomador de decisões um leque de opções, obtidos segundo o grau de subjetividade das informações.

1.4 - Limitações do Trabalho

No desenvolvimento do presente trabalho não se considerou os casos markovianos com estágios infinitos.

1.5 - Estrutura do Trabalho

O presente trabalho foi dividido em seis capítulos.

Este primeiro capítulo visa apresentar a origem do trabalho, definir seus objetivos, bem como sua importância e limitações.

O segundo capítulo pode ser dividido em duas partes : na primeira parte, apresentam-se as definições, terminologia e o algoritmo iterativo de programação dinâmica referente ao caso determinístico. Na segunda parte são abordados os conceitos e propriedades dos processos de decisão de Markov e sua extensão à programação dinâmica estocástica.

No terceiro capítulo apresentam-se as definições e propriedades da teoria dos conjuntos difusos, dando ênfase às operações com números difusos e ao princípio da extensão.

No quarto capítulo é enfocado a programação dinâmica difusa, que é a técnica proposta.

No quinto capítulo apresentam-se as aplicações da técnica proposta, para os casos determinísticos e estocásticos.

Por fim são apresentadas, no sexto capítulo, as conclusões e recomendações.

CAPÍTULO II

2. Programação Dinâmica

2.1 - Introdução

A programação dinâmica é uma técnica de otimização utilizada na solução de problemas cuja estrutura pode ser formulada como uma seqüência de decisões.

Para o tomador de decisões é muito importante dispor de técnicas para resolução de problemas que analise todas as possibilidades e situações envolvidas no problema e identifique a(s) mais viável(eis) ou a(s) melhor(res) de todas as alternativas analisadas, determinando, assim, um plano ótimo de ações.

A otimização dos sistemas gerenciais como o planejamento e controle da produção, a política ótima de um sistema de controle de estoques, a manutencão ou substituição de equipamentos tem sido imperiosa dentro de muitas estruturas organizacionais, e são exemplos clássicos da aplicação desta técnica.

A redução dos custos dentro das empresas, como forma de manter seus produtos e/ou serviços competitivos no mercado, pode, às vezes, ser visto como um problema de otimização o qual exige a aplicação de programação dinâmica. A estrutura desses problemas exige diferentes considerações e formulações matemáticas. No presente trabalho consideram-se os casos de programação dinâmica determinística e programação dinâmica

estocástica (markoviana finita).

2.2 - Terminologia

Hastings [15], define a seguinte terminologia para a solução de um problema de programação dinâmica, através da utilização de equações recursivas:

1 - Estado : representa a configuração de um sistema e é identificado por um rótulo que caracteriza as propriedades correspondentes a esse estado;

2 - Estágio : pode ser definido como um passo que representa a transição de um sistema de um estado a um outro estado adjacente;

3 - Ação : é uma alternativa que pode ser selecionada para realizar a transição de um estado a um outro estado adjacente;

4 - Valor de Estado : é uma função que representa a soma dos retornos associados a um estado ou, genericamente, a uma dada política;

5 - Retorno : é uma função que caracteriza, por exemplo, o lucro esperado ou os custos atribuidos ou ainda o consumo de recursos.

As variáveis ou parâmetros definidos para estas equações são:

- Estágio = n;
- Estado = (n,i);
- Ação = k;
- Retorno = r(n,i,k);
- Valor ótimo de estado = f(n,i).

Com base nessa terminologia, apresenta-se a seguir, os casos de programação dinâmica determinística e estocástica.

2.3 - Programação Dinâmica Determinística

Os problemas cujas variáveis e parâmetros são determinísticos caracterizam o caso de programação dinâmica determinística.

Os problemas de programação dinâmica são formulados matemáticamente e resolvidos segundo o princípio da otimalidade, o qual afirma que uma política ótima deve ser tal que, qualquer que seja o estado e a decisão inicial, as decisões seguintes devem constituir uma política ótima em relação ao estado resultante da primeira decisão.

Para o caso determinístico o algoritmo iterativo utiliza os seguintes parâmetros, para um problema de maximização:

1 - Equação de recorrência :

$$F(n,i) = \max_{k \in k_{ni}} [\, r(n,i,k) + F(n-1, j) \,] \qquad (2.1)$$

onde $F(0,i)$ = condições de contorno conhecidas para o problema.

2 - Equação de Transição :

$$j = t(n,i,k) \qquad (2.2)$$

A solução do problema consiste, portanto, na resolução da recorrência, tendo como ponto de partida as condições de contorno conhecidas.

2.4 - Programação Dinâmica Estocástica

2.4.1 - Processos Markovianos

Existem vários problemas de programação dinâmica onde as transições entre estados e retornos são regidos por leis probabilisticas.

Nesses casos o problema se caracteriza por possuir em sua estrutura seqüências de variáveis aleatórias. A influência dos efeitos aleatórios se dá por todo o intervalo de

tempo.

Os processos que envolvem uma seqüência de variáveis aleatórias são definidos como Processos Estocásticos. Os estudos dos processos estocásticos foram feitos pelo matemático russo A. A. Markov, que desenvolveu um modelo para resolver várias classes de problemas que envolviam processos estocásticos. Esses modelos são conhecidos como Processos Markovianos.

No presente trabalho considera-se apenas os casos de processos markovianos de estágios finitos.

A estrutura de probabilidade de uma seqüência aleatória de parâmetros discretos pode ser determinada através dos conceitos de probabilidades conjuntas, onde para todo n finito e para toda seqüência $j_0, j_1, \ldots, j_n$ de estados tem-se:

$$P(j_0, j_1, \ldots, j_k) = P[X_0 = j_0,\ X_1 = j_1, \ldots, X_n = j_n] \qquad (2.3)$$

Este processo é definido como um processo markoviano se a probabilidade condicional após o sistema estar em um determinado estado após n passos, for a mesma conhecendo-se todos os estados do sistema em todos os passos anteriores ou sómente a probabilidade condicional advinda do conhecimento do estado em um passo imediatamente anterior.

Isto significa que o fato de se conhecer $X_0, X_1, \ldots, X_{n-1}$ não adiciona nenhuma informação relevante no que

se refere ao cálculo do valor de X_n. Apenas o conhecimento do valor de X_{n-1} é suficiente para calcular o valor de X_n, isto é, a probabilidade de que a n-ésima variável aleatória X_n assuma o valor x_n, depende basicamente do conhecimento do valor de X_{n-1}.

Formalmente tem-se:

$$P\,[X_n = x_n \,|\, X_{n-1} = x_{n-1}] = P\,[\, X_n = x_n \mid X_1 = x_1, X_2 = x_2, \ldots, X_{n-1} = x_{n-1}\,] \qquad (2.4)$$

Considerando-se um sistema que pode se encontrar em um número finito de estados, cuja enumeração pode ser dada por $i = 0,1,2,..,n$ e admitindo-se que nos tempos $t_o, t_1, \ldots, t_n$ o sistema passe de forma aleatória de um estado a um outro estado adjacente, pode-se definir, nesse processo, uma matriz de transição

$$P = (p_{ij}) \qquad (2.5)$$

onde : p_{ij} = probabilidade de que o sistema se encontre no estad
j no tempo (t + 1) , dado que ele estava no estado
i no tempo t;

P = matriz de transição do processo.

Considerando-se que a matriz de transição P independe do tempo, pode-se expressá-la, em termos das probabilidades de transição de 1 passo, através da seguinte matriz:

$$P=(P_{ij})=\begin{bmatrix} P_{00} & P_{01} & P_{02} \cdots & P_{0n} \\ P_{10} & P_{11} & P_{12} \cdots & P_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ P_{n0} & P_{n1} & P_{n2} \cdots & P_{nn} \end{bmatrix} \qquad (2.6)$$

Toda matriz que satisfizer a condição de possuir entradas não negativas com soma de linhas igual a 1, pode ser definida como uma matriz estocástica.

As probabilidades de transição de n-passos $p^{(n)}_{ij}$ pode ser definida por:

$$p^{(n)}_{ij} = p\,[\, x_{k+n} = j \mid x_k = i \,], \qquad (2.7)$$

$$i,j = 0,1,2,\ldots$$

$$n \geq 0, \text{ onde:}$$

$p^{(n)}_{ij}$ é a probabilidade condicional de se

estar no estado j dado que o processo estava no estado i, em n passos anteriores.

A probabilidade incondicional de que o processo esteja no estado j após n passos, pode ser definida por :

$$p^{(n)}_{j} = p [x_{n} = j] \qquad (2.8)$$

$p^{(n)}_{j}$ é uma probabilidade marginal e representa a probabilidade de se estar no estado j após n passos, independente do estado inicial.

A hipótese de que a matriz de transição P independe do tempo, configura um caso estacionário.

Os problemas considerados nesse trabalho abordam sistemas que passam por processos markovianos que são caracterizados por estágios.

Quando n estágios ainda devem ser alcançados, o sistema pode estar em qualquer dos estados (n,1), (n,2),...(n,N).

Para a seqüência de variáveis aleatórias $...X_{n}$, X_{n-1}, ... o dominio de X_{n} é o conjunto { 1, 2, ..., N }.

Define-se a probabilidade condicional p $[X_{n-1} = j \mid X_{n} = i]$ como sendo a probabilidade que o sistema vá para o estado (n - 1,j), dado que ele estava no estado (n,i) no

estágio n.

Formalmente pode-se escrever:

$$p(n,i,j) = p(X_{n-1} = j \mid X_n = i) \qquad (2.9)$$

onde: n = estágio;

i = estado;

j = ação.

A probabilidade de transição p(n,i,j) apresenta as seguintes propriedades:

1. $0 \leq p(n,i,j) \leq 1$; (2.10)

2. $\sum_{j=1}^{N} p(n,i,j) = 1$ (2.11)

No caso em que as probabilidades de transição são estacionárias em n , a probabilidade de transição dada por p (n,i,j) será denotada por p (i,j).

2.4.2 - Processos de Decisão de Markov com Retornos Associados

No inicio desse capítulo, definiu-se que, em um processo markoviano , a probabilidade que uma variável aleatória X_n assumisse um valor x_n, dependia apenas do estado imediatamente predecessor, isto é, do conhecimento do valor de x_{n-1} .

Quando um sistema evolui de um estado (n,i) a um outro estado (n-1,j), pode-se associar a ele um retorno C(n,i,j). Como esse retorno está associado com a transição particular entre dois estados quaisquer, ele é definido como retorno de transição.

Denotando-se o valor de estado por uma função F(n,i), que representa o valor esperado do retorno que é gerado quando o estado é um estado inicial, o valor F(n,i) quando o sistema passa de um estado (n,i) para um estado (n-1,j), através de uma probabilidade de transição p(n,i,j), pode ser calculado por:

$$F(n,i) = \sum_{j=1}^{N} p(n,i,j).C(n,i,j) + \sum_{j=1}^{N} p(n,i,j).F(n-1,j) \qquad (2.12)$$

ou

$$F(n,i) = \sum_{j=1}^{N} p(n,i,j).[C(n,i,j) + F(n-1,j)] \qquad (2.13)$$

Associado ao estado (n,i) existe um retorno de estágio, denotado por r(n,i), que representa o retorno esperado gerado no estágio corrente, considerando-se que o sistema iniciou no estado (n,i).

Como ao estágio considerado existe uma probabilidade e um retorno associado, para o caso genérico pode-se definir o retorno de estágio por:

$$r(n,i) = \sum_{j=1}^{N} p(n,i,j).C(n,i,j) \qquad (2.14)$$

Levando (2.14) em (2.12) tem-se:

$$F(n,i) = r(n,i) + \sum_{j=1}^{N} p(n,i,j) + F(n-1,j) \qquad (2.15)$$

Se ao estado (n,i) estiver associado um conjunto de ações K_{ni}, e se uma ação particular k for considerada, então a probabilidade de transição do estado (n,i) para o estado (n-1,j) pode ser definida por p(n,i,j,k).

A esse estado também vai está associado um retorno de transição C(n,i,j,k).

O retorno de transição r(n,i,k) quando o sistema passa do estado (n.i) para o estado (n-1,j) é definido

por:

$$r(n,i,k) = \sum_{j=1}^{N} p(n,i,j,k).\ C(n,i,j,k) \qquad (2.16)$$

Os problemas cuja formulação matemática exige a aplicação de processos markovianos, pode ser resolvido computacionalmente através do algoritmo iterativo.

Em termos computacionais, o cálculo do valor de estado ótimo de um problema de maximização, pode ser resolvido através da aplicação da programação estocástica, utilizando-se a seguinte equação de recorrência:

$$F(n,i) = \max_{k \in k_{ni}} \left[r_1(n,i,k) + \sum_{j=1}^{N} p(n,i,j,k)\left[r_2(n.i.j,k) + F(n-1,j) \right] \right] \qquad (2.17)$$

Na equação acima, o parâmetro r_1 independe dos processos aleatórios e o parâmetro r_2, não.

A aplicação da programação dinâmica estocástica pode ser feita em problemas como a escolha de uma política ótima de um sistema de estoques, manutenção ou substituição de equipamentos sob condições de incerteza ou a problemas de marketing, onde a decisão de fazer ou não publicidade pode afetar a posição de um produto frente aos seus concorrentes.

2.5 Condições de Validação dos Modelos

Para se fazer uso da técnica de programação dinâmica, faz-se necessário que as condições de validação dos modelos, isto é, as condições de separabilidade e as condições de otimalidade sejam satisfeitas.

Considere-se um sistema que em um dado estágio evolui do estado (n,i_n) para o estado $(n-1,i_{n-1})$, sob uma ação k_n, e gera o retorno $r(n,i_n,k_n)$. Considere-se, ainda, que A_n é um plano genérico que determina a seqüência de ações $k_n,k_{n-1},\ldots,k_1$. Suponha-se que todo o processo tenha m estágios, e que o objetivo seja maximizar a função ϕ_m de retornos dos estágios, isto é, que se deseja obter o máximo valor de $F(n,i_m)$, definido por:

$$F(m,i_{im}) = \max_{A_m \in W_m} [\phi_m(r(m,i_m,k_m),\ldots,r(n,i_n,k_n),\ldots,r(1,i_1,k_1))], \qquad (2.18)$$

onde w_m é o conjunto de todos os planos que iniciam no estado (m,i_m). Para que a função ϕ_m seja maximizada, as condições de separabilidade e otimalidade, discutidas abaixo, devem ser satisfeitas.

2.5.1 - Condição de Separabilidade

A condição de separabilidade possibilita a aplicação de recursividade dado um plano fixo.

Definição 2.1 - Bellman [03] : Para todo plano, o valor de cada estado pode ser necessariamente calculado como uma função do retorno do estágio e do valor do estado subsequente.

Seja o valor de estado (m,i_m), sob um plano A_m denotado por $F(m,i_m,A_m)$. Então:

$$F(m,i_m,A_m) = \phi_m(r(m,i_m,k_m),\ldots,r(n,i_n,k_n),\ldots,r(1,i_1,k_1)) \qquad (2.19)$$

Se a condição de separabilidade se aplica, então, para cada estado (n,i_n) e um dado plano A_n pode-se escrever a equação (2.19) na forma:

$$F(n,i_n,A_n) = \Phi_n(r(n,i_n,k_n),F(n-1,i_{n-1},A_{n-1})), \qquad (2.20)$$

onde:

$$F(n-1,i_{n-1},A_{n-1}) = \phi_{n-1}(r(n-1,i_{n-1},k_{n-1}),\ldots,r(1,i_1,k_1)) \qquad (2.21)$$

sendo ϕ e Φ funções apropriadas.

2.5.2 - Condição de Otimalidade

Considere-se novamente o problema apresentado na equação (2.18). A condição de separabilidade requer que o valor de cada estado possa ser calculado recursivamente para um dado plano. No processo iterativo não só se calculam os valores recursivamente como também se descartam sub-planos em cada estado. Para que isto tenha validade a condição de otimalidade, descrita abaixo, deve ser satisfeita:

Definição 2.2 - Bellman [03] : Para cada estado e ação, o plano ótimo deve consistir de uma determinada ação seguida do plano que é ótimo para o estado sucessor.

A seguir, apresenta-se, uma formulação algébrica da condição de otimalidade. Considerando-se que pela condição de separabilidade tem-se, para cada plano A_n:

$$F(n,i_n,A_n) = \Phi\,(r(n,i_n,k_n),F(n-1,i_{n-1},A_{n-1}))$$

pode-se dizer que um plano A_n consiste de uma ação k_n, seguida de um plano A_{n-1} do estágio n-1, isto é:

$$A_n = k_n + A_{n-1} \qquad (2.22)$$

Tem-se então:

$$F(n,i_n,A_n) = F(n,i_n,k_n + A_{n-1}) \qquad (2.23)$$

Se A°_{n-1} é um plano ótimo para o estado $(n-1,i_{n-1})$, sucessor de (n,i_n) dado uma ação k_n, e assumindo-se maximização, a condição de otimalidade requer que para qualquer estado (n,i_n), ação k_n e plano A_{n-1}, se tenha:

$$F(n,i_n,k_n+A^{\circ}_{n-1}) \geq F(n,i_n,k_n+A_{n-1}) \qquad (2.24)$$

A equação (2.24) acima, garante a condição de otimalidade para o caso de maximização. Para o caso de minimização se teria:

$$F(n,i_n,k_n+A^{\circ}_{n-1}) \leq F(n,i_n,k_n+A_{n-1}) \qquad (2.25)$$

CAPÍTULO III

3. - Conjuntos Difusos

3.1 - Introdução

O conceito do termo difuso utilizado neste trabalho, difere do conceito usual. Um sinônimo para o termo difuso é a palavra vago.

Para a expressão conjunto difuso, a literatura registra duas interpretações distintas.

Segundo Zadeh [25], o termo difuso é utilizado em situações nas quais um conjunto A, definido sobre um universo de discurso X, não possui limites bem definidos. Por exemplo, o conjunto difuso A pode representar o conjunto de homens altos de uma comunidade (universo) X. De fato, não existe um limite preciso e bem definido que diferencie os homens de estatura alta dos de estatura mediana, os quais não são perfeitamente definidos.

Sugeno [23], por outro lado, propõe a utilização do termo difuso em um contexto radicalmente diferente, onde uma medida difusa é usada para caracterizar a probabilidade, ou a possibilidade ou o grau de credibilidade subjetiva que se tem no fato de um elemento x pertencer a um conjunto A. Um exemplo seria a genuinidade atribuida a uma obra de arte. Embora o

conceito de genuinidade seja preciso (verdadeiro ou falso), um grau de credibilidade (medida difusa) poderá ser associado à sentença "a obra de arte é verdadeira", refletindo a convicção que se tem no fato de ao se adquirir uma obra de arte, esta pertencer ao conjunto de obras genuínas.

Em ambas interpretações, tais medidas refletem índices de tendência, designadas de forma subjetiva por um individuo ou grupo de individuos, dependendo ainda do contexto.

3.2 - Definições Básicas

Para melhor compreensão da estrutura matemática e computacional utilizada nesse trabalho, são apresentadas abaixo, as definições e conceitos advindos do desenvolvimento da teoria dos conjuntos difusos.

Definição 1 - (Zadeh [25]) : Seja X um conjunto clássico de objetos chamado universo, cujos elementos genéricos são denotados por x. A função de pertinência de um elemento x em um subconjunto clássico $A \subseteq X$ é uma função característica $\mu_A(x)$, $x \rightarrow \{0,1\}$, tal que:

$$\mu_A(x) = \begin{cases} 1 & \text{se e somente se } x \in A \\ 0 & \text{se e somente se } x \notin A \end{cases} \qquad (3.1)$$

onde ⟨0,1⟩ é definido como conjunto de avaliação. Se o conjunto de avaliação for o intervalo real [0,1], então A é um conjunto difuso.

O termo $\mu_A(x)$ representa o grau de pertinência de x em relação a A, onde A é um subconjunto de X que não tem fronteira bem definida e pode ser caracterizado através da seguinte notação:

$$A = \{ (x, \mu_A(x)), x \in X \}$$

A definição acima comporta as seguintes considerações :

1 - Quanto mais próximo de 1 for $\mu_A(x)$, maior será a pertinência de x em relação ao conjunto difuso A;

2 - O conjunto universo X nunca é difuso.

No desenvolvimento da teoria dos conjuntos difuso, diversos autores apresentaram várias alternativas para expressar a representação dos conjuntos difusos. Zadeh [28], em 1972, propos uma notação bastante conveniente para se representar conjuntos difusos: Quando X é definido como um

conjunto discreto $\{x_1, x_2, \ldots, x_n\}$, o conjunto difuso $A \subseteq X$ poderá ser expresso como:

$$A = \mu_A(x_1) / x_1 + \ldots + \mu_A(x_n) / x_n = \sum_i^n \mu_A(x_i) / x_i \qquad (3.2)$$

Na expressão acima os elementos com grau de pertinência nulo podem ser omitidos e (3.2) pode ser utilizada para caracterizar conjuntos infinitos com suportes discretos, de

Quando X é um conjunto contínuo, o conjunto difuso $A \subseteq X$, poderá ser expresso como:

$$A = \int_X \mu_A(x) / x \qquad (3.3)$$

Definição 2 - (Zadeh [25]) : Se A e B são conjuntos difusos definidos em um universo X, então:

i) O suporte de A é um sub-conjunto ordinário de X caracterizado por:

$$\text{Supp } A = \{ x \in X \mid \mu_A(x) > 0 \} \qquad (3.4)$$

ii) O conjunto A é dito normalizado se, e somente se, $\exists$ $x \in X$ tal que :

$$0 \leq \mu_A(x) \leq 1 \qquad (3.5)$$

iii) $\emptyset$ é um conjunto vazio, definido para qualquer $x \in X$, tal que:

$$\mu_{\emptyset}(x) = 0 \qquad (3.6)$$

Isso implica que para qualquer $x \in X$

$$\mu_x(x) = 1 \qquad (3.7)$$

iv) Dois conjuntos difusos, A e B são iguais, e denota-se A = B, se, e somente se :

$$\mu_A(X) = \mu_B(x) \quad \text{para qualquer } x \in X \qquad (3.8)$$

v) A altura de um conjunto difuso A é definida por:

$$hgt(A) = Supp_{x \in X}\, \mu_A(x) \qquad (3.9)$$

3.2.1 - União e Intersecção de Conjuntos Difusos

A clássica notação utilizada para caracterizar os termos união (∪) e intersecção (∩) de subconjuntos ordinários de X também pode ser extendida à teoria dos conjuntos difusos.

A formulação dada abaixo, para caracterizar a união e intersecção de conjuntos difusos é atribuida a Zadeh [25], 1965.

Definição 3.3 - (Zadeh[25]): Sejam A e B conjuntos difusos definidos em um universo X. Então:

i) O conjunto união, denotado por A ∪ B, é definido, para qualquer x ∈ X, pela seguinte função de pertinência:

$$\mu_{A \cup B}(x) = \max \{ \mu_A(x), \mu_B(x) \} \qquad (3.10)$$

ii) O conjunto intersecção, denotado por A ∩ B, é definido, para qualquer x ∈ X, pela função de pertinência:

$$\mu_{A}(x) = \min \{ \mu_{A}(x) , \mu_{B}(x) \} \qquad (3.11)$$

As propriedades matemáticas das equações (3.10) e (3.11), propostas por Zadeh, foram estudadas por Bellman e Giertz[04].

Definição 3.4 - (Zadeh [25]) O complementar $\bar{A}$ de um conjunto difuso é definido pela função pertinência:

$$\mu_{\bar{A}}(x) = 1 - \mu_{A}(x), \text{ para qualquer } x \in X \qquad (3.12)$$

A justificativa de (3.12) também é atribuida a Bellman e Giertz[04].

Para as operações de união, intersecção e complementar (∪ ,∩ e $^{-}$), apresentadas nas definições acima, tem-se, segundo Dubois e Prade[10], as seguintes propriedades:

a) Comutatividade

$A \cup B = B \cup A;$

$A \cap B = B \cap A;$

b) Associatividade

$A \cup (B \cup C) = (A \cup B) \cup C ;$

$A \cap (B \cap C) = (A \cap B) \cap C ;$

c) Idempotência

$A \cup A = A ;$

$A \cap A = A ;$

d) Distributividade

$A \cup (B \cap C) = (A \cup B) \cap (A \cup C);$

$A \cap (B \cup C) = (A \cap B) \cup (A \cap C);$

e) $A \cap \emptyset = \emptyset ;$

$A \cup X = X ;$

f) Identidade

$A \cup \emptyset = A;$

$A \cap X = A;$

g) Absorção

$A \cup (A \cap B) = A ;$

$A \cap (A \cup B) = A ;$

h) Involução

$\bar{\bar{A}} = A$;

i) Fórmula de Equivalência

$(\bar{A} \cup B) \cap (A \cap \bar{B}) = (\bar{A} \cap \bar{B}) \cup (A \cap B)$;

j) Fórmula de Diferença Simétrica

$(\bar{A} \cap B) \cup (A \cap \bar{B}) = (\bar{A} \cup \bar{B}) \cap (A \cup B)$.

Embora os conjuntos difusos satisfaçam as propriedades (a) a (j) apresentadas acima, válidas também para os conjuntos ordinários, deve-se ressaltar as propriedades abaixo por serem válidas apenas para os conjuntos ordinários.

$A \cap \bar{A} = \emptyset$;

$A \cup \bar{A} = X$;

3.2.2 - Conjunto de Nível α e Cardinalidade de Conjuntos Difusos

A representação dos elementos de um conjunto difuso A que possuam grau de pertinência maior, igual ou menor que um determinado valor α, pode ser feita através das noções de conjunto de nivel α.

Por outro lado, a cardinalidade de conjuntos difusos é muito importante na caracterização da proporção dos elementos de X que estão em A.

Definição 3. 5 - (Zadeh [25]): Seja A um conjunto difuso definido sobre o universo X. Seja $\alpha \in [0.1]$. Então o conjunto nível A_α pode ser definido por:

$$A_\alpha = \{ x \in X \mid \mu_A(x) \geq \alpha \} \qquad (3.13)$$

A função de pertinência de um conjunto difuso A pode ser expressa, em termos de funções características de seus niveis-α, através das seguintes expressões:

$$\mu_A(x) = \text{Sup}_{\alpha \in]0,1]} \min (\alpha , \mu_{A_\alpha}(x)) \qquad (3.14)$$

onde:

$$\mu_{A_\alpha}(x) = \begin{cases} 1 \text{ se, e somente se } x \in A_\alpha ; \\ 0, \text{ para outros valores} \end{cases} \qquad (3.15)$$

As expressões (3.14) e (3.15) permitem concluir que:

$$(A \cup B)_{\alpha} = A_{\alpha} \cup B_{\alpha} \qquad (3.16)$$

$$(A \cap B)_{\alpha} = A_{\alpha} \cap B_{\alpha} \qquad (3.17)$$

3.2.3 - Cardinalidade de Conjuntos Difusos

Definição 3.6 - (Zadeh [25]): A cardinalidade $| A |$ de um conjunto difuso A, definido em um universo X, é:

i) para X finito, definido como:

$$| A | = \sum_{x \in X} \mu_{A}(x) \qquad (3.18)$$

ii) Para X não finito, porém se A tem um suporte finito, $| A |$ é definido como:

$$| A | = \sum_{x \in \text{Supp } A} \mu_{A}(x) \qquad (3.19)$$

iii) Não definido se Supp A é não finito

3.2.4 - Conjuntos Difusos Convexos e Estruturas Difusas

Zadeh, [25], 1965, extendeu a noção de convexidade aos conjuntos difusos, assumindo que um determinado conjunto difuso de um universo X estivesse sobre um espaço euclidiano N-dimensional.

Definição 3.7 - (Zadeh [25]): Um conjunto difuso é convexo se, e somente se, seus conjuntos de nivel α são todos convexos.

A noção de convexidade pode também ser apresentada, através de combinação linear.

Definição 3.8 - (Zadeh[25]) : Um conjunto difuso A é convexo se para qualquer $x_1 \in X$, para qualquer $x_2 \in X$, para qualquer $\lambda \in [0,1]$, tem-se:

$$\mu_A(\lambda x_1 + (1-\lambda) x_2) \geq \min (\mu_A(x_1) , \mu_A(x_2)) \qquad (3.20)$$

Como pode ser visto na figura (3.1), a definição (3.11), não implica no fato de que μ_A seja uma função convexa de X.

Note-se ainda que se A e B forem conjuntos difusos convexos, então A ∩ B também o será.

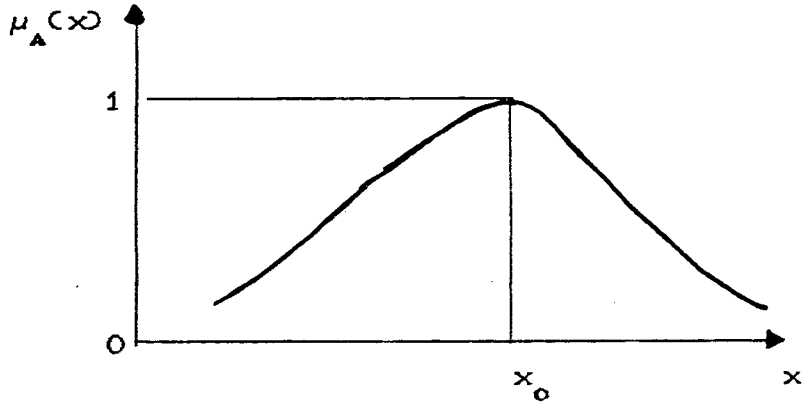


Figura 3.1 - Representação de um número difuso através de sua função de pertinência

Definição 3.9 - (Zadeh [25]): Um número difuso é um conjunto convexo normalizado A, definido sobre $\mathbb{R}$ se:

i) $\exists\, x_o \in \mathbb{R}$, chamado valor mais provável de A, tal que $\mu_A(x_o) = 1$;

ii) μ_A é continua por partes.

3.3 - Princípio da Extensão

Zadeh, [27], em 1975, formulou um dos princípios mais importantes para a teoria dos conjuntos difusos, conhecido como princípio da extensão. Este princípio é utilizado para extender os conceitos, rigor e formalismo da matemática clássica, à teoria dos conjuntos difusos. Através deste princípio, oprerações tais como soma, subtração, multiplicação, exponenciação, entre outras, podem ser extendidas para o campo dos números difusos.

Para se fazer uso da técnica de programação dinâmica clássica, os problemas devem ser precisamente definidos, no sentido de não admitir definições e/ou medições subjetivas. Isto significa que um problema definido de forma subjetiva não pode ser resolvido pela técnica de programação dinâmica clássica.

Nesse contexto, o princípio da extensão que pode ser caracterizado como a base teórica para o desenvolvimento do presente trabalho, isto é, um problema de programação dinâmica definido de forma subjetiva pode, via princípio da extensão, ser transformado num problema de programação dinâmica difusa e resolvido como tal, com a vantagem de poder incorporar aspectos qualitativos das informações.

Definição 3.10 - (Zadeh [27]): Seja X o produto cartesiano de universos, $X = X_1 x X_2 x \ldots x X_r$, e $A_1, \ldots A_r$, r

conjuntos difusos em $X_1,\ldots,X_r$, respectivamente. O produto cartesiano de $A_1,\ldots A_r$ é definido como:

$$A_1 x \ldots, A_r = \int_{X_1 x,\ldots,X_r} (\min \{\mu_{A_1}(x_1),\ldots,\mu_{A_r}(x_r)\}/(x_1,\ldots,x_r) \tag{3.21}$$

Definição 3.11 - (Zadeh [27]): Seja f um mapeamento de $x_1 x,\ldots,xx_r$ para o universo Y tal que $Y = f(x_1,\ldots,x_r)$. Então os r conjuntos difusos, denotados por A_i, induzem um conjunto difuso do mapeamento f, tal que:

$$\mu_B(y) = \begin{cases} \text{Sup } \min(\mu_{A_1}(x_1),\ldots,\mu_{A_r}(x_r)) \\ 0 \text{ se } f^{-1}(y) = 0 \end{cases} \tag{3.22}$$

onde f^{-1} é a imagem inversa de y, e $\mu_B(y)$ é o grau de pertinência do valor y no conjunto difuso B definido em Y, e $\mu_B(y)$ é o maior entre os graus de pertinência $\mu_{A_1}(x_1),\ldots,\mu_{A_r}(x_r)$.

A expressão (3.24) também pode ser escrita como:

$$B=f(A_1,\ldots,A_r) = \int_{x_1 x,\ldots,x_r} \min(\mu_{A_1}(x_1),\ldots,\mu_{A_r}(x_r))/\ f(x_1,\ldots,x_2) \qquad (3.23)$$

3.4 - Operações com Números Difusos

O princípio da extensão definido acima, tem aplicação geral, mas no presente trabalho é dado ênfase sómente às operações de adição, subtração, multiplicação, máximos e mínimos, e comparações de números difusos, os quais são utilizados no desenvolvimento das aplicações computacionais (programação dinâmica deterministica difusa e programação dinâmica estocástica difusa).

3.4.1 - Adição de Números Difusos

A adição é uma operação crescente, isto é, dados $x_1 > x_2$ e $y_1 > y_2$, então $x_1 + y_1 > x_2 + y_2$.

A extensão difusa também é crescente, sendo definida através do princípio da extensão como:

$$\mu_{A \oplus B}(z) = \sup_{\substack{z=x+y \\ x \in A \\ y \in B}} \min(\mu_A(x), \mu_B(y)), \quad \forall\ A,B \in R \qquad (3.24)$$

onde:

i) R denota o conjunto dos números difusos

ii) $\oplus$ denota a soma difusa extendida;

iii) $A \oplus B$ é o número difuso resultante da soma dos números difusos A e $B \in R$.

A operação de adição, conforme foi definida, apresenta as seguintes propriedades:

a) Comutatividade

$$A \oplus B = B \oplus A;$$

b) Associatividade

$$A \oplus (B \oplus C) = (A \oplus B) \oplus C$$

c) Identidade

$A \oplus 0 = A$ (zero não difuso)

O operador $\oplus$ não apresenta elemento simétrico, no sentido de estrutura de grupo, isto é:

$$\mu_M + (-\mu_M) \neq 0, \ \forall \ M \in R\text{-}\mathbb{R} \qquad (3.25)$$

Entretanto, o elemento simétrico de $M \in R$, é definido pelo principio da extensão como:

$$\mu_{-M}(x) = \mu_M(-x) \quad , x \in \mathbb{R} \qquad (3.26).$$

3.4.2 - Multiplicação de Números Difusos

A multiplicação é uma operação crescente em $\mathbb{R}^+$ e decrescente em $\mathbb{R}^-$. Assim, o produto de números difusos, denotado pelo operador $\odot$, que são todos positivos, é um número positivo.

A operação de multiplicação difusa,

definida por, Zadeh [25], através do princípio da extensão pode ser definida como:

$$\mu_{A \odot B} = \sup_{\substack{z=x.y \\ x \in A \\ y \in B}} \min (\mu_A(x), \mu_B(y)), \quad \forall \ A, B \in R \qquad (3.27)$$

onde A ⊙ B é o número difuso resultante da multiplicação dos números difusos A e B.

Esta operação, tal qual foi definida, satisfaz as seguintes propriedades:

a) Comutatividade

$A \odot B = B \odot A$

b) Associatividade

$A \odot (B \odot C) = (A \odot B) \odot C$

Note-se que o conjunto de números difusos positivos não representa estrutura de grupo para o operador extendido ⊙, haja visto que:

para qualquer M tem-se :

$M \odot 1 = M$;

$M \odot M^{-1} \neq 1$, pois M não é um número real.

Pode-se ver também que se M for um número difuso estritamente positivo ou negativo, e que se N e P forem também números difusos estritamente positivos ou negativos, então tem-se :

$$M \odot (N \oplus P) = (M \odot N) \oplus (M \odot P) \qquad (3.28)$$

3.4.3 - Máximos e Mínimos de Números Difusos

As operações de max e min são crescentes em $\mathbb{R}$. O máximo e o mínimo de n números difusos denotados por $\tilde{max}(M_1, M_2, \ldots, M_n)$ e por $\tilde{min}\ (M_1, M_2 \ldots, M_n)$, é um número difuso.

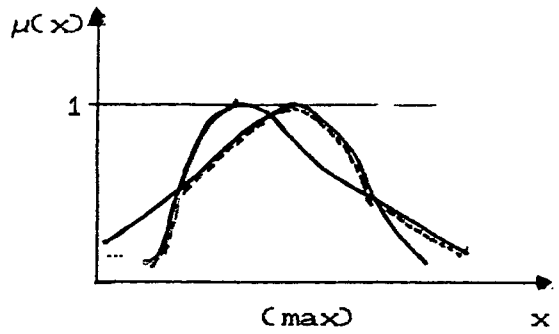
Aplicando o princípio da extensão aos operadores $\tilde{max}$ e $\tilde{min}$, tem-se:

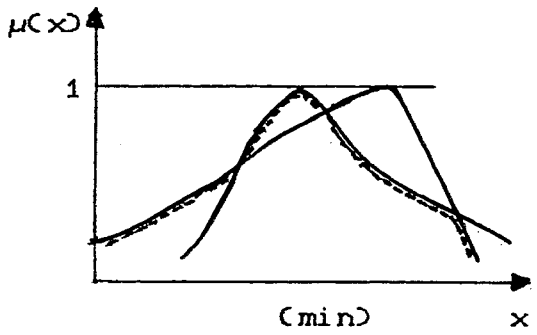
$$\mu_{\tilde{max}(M_1,\ldots,M_n)}(z) = \sup_{\substack{z = max(x_1,\ldots,x_n) \\ x_1 \in M_1 \\ \vdots \\ x_n \in M_n}} \min\left(\mu_{M_1}(x_1),\ldots,\mu_{M_n}(x_n)\right) \qquad (3.29)$$

e

$$\mu_{\tilde{min}(M_1,\ldots,M_n)}(z) = \sup_{\substack{z = min(x_1,\ldots,x_n) \\ x_1 \in M_1 \\ \vdots \\ x_n \in M_n}} \min\left(\mu_{M_1}(x_1),\ldots,\mu_{M_n}(x_n)\right) \qquad (3.30)$$

Nas figuras (3.2)e (3.3) abaixo, apresenta-se a interpretação grafica dos operadores $\tilde{max}$ e $\tilde{min}$.





Figuras 3.2 e 3.3 - Representação gráfica das operações de máximo e mínimo de conjuntos difusos.

As seguintes propriedades são satisfeitas para estes operadores:

a) Comutatividade

$$\tilde{min}\ (A,B) = \tilde{min}\ (B,A)$$

$$\tilde{max}\ (A,B) = \tilde{max}\ (B,A)$$

b) Associatividade

$$\tilde{min}\ (\ A, \tilde{min}\ (B,C)\ ) = \tilde{min}\ (\tilde{min}\ (A,B),C)$$

$$\tilde{max}\ (\ A, \tilde{max}\ (B,C)\ ) = \tilde{max}\ (\tilde{max}\ (A,B),C)$$

c) Distributividade

$$\tilde{min}\,(A, \tilde{max}\,(B,C)) = \tilde{max}\,(\tilde{min}\,(A,B), \tilde{min}\,(A,C));$$

$$\tilde{max}\,(A, \tilde{min}\,(B,C)) = \tilde{min}\,(\tilde{max}\,(A,B), \tilde{max}\,(A,C));$$

$$A + \tilde{max}\,(B,C) = \tilde{max}\,((A + B), (A + C));$$

$$A + \tilde{min}\,(B,C) = \tilde{min}\,((A + B), (A + C));$$

d) Lei da absorção

$$\tilde{max}\,(A, \tilde{min}\,(A,B)) = A;$$

$$\tilde{min}\,(A, \tilde{max}\,(A,B)) = A;$$

e) Lei de Morgan

$$1 - \tilde{min}\,(M,N) = \tilde{max}\,((1 - M),(1 - N));$$

$$1 - \tilde{max}\,(M,N) = \tilde{min}\,((1 - M),(1 - N));$$

f) Idempotência

$$\tilde{max}\,(M,M) = M;$$

$$\tilde{min}\,(M,M) = M.$$

Note-se que, a operação de máximo (mínimo), definida para os números ordinários determina o maior (menor) argumento da lista, entretanto, o mesmo não acontece para as operações com números difusos.

3.4.4 - Distribuições de Probabilidades Difusas

3.4.4 - Distribuições de Probabilidades Difusas

No presente trabalho utiliza-se os conceitos de probabilidade difusa à programação dinâmica difusa.

A probabilidade de ocorrer um subconjunto X, isto é, $X = \{x_1,,\ldots,x_k\}$, $k \leq n$, onde n é igual ao número de eventos possíveis é definida como uma soma difusa interativa σ_k, de uma probabilidade linguistica (não muito bem definida) $\tilde{p}_i$,(i = 1,k), tal que:

$$\mu_{\sigma k}(z) = \sup_{\substack{z=u_1+\ldots+u_K \\ u_1+\ldots+u_k \leq 1}} \min_{i=1,k} \mu_{\tilde{P}}(u_i) = \tilde{min}(1,\tilde{p}_1+\ldots+\tilde{p}_k) \qquad (3.31)$$

O valor esperado difuso de uma variável aleatória V, que assume valores no conjunto $\{ a_1,\ldots,a_n \} \subset \mathbb{R}$ com probabilidades linguisticas $\tilde{p}_i$(i=1,n), é definida como a soma interativa E(V), tal que:

$$\mu_{E(V)}(z) = \sup_{\substack{z=a_1u_1+\ldots+a_nu_n \\ u_1+\ldots+u_n = 1}} \min_{i=1,n} \mu_{\tilde{P}}(u_i) \qquad (3.32)$$

Deve-se notar que a equação acima sómente pode ser aplicada quándo se deseja calcular o valor esperado de

uma variável com dois eventos.

Quando se deseja calcular o valor esperado de uma função com mais de duas variáveis, deve-se fazer uso de um modelo de programação matemática equivalente :

$$\max \quad \alpha$$

$$\text{s.a}$$

$$\mu_{\tilde{P}_i}(u_i) \geq \alpha \quad , \; i=1,\ldots,u$$

$$\sum_{i=1,n} a_i u_i = Z \qquad (3.33)$$

$$\sum_{i=1,n} u_i = 1$$

onde α é o grau de pertinência de Z ser o valor de E(V), isto é,:

$\alpha = \mu_{E(V)}(Z)$

3.4.5 - Comparação de Números Difusos

Quando se deseja comparar números difusos, duas questões são levantadas:

1 - Qual é o valor difuso do maior ou menor número de uma conjunto de números difusos ?

2 - Qual é o maior ou o menor entre vários números difusos ?

A primeira pergunta pode ser resolvida através das operações $\tilde{max}$ e $\tilde{min}$. A segunda pergunta requer a seguinte metodologia:

Dados M,N $\in \mathbb{R}$, deseja-se determinar qual o grau de possibilidade de $M \geq N$.

Utilizando-se o princípio da extensão tem-se:

$$\mu (M \geq N) = \sup_{\substack{x \geq y \\ x \in M \\ y \in N}} (\mu_M(x) , \mu_N(y)) \qquad (3.34)$$

O fato de $\mu (M \geq N) = 1$, não implica que, necessariamente tenhamos $\mu (M < N)$ nulo.

$\mu (M < N)$ pode ser diferente de zero. Entretanto, quanto mais próximo desse valor ele for, maiores serão as possibilidades de que M seja maior que N.

Na figura (3.4) estes conceitos são apresentados graficamente.

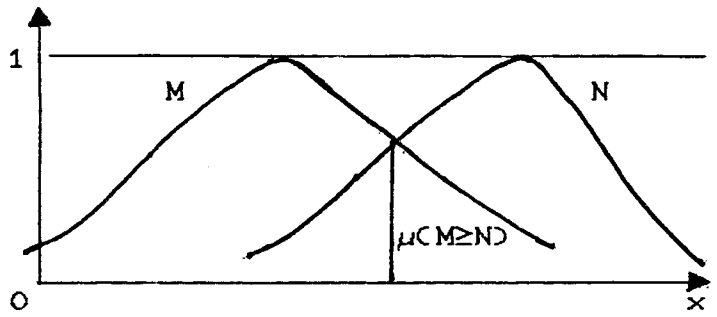


Figura 3.4 - Representação gráfica da operação de comparação de números difusos.

3.5 - Computação Numérica de Números Difusos

Dubois e Prade [10], em 1978, sugeriram a realização das operações difusas, através de funções características, à esquerda e à direita do valor esperado.

Definição 3.12 - (Dubois [10]) Uma função denotada por L (esquerda) e R (direita) é uma função característica de

números difusos se, e somente se, as seguintes propriedades forem satisfeitas:

1 . L (x) = L (- x) ;

2 . L (0) = 1;

3 . L é não decrescente no intervalo [0,∞].

Definição 3.13 - (Dubois [10]): Um número difuso M é um número difuso do tipo L - R se, e somente se:

$$\mu_M(x) = \begin{cases} L\,((m - x) / \alpha), & x \leq m\ ,\ \alpha > 0 \\ R\,((x - m) / \alpha), & x \geq m\ ,\ \beta > 0 \end{cases} \qquad (3.35)$$

onde

m = valor mais significativo de M;

α e β são definidos como medidas de dispersão. Deve-se notar que, quando α e β assumem o valor zero, M é, por convenção, um número preciso.

A computação númerica através da representação L-R, proporciona muita rápidez no cálculo das operações difusas. Entretanto, o resultado obtido nem sempre é exato, e sua aplicação é restrita a alguns tipos de funções

características.

Mayerle [20], em 1988, propos uma representação alternativa, que embora não apresente a mesma rápidez do método L-R, apresenta os resultados comforme a precisão estabelecida. Além disso, independem da definição de funções características para a representação de números difusos.
A notação alternativa proposta para a representação do número difuso é:

$$N = [\ n^{-}_{\alpha i}\ , n_{1}\ ,\ n^{+}_{\alpha_{k+1-i}}\ ]\ ,\ i\ =\ 1,..,k \qquad (3.36)$$

A imagem [0,1] da função é dividida em k-intervalos iguais. Na figura (3.5), os parâmetros dessa representação são identificados.

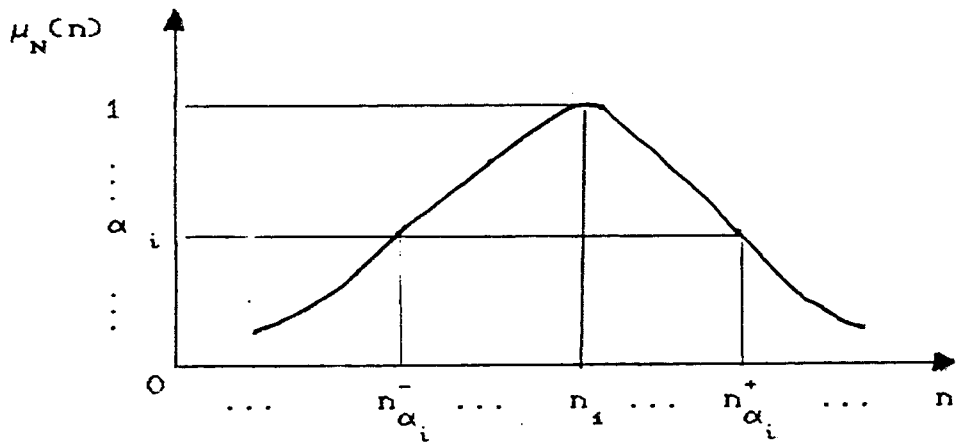


Figura 3.5 - Representação gráfica da notação alternativa proposta para números difusos

Assim, as operacões com números difusos, segundo esta notação, passa a ser:

a) Adição

$$M \oplus N = [\ m^{-}_{\alpha i} + n^{-}_{\alpha i}\ ,\ m_1 + n_1,\ m^{+}_{\alpha_{k+1-i}} + n^{+}_{\alpha_{k+1-i}}\ ] \qquad (3.37)$$

b) multiplicação de números difusos

$$M \odot N = [\ m^{-}_{\alpha i} \cdot n^{-}_{\alpha i}\ ,\ m_1 \cdot n_1,\ m^{+}_{\alpha_{k+1-i}} \cdot n^{+}_{k+1-i}\ ] \qquad (3.38)$$

c) Máximos de Números Difusos

$$\tilde{max}\,(M,\ N) = [\ \max(m^{-}_{\alpha i},\ n^{-}_{\alpha i}),\ \max(m_1, n_1),\ \max(m^{+}_{\alpha_{k+1-i}},\ n_{\alpha_{k+1-i}})\ ] \qquad (3.39)$$

d) Operador Mínimo Difuso

$$\tilde{min}(\min)\ [\min(m^{+}_{\alpha i},\ n^{+}_{\alpha i}),\ \min(m_1, n_1),\ \min(m^{+}_{\alpha_{k+1-i}},\ n^{+}_{\alpha_{k+1-i}}) \qquad (3.40)$$

D) Comparação de Números Difusos

$$\mu(M \geq N) = \begin{cases} 1 \text{ se } m_1 \text{ se } m_1 \geq n_1 \\ hgt(M \cap N) \text{ se } m_1 < n_1 \end{cases} \qquad (3.41)$$

Através desse procedimento, os números difusos podem ser armazenados em listas (vetores) de dimensão 2K + 1, onde a complexidade de operação é da ordem de 2K + 1 operações aritméticas elementares.

A precisão das operações depende do valor de k estipulado. Quanto maior o valor de k, menor será a eficiência computacional e maior será a precisão obtida.

O problema reside, então, em se determinar um valor ótimo de k, onde tanto a precisão, quanto a eficiência, sejam computacionalmente aceitáveis.

3.6 - Determinação de Conjuntos Difusos [10]

Apesar dos modelos de estimação de funções de pertinência não terem sidos sistematicamente estudados, alguns métodos foram desenvolvidos e propostos por vários autores, os quais são apresentados a seguir.

Definido um problema de forma subjetiva, pode-se, através dos métodos abaixo apresentados, estimar a função de pertinência, e assim poder avaliar analíticamente os parâmetros e/ou informações.

1 - Método da Exemplificação : Seja U um universo de objetos e A um conjunto difuso em U. A função de pertinência μ_A pode ser estimada através de informações parciais, tal que o valor que μ_A assume é obtido a partir de uma amostragem de U. Nesse caso, a determinação de conjuntos difusos através do método da exemplificação é uma extensão da noção lingüistica da definição extendida. O problema de estimar a função de pertinência de um conjunto difuso através do conhecimento dos valores de um número finito, isto é, uma amostragem de U, é um problema de abstração.

Um exemplo seria o conjunto de homens altos, onde não se tem um limite de altura bem definido para esse grupo. Esse problema lingüistico pode ser representado por níveis númericos : 1 ; 0.75 ; 0.5 ; 0.25 ; 0, respectivamente. A representação discreta da função de pertinência é obtida pela repetição da pergunta (o quanto alto, o quanto baixo) para diversas estaturas.

2- Método da Definição Analítica Implícita: Esse método é atribuido a Kochen e Badre. A função de pertinência é assumida contínua e diferenciável. Como exemplo considere-se o conjunto A denotado pelo adjetivo largo. O crescimento marginal da intensidade das pessoas que acreditam que x é igual a A é assumido proporcional à intensidade das pessoas que acreditam que x não é igual a A, isto é:

$$\frac{d\mu_A(x)}{dx} = k\mu_A(x)\,(1 - \mu_A(x)) \qquad (3.42)$$

cuja solução é:

$$\mu_A(x) = 1 / (1 + e^{a - bx}) \qquad (3.43)$$

3 - Uso de Estatística: Dado um conjunto de dados estatísticos na forma de um histograma, a indução de uma distribuição de possibilidade é diferente da distribuição de probabilidades. No primeiro caso, o histograma é normalizado, no caso da distribuição de probabilidades a superfície do histograma é assumida igual a 1,0.

O princípio da consistência afirma que a possibilidade de um evento é sempre maior ou igual que sua probabilidade. Seja h uma função definida de $\mathbb{R}$ para $\mathbb{R}^+$, representando um histograma. A associação das distribuições de probabilidades e possibilidades pode ser satisfeita para qualquer união D de intervalos disjuntos, através de:

$$\Pi(D) = \sup_{x \in D} h(x) / \sup h \geq prob(D) = \int_{D} h(x)dx \; / \; \int_{-\infty}^{+\infty} h(x)dx \qquad (3.44)$$

4- Método das Preferências Relativas: Esse método foidesenvolvido por Saaty. Seja A um conjunto difuso sobre um universo U. O valor da função de pertinência $\mu_A(x_i) = w_i$, $x_i \in U$ são calculados de um conjunto de dados representativos do valor relativo de pertinência t_{ij} de um elemento x_i de A em relação a um elemento x_j de A. Utiliza-se então uma escala dividida em 7 níveis { 1/9 , 1/8 ,..., 1/2 , 1 , 2 ,..., 8 , 9}. Cada nível tem uma interpretação semântica: O maior t_{ij} é a maior função de pertinência x_i comparado em relação a x_j. A matriz t_{ij} é tal que $t_{ij} = 1 / t_{ji}$. T é dito consistente se e somente se $\exists$ $w = (w_1,\ldots,w_n)$ com $n = |U|$, onde w é um conjunto de autovalores de T, tal que $t_{ij} = w_i/w_j$. Quando T é consistente, T é transitivo, tal que:

$$t_{ij} \times t_{jk} = t_{ik} \qquad (3.45)$$

5 - Método da Função Filtro: Esse método foi desenvolvido por MacVicar e Whelan em 1978, para identificar as funções de pertinência de conjuntos difusos modelados em forma de adjetivos como alto e largo.

Uma função filtro F é caractérizado por dois parâmetros: a posição NP do ponto neutro (valor mais provável), tal que F(NP)= 1/2) e a extensão 2w de transição entre a não-pertinência e a pertinência. O objetivo é determinar a forma

gráfica do conjunto difuso em $\mathbb{R}$. Mais especificamente:

$$F(x;NP,w) = \begin{cases} 0 & \text{se } x \in (-\infty, NP - w] \\ (1/2)(x-NP+w) & \text{se } x \in [NP - w, NP + w], \\ 1 & \text{se } x \in [NP + w, +\infty] \end{cases} \tag{3.46}$$

Como exemplo seja o conjunto A de pessoas com estatura alta, onde se assume que a distribuição de probabilidade é conhecida. Seja $\bar{x}$ e σ parâmetros da distribuição. Afirmar que uma pessoa é alta é assumir que ela tem uma grande estatura, onde grande é modelado pela função pertinência μ tal que:

$$\mu(x) = F(x;\bar{x} + \alpha\sigma, \beta\sigma) \tag{3.47}$$

onde x é a altura e α e ß são determinados experimentalmentes.

No presente trabalho utiliza-se funções filtros, denotadas através de um spread (medida de dispersão) para estimar a determinação dos conjuntos difusos. A forma dessa função filtro é triangular, sendo modelada como:

$$F = VMP \mp w \tag{3.48}$$

onde:

VMP = valor mais provável;

w = medida de dispersão

Na figura 3.6, abaixo apresenta-se graficamente a função triangular.

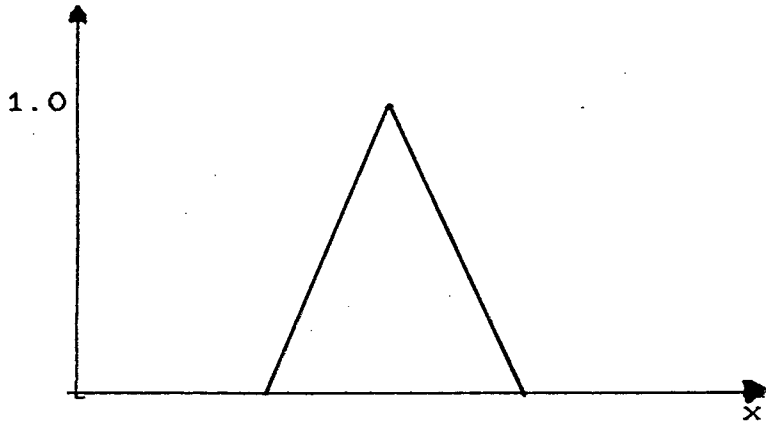


Figura 3.6 - Representação da função triangular

CAPÍTULO IV

4. - Programação Dinâmica Difusa

4.1 - Introdução

Nos Capítulos anteriores apresentou-se os conceitos de programação dinâmica clássica, e as propriedades e conceitos relativos à teoria dos conjuntos difusos.

Nesse capítulo, procura-se desenvolver uma técnica que concatena esses dois conceitos, para resolver problemas de programação dinâmica em cuja formulação estão presentes aspectos difusos. A técnica de programação dinâmica difusa, ora desenvolvida, pode ser vista como uma generalização modelo clássico.

4.2 - Extensão Difusa das Condições de Validação

No capítulo II, mostrou-se que para o algoritmo recursivo ser aplicado, as condições de validação do modelo,isto é, a condição de separabilidade e a condição de otimalidade, devem ser satisfeitas.

A condição de separabilidade foi desenvolvida como sendo a base da aplicação da recursividade, dado um plano fixo, sendo denotada através da equação (2.19):

$F(n,i_n,A_n) = \Phi_n(r(n,i_n,k_n)\ ,\ F(n-1,i_{n-1},A_{n-1}))$, onde

$F(n-1,i_{n-1},A_{n-1}) = \phi_{n-1}(r(n-1,i_{n-1},k_{n-1}),\ldots,r(1,i_1,k_1))$

sendo ϕ e Φ funções apropriadas.

Para se aplicar a técnica de programação dinâmica ora proposta, essas condições de validação devem também ser satisfeitas. Considerando que os retornos de estado são difusos, então a extensão difusa do princípio da separabilidade é feita via princípio da extensão, nos termos dos conceitos e definições apresentadas na secção (3.3) do capítulo III, isto é:

$\tilde{F}(n,i_n,A_n) = \Phi_n(\tilde{r}(n,i_n,k_n),\tilde{F}(n-1,i_{n-1},A_{n-1}))$, onde (4.1)

onde

$\tilde{F}(n-1,i_{n-1},A_{n-1}) = \Phi_{n-1}(\tilde{r}(n-1,i_{n-1},k_{n-1}),\ldots,\tilde{r}\ (1,i_1,k_1))$

(4.2)

A condição de separabilidade sendo satisfeita, tem-se que para todo plano, o valor de cada estado será calculado como uma função do retorno do estágio imediato e do valor do estado subsequente.

Por outro lado, a condição de otimalidade requer que o valor de cada estado seja calculado recursivamente para um dado plano. A formulação algébrica dessa condição é dada,

para o caso clássico, através da equação (2.24):

$$F(n,i_n,k_n+A_{n-1}) \geq F(n,i_n,k_n+A_{n-1})$$

A inequação acima é a sentença que garante a condição de otimalidade, assumindo-se o caso de maximização.

A extensão difusa do princípio da otimalidade é feita via princípio da extensão, nos termos dos conceitos e definições apresentados na secção (3.3) do capítulo III, sendo formalizada como:

$$\tilde{F}(m,i_n,k_n+A^{o}_{n-1}) \geq \tilde{F}(n,i_n,k_n+A_{n-1}) \qquad (4.3)$$

4.3 - Extensão Difusa da Programação Dinâmica Determinística

A programação dinâmica difusa determinística procura solucionar problemas em cuja estrutura há uma certa quantidade de informações e/ou medições subjetivas. Para solucionar computacionalmente o problema, tem-se que, a priori, dá um tratamento matemático a tais informações.

Mostrou-se no capítulo II que o problema clássico de programação dinâmica tem a seguinte terminologia:

1 - Estágio (n);

2 - Estado (n,i);

3 - Ação (k);

4 - Retorno de Estado r(n,i,k);

5 - Valores de Estado F(n,i).

A relação de recorrência do algoritmo recursivo é, para um problema de maximização, dado pela equação (2.1):

$$F(n,i) = \max_{k \in K} [r(n,i,k) + F(n-1,j)]$$

A equação acima significa que todas as ações serão consideradas e selecionar-se-á aquela que forneça o máximo valor de estado.

A equação de transição é função dos estágios, estados e ações envolvidos no problema, sendo dado pela equação (2.2):

$$j = t(n,i,k)$$

A equação de transição está relacionada à transição de um estado a um outro estado adjacente.

Segundo as definições e conceitos apresentados no capítulo III, a equação (2.1) pode ser extendida para o caso difuso, e a relação de recorrência para um problema de programação dinâmica determinística difusa pode ser escrita como:

$$\tilde{F}(n,i) = \tilde{\max}_{k \in k} \{\tilde{r}(n,i,k) \oplus \tilde{F}(n-1,j)\} \qquad (4.4)$$

Na equação acima, os estados, estágios e ações envolvidos no problema são conhecidos com certeza, sendo, portanto, números ordinários (precisos). Considera-se também que a transição entre estados é conhecida com certeza, haja visto que esta é função dos estados, estágios e ações. O valor de estado obtido segundo esta equação será um número difuso.

Em um problema de programação dinâmica, o retorno é uma caracterização do lucro esperado, ou dos custos atribuidos, ou ainda dos recursos comsumidos. Entretanto, pode ocorrer o caso onde os valores que caracterizam os retornos não são conhecidos com precisão. São valores que foram de certa forma estimados ou medidos com determinado grau de imprecisão ou subjetividade. O termo $\tilde{r}(n,i,k)$ da equação (4.4) denota um retorno difuso, que agora pode ser avaliado através da técnica propostana seção 3.6.

A nível computacional os cálculos com os operadores difusos utilizados na programação dinâmica determinística difusa , isto é, soma extendida, máximos e mínimos e a comparação de números difusos são executados da mesma forma que as operações da matemática clássica, sendo que no caso da teoria dos conjuntos difusos estas operações são executadas considerando-se um conjunto de nível α, que denota o grau de pertinência, comforme proposto na seção 3.5.

4.4 - Extensão Difusa da Programação Dinâmica Estocástica

Um problema de programação dinâmica estocástica difusa se caracteriza pelo fato de que os retornos entre estados são difusos, os valores de estados terminais são difusos e existe uma probabilidade difusa associada à transição entre estados. Tal problema surge quando não se conhece as distribuições de probabilidades associadas ao sistema, e procura-se estimar determinados valores de probabilidade de forma a melhor caracterizar as incertezas.

Os critérios adotados para estimar a distribuição de probabilidade, contém uma quantidade relevante de informações subjetivas. Pode-se obter informações mais consistentes para esses casos através da teoria dos conjuntos difusos e, se o problema for caracterizado como sendo um problema de decisão de markov, pode-se solucioná-lo atrvés da programação dinâmica difusa estocástica.

Um caso clássico de programação dinâmica são os modelos de estoques. Em geral, nesses problemas ocorre uma demanda aleatória não conhecida, estimada por distribuições. A determinação da política ótima, poderia englobar os custos estimados de armazenamento, reposição e transportes e os custos de preparação de pedidos, entre outros.

Na programação dinâmica estocástica difusa, para uma ação particular k, a probabilidade de transição do estado (n,i) para o estado (n-1,j) é denotada por $\tilde{P}(n,i,j,k)$.

Associado com a transição do estado (n,i) para o estado (n-1,j), existe também, um retorno de transição, para uma dada ação k, denotado por $\tilde{C}(n,i,j,k)$

O retorno de estágio difuso, será função do somatório do produto das probabilidades e dos retornos de transição, sendo dado pela seguinte equação:

$$\tilde{r}(n,i,k) = \sum_{i=1}^{N} \tilde{P}(n,i,j,k) \odot \tilde{C}(n,i,j,k) \qquad (4.5)$$

A existência de probabilidade de transição entre os estados caracteriza um problema de decisão markoviana.

O conjunto de ações para cada estado, constitue uma política que define um processo markoviano difuso. Nesse processo, o valor de estado ótimo, pode ser calculado, através da versão modificada (difusa) do algoritmo valor da iteração, ou seja:

$$\tilde{F}(n,i) = \max_{k \in k_{ni}} [\tilde{r}(n,i,k) \oplus \sum_{j=1}^{n} \tilde{P}(n,i,j,k) \odot [\tilde{F}(n-1,j)] \qquad (4.6)$$

O somatório apresentado na equação acima, consiste no cálculo do valor esperado da variável valor de estado. Este cálculo, será apresentado em mais detalhes na seção 4.4.1.

Note-se que, assim como no caso difuso determinístico, na programação dinâmica markoviana difusa, não se

obterá uma solução ótima mas um conjunto de soluções que permitem ao tomador de decisões, avaliar, para diversas graus de pertinência , os valores ótimos de estado obtido.

Assim as mesmas considerações no que se refere aos valores de estado no caso difuso determinístico, são válidos para o caso estocástico.

4.4.1 - Valores Esperados Difusos

Na seção anterior, quando da apresentação da programação dinâmica estocástica ficou caracterizado a necessidade de se calcular o valor esperado do valor de estado, considerando-se uma distribuição de probabilidade difusa.

Definindo-se um conjunto $X = \{x_1, \ldots, x_n\}$ e uma probabilidade difusa $\tilde{p}_i$ associada a cada x_i, pode-se calcular os valores esperados difusos das variáveis definidas para o problema da seguinte forma:

$$\mu_{E(X)}(z) = \sup_{z = x_1 u_1 + \ldots + x_n u_n \; ; \; 1 = u_1 + \ldots + u_n} \; \min_{i=1,n} \mu_{\tilde{p}}(u_i) \qquad (4.7)$$

Quando se tem mais de duas váriaveis, o cálculo do grau de pertinência $\mu_{E(X)}(Z)$ pode ser feito através de programação matemática, resolvendo o seguinte problema de programação não linear:

$$\max \quad \theta$$

s.a

$$\mu_{\tilde{P}_i}(u_i) \geq \theta \quad , \qquad i = 1,n$$

$$\sum_{i=1}^{n} u_i = 1 \qquad (4.8)$$

$$\sum_{i=1}^{n} x_i u_i = Z$$

Outra forma equivalente para abordar o mesmo problema é, resolver os problemas de programação linear abaixo para valores distintos de grau de pertinência α :

$$\max \quad Z = \sum_{i=1}^{n} x_i u_i$$

s. a (4.9)

$$\sum_{i=1}^{n} u_i = 1$$

$$u_i^{min} \leq u_i \leq u_i^{max}$$

e

$$\min Z = \sum_{i=1}^{n} x_i u_i$$

s.a (4.10)

$$\sum_{i=1}^{n} u_i$$

$$u_i^{min} \leq u_i \leq u_i^{max}$$

Para resolver os problemas de programação linear acima, não é necessário utilizar um algoritmo tipo Simplex. Esses problemas podem ser resolvidos através de um processo de ordenação dos valores x_i, atribuindo-se valores para as probabilidades de acordo com esta ordenação.

CAPÍTULO V

5. Aplicação da Técnica Proposta

5.1 - Introdução

Para melhor caracterização e compreensão dos conceitos até aqui abordados, serão apresentados dois problemas: um focalizando o caso determinístico, e outro o caso estocástico. Esses problemas foram adaptados a partir de um exemplo proposto por Hastings[15].

5.2 - Um Exemplo de Programação Dinâmica Determinística Difusa

5.2.1 - Descrição do Problema

Seja o seguinte problema de expansão de linhas de produção, de uma empresa que possui apenas uma linha instalada, conforme apresentado a seguir.

As pesquisas de mercado apontam que pode haver viabilidade econômica na expansão da capacidade de produção.

O planejamento inicial de expansão prevê a análise ao longo de três anos. As estimativas de investimento e retorno nos periodos considerados envolvem fatores subjetivos que impedem o cálculo preciso dos parâmetros investimento e retorno.

Tais fatores se referem ao comportamento dos preços no mercado, à demanda pelos produtos da empresa e ao

nível de competitividade dos produtos concorrentes.

Alguns alternativas foram analisadas e, considerando-se os fatores acima apresentados foram obtidos as avaliações constantes nas tabelas (5.1) e (5.2).

Os dados da tabela (5.1) , associam a uma determinada avaliação um valor médio (ou valor mais provável), denotado genericamente por r(n,i,k), e um spread, denotado genericamente por sp(n,i,k), que representa uma medida de dispersão em torno do valor mais provável. Assim, por exemplo, para estado (3,1), isto é, faltando 3 anos para o término do periodo de planejamento e existindo 1 linha de produção implantada, as avaliações indicam que existem duas ações alternativas. A primeira seria manter o nível de capacidade (k=1) e ter um retorno líquido de NCz$ 2 milhões, que representa o valor mais provável, associado a uma dispersão à esquerda ou à direita do valor mais provável de NCz$ 0.5 milhões. A segunda ação alternativa, seria expandir o nível de capacidade (k = 2), o que geraria um déficit de NCz$ 7 milhões (valor mais provável), associado a uma dispersão de NCz$ 1.5 milhões. Esse déficit decorre do fato da empresa arcar com os custos de expansão da capacidade de produção.

Para os demais estados, as interpretações são analógas à feita acima.

Na tabela 5.2, encontram-se os valores dos estados terminais, que representam os retornos esperados ao final do horizonte de planejamento.

ESTADO (n,i)	ACAO k	RETORNO r(n,i,k)	SPREAD sp(n,i,k)
1,1	1	4	0.8
	2	-5	0.6
1,2	1	9	1
	2	4	0.2
1,3	1	12	3
	-	-	-
2,1	1	3	0.4
	2	-6	1.0
2,2	1	7	2.0
	2	2	0.3
3,1	1	2	0.5
	2	-7	1.5

Tabela 5.1 - Dados do Problema de Expansão de Linhas de Producão

Estado (n,i)	(0,1)	(0,2)	(0,3)
Valor Terminal	0	4	8
Spread	0	1	2

Tabela 5.2 - Valores dos Estados Terminais

A empresa deseja determinar o programa de investimento de capital que maximize os retornos, na expansão da capacidade de produção.

5.2.2 - Modelagem do Problema

Considere-se, inicialmente, que os parâmetros apresentados no problema descrito em 5.2.1 são definidos precisamente. Da terminologia adotada no capítulo II, para caracterizar o problema de programação dinâmica, tem-se:

Parâmetro	Descrição
Estágio [n]	número de anos remanescentes até o horizonte de planejamento;
Estado [i]	número de linhas de produção instaladas no estágio n;
Ação [k]	1 - manter o nível de capacidade atual; 2 - expandir a um nível mais alto
Retorno r(n,i,k)	resultado financeiro em um determinado estágio, sob determinada ação e estado;
Valor de Estado F(n,i)	retorno total obtido a partir de um estado quando um plano é executado.

Função de Transição $j = i + k - 1$

Função de recorrência $F(n,i) = \max_{k \in k_{ni}} (r(n,i,k) + F(n-1,j))$

Conjunto de ações viáveis $k_{ni} = \begin{cases} 1,2 \text{ se } i = 1 \text{ ou } 2 \\ 1 \quad \text{se } i = 3 \end{cases}$

Na formulação até aqui apresentada, supos-se que o problema é preciso. Em princípio, sem perda de generalidade, as definições apresentadas para o problema podem ser extendidas para o caso difuso. Apenas as informações referentes aos retornos devem sofrer algumas considerações.

Os valores apresentados nas tabelas (5.1) e (5.2) associam a uma determinada avaliação um valor medio mais provável e um spread (medida de dispersão). Para modelar tais informações, serão utilizados funções filtros, conforme apresentado na secção (3.8) do capítulo III.

Assim, para o problema corrente, a função de recorrênia será dada pela equação (4.4):

$$\tilde{F}(n,i) = \tilde{\max}_{k \in k_{ni}} [\tilde{r}(n,i,k) \oplus \tilde{F}(n-1,j)]$$

5.2.3 - Solução do Problema

Para resolver o problema de expansão de expansão de linhas de produção, utilizou-se um programa computacional, em Turbo-Pascal, versão 3.0, cuja listagem

encontra-se no apêndice A. Os resultados obtidos estão sumarizados na tabela 5.3.

ESTADO		ACAO	0	0.5	1.0	0.5	0.0	μ_i
0	1	-	0	0	0	0	0	-
0	2	-	3	3.5	4.0	4.5	5.0	-
0	3	-	6	7	8	9	10	-
		1	3.2	3.6	4.0	4.4	4.8	0.0
1	1	2	-2.6	-1.8	-1.0	-0.2	0.6	1.0
		MAX	3.2	3.6	4.0	4.4	4.8	
		1	11	12	13	14	15	1.0
1	2	2	9.8	10.9	12	13.1	14.2	0.76
		MAX	11	12	13	14	15	
		1	15	17.5	20	22.5	25	1.0
1	3	2	-	-	-	-	-	-
		MAX	15	17.5	20	22.5	25	
		1	5.8	6.4	7.0	7.6	8.2	1.0
2	1	2	4.0	5.5	7.0	8.5	10.0	1.0
		MAX	5.8	6.4	7.0	8.5	10.0	
		1	16	18	20	22	24	0.79
2	2	2	16.7	19.35	22	24.65	27.3	1.0
		MAX	16.7	19.35	22	24.65	27.3	
		1	7.3	8.15	9.0	9.85	10.7	0.51
3	1	2	8.2	11.6	15	18.4	21.8	1.0
		MAX	8.2	11.6	15	18.4	21.8	

Tabela 5.3 - Valores de estado esperados para o problema de expansão de linhas de produção

5.2.4 - Análise da Solução

Na tabela (5.3), apresentam-se os resultados para o problema de expansão de linhas de produção. Essa tabela contém os valores de estado obtidos para determinada ação, considerando os níveis α a esquerda e à direita do valor mais provável. Para cada ação calculou-se o grau de pertinência (μ) do valor de estado obtido ser maior que o máximo valor obtido entre todas as ações consideradas, para um determinado estado. Esse máximo valor de estado, na tabela (5.3), é denotado por MAX.

Assim, por exemplo, para o estado (1,2) caso a ação 1 seja selecionada, o grau de pertinência de que ela seja a melhor é 1.0, enquanto que o grau de pertinência de que a ação 2 seja a melhor é 0.76, conforme se pode verificar na última coluna da tabela (5.3).

Pelos dados da tabela (5.3), pode-se notar, ainda, que para um determinado nível α considerado, tem-se um conjunto de soluções. Exemplificando, considerando-se $\alpha = 0.75$, os seguintes planos relacionados à expansão ou mantutenção das linhas de produção, conforme figura (5.1), poderiam ser obtidos:

1 : Expandir - Expandir - Manter

2 : Expandir - Manter - Manter

3 : Expandir - Manter - Expandir

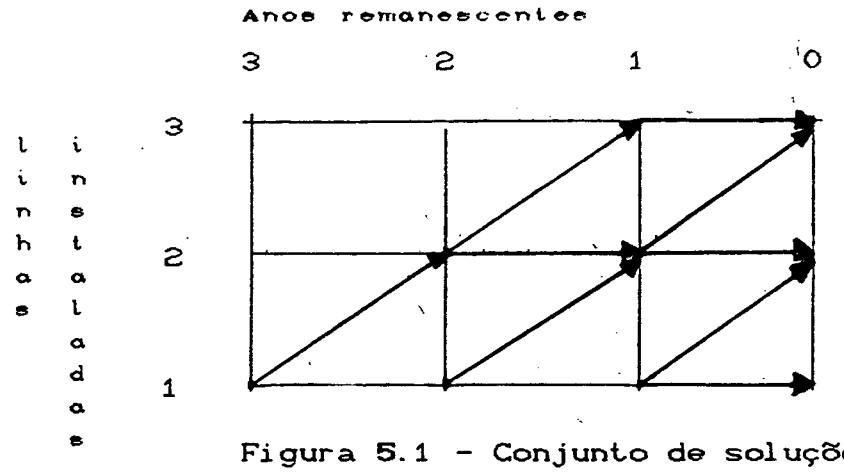


Figura 5.1 - Conjunto de soluções alternativas

5.3 - Um Exemplo de Programação Dinâmica Estocástica Difusa

5.3.1 - Descrição do Problema

Uma empresa tem um produto no mercado, cuja avaliação de desempenho, para o início de cada ano, é que este tenha sucesso ou insucesso, no mercado consumidor.

A empresa considera, para os seus propósitos de avaliação de desempenho, um horizonte de planejamento de dois anos. A fim de se ampliar a participação do produto no mercado, dispõe-se de duas alternativas:

1 - Fazer publicidade;

2 - Não fazer publicidade.

A tabela (5.4) abaixo, mostra um levantamento de dados obtidos através de uma pesquisa junto aos consumidores, onde os retornos esperados bem como as probabilidades associadas ao sistema, foram avaliados subjetivamente.

O efeito de não se fazer publicidade é uma diminuição na probabilidade de que o produto tenha sucesso, mas ao mesmo tempo tem-se uma redução nos custos. Por outro lado, ao se fazer publicidade obtem-se um aumento na probabilidade de que o produto tenha sucesso junto aos consumidores.

Assim, por exemplo, para o estado (1,1), isto é, faltando 1 ano para o término do periodo de planejamento e não estando a empresa fazendo publicidade, caso a ação (1) seja selecionada, haverá uma probabilidade igual a 0.9 de que o produto tenha sucesso no mercado, gerando um retorno de transição de NCz$ 2 milhões. Associado à probabilidade existe um spread, denotado na tabela (5.4) por SP, de 0.1, e ao retorno de transição existe um spread , denotado na mesma tabela por SC, de NCz$ 0.3 milhões. Para essa mesma ação, haverá uma probabilidade igual a 0.1, associada a uma dispersão de 0.03, de que o produto tenha insucesso no mercado, gerando um déficit de NCz$ 1 milhão, associado a uma dispersão de NCz$ 0.2 milhões em torno desse valor mais provável. Para a ação 2 do estado (1,1), bem como para os demais estados, a interpretação é analóga.

Na tabela (5.5), encontram-se os valores dos estados terminais e os spreads associados aos valores mais prováveis.

O objetivo da empresa é determinar

políticas promocionais com estrutura de decisão alternativa para cada ano.

ESTADO (n,i)	ACAO k	PROBABILIDADE DE TRANSICAO				RETORNOS DE TRANSICAO			
		P(n,i,1,k)	SP	P(n,i,2,k)	SP	C(n,i,1,k)	SC	C(n,i,2,k)	SC
1,1	1	0.9	0.1	0.1	0.03	2	0.3	-1	0.1
	2	0.7	0.2	0.3	0.15	4	0.8	1	0.2
1,2	1	0.6	0.1	0.4	0.1	1	0.1	-3	0.4
	2	0.2	0.08	0.8	0.2	2	0.3	-1	0.1
2,1	1	0.8	0.2	0.2	0.06	1	0.2	-2	0.2
	2	0.6	0.1	0.4	0.2	3	0.3	1	0.1

Tabela 5.4 - Dados do Problema de Marketing

Estado	(0,1)	(0,2)
Valor Terminal	1	0
Spread	0	0

Tabela 5.5 - Valores dos Estados Terminais

5.3.2 - Modelagem do Problema

O problema será modelado de acordo com a terminoliga apresentada no capítulo II.

Parâmetro	Descrição
Estágio [n]	número de anos remanescentes até o horizonte de planejamento
Estado [i]	resultado obtido,isto é, sucesso(i=1) ou insucesso (i=2), quando restam n anos
Ação [k]	1 - fazer publicidade; 2 - não fazer publicidade
Retorno de Transição C(n,i,j,k)	retorno associado com a transição do estado (n,i) ao estado (n-1,j), sob uma ação k.
Probabilidade de Transição P(n,i,j,k)	probabilidade de transição do estado (n,i) ao estado (n-1,j), sob uma ação k.
Retorno r(n,i,k)	resultado esperado associado com o estado (n,i) e uma ação k

Valor de Estado	resultado total esperado obtido a partir de um estado corrente, quando um plano é executado.

Os valores apresentados na tabela (5.5) associam a uma determinada avaliação um valor médio (mais provável) e um spread (medida de dispersão). Na modelagem das informações do problema será utilizado uma função filtro (triangular), de acordo com o teoria da secção (3.3) do capítulo III. Assim, a relação de recorrência para o problema corrente será dada pela equação(4.8):

$$\tilde{F}(n,i) = \max_{k \in k}[\tilde{r}(n,i,k) \oplus \sum_{j=1}^{z} [\tilde{p}(n,i,j,k)] \circ \tilde{F}(n-1,j)]$$

5.5.3 - Solução do Problema

Para resolver esse problema de programação dinâmica difusa estocástica, utilizou-se um programa computacional , em Turbo-Pascal, versão 3.0. Os resultados estão sumarizados na tabela (5.6).

ESTADO		ACAO	0	0.5	1.0	0.5	0.0	μ
0	1	-	1	1	1	1	1	-
0	2	-	0	0	0	0	0	-
1	1	1	1.81	2.19	2.60	3.04	3.52	0.53
		2	2.22	2.96	3.80	4.73	5.76	1.0
		MAX	2.22	2.96	3.80	4.73	5.76	
1	2	1	-0.07	-0.05	0.0	0.07	0.17	1.0
		2	-0.37	-0.30	-0.20	-0.08	0.07	0.0
		MAX	-0.07	-0.05	0.0	0.07	0.17	
2	1	1	1.40	2.32	3.44	4.85	6.59	0.79
		2	2.58	3.44	4.48	5.72	7.17	1.0
		MAX	2.58	3.44	4.48	5.72	7.17	

Tabela 5.6 - Valores de Estado Esperados para o Problema de Marketing

5.3.4 - Análise da Solução

Pelos resultados obtidos através da programação dinâmica estocástica difusa, sumarizados na tabela (5.6), o tomador de decisões tem agora, para as duas ações consideradas um conjunto de soluções alternativas. A exemplo do que foi apresentado na seção 5.2.4, também aqui pode-se estabelecer um conjunto de planos alternativos para um nível α estabelecido.

Na tabela (5.6), acima é feita uma comparação entre as ações envolvidas no problema. Isso é feito para se determinar o grau de pertinência (μ) de uma ação ser a melhor. O máximo valor de estado, é denotado na referida tabela por MAX.

Isso significa, que por exemplo, para o estado (1,2) , o grau de pertinência de que a ação 1 seja a melhor é 1.0, enquanto que o grau de pertinência de que a ação 2 seja a melhor é 0.0.

Novamente observa-se que inicialmente tinha-se um problema de programação dinâmica estocástica definido de forma subjetiva, que não poderia ser resolvido pelo modelo clássico. A técnica ora proposta permite que se resolva o problema modelado subjetivamente, servindo portanto como eficiente instrumento de análise para problemas de programação dinâmica definidos de forma subjetiva.

CAPÍTULO VI

6. Conclusões e Recomendações

6.1 - Conclusões

Em quase todos os processos gerênciais, tem-se o elemento tomador de decisões, para o qual interessa, por exemplo, a maximização dos lucros oriundos das vendas de um determinado bem ou da prestação de determinado serviço, a minimização dos custos associados à fabricação ou a combinação ótima de insumos na confecção de um ou vários produtos.

A programação dinâmica clássica, para estes casos, oferece ao tomador de decisão, apenas uma única solução para o problema, limitando sensivelmente o poder de tomada de decisão. No caso da programação dinâmica difusa, o tomador de decisão é inserido no processo, como um elemento importante, dado que esta técnica, permite fornecer, com base em um nível α especificado, não apenas uma única solução, mas um conjunto de soluções, permitindo a consideração da qualidade técnica dos planos apresentados. Isso permite ao tomador de decisão, um maior grau de liberdade, quanto a seleção da(s) melhor(res) alternativa(s), dentre o conjunto de soluções apresentadas.

Além disso, a técnica de programação dinâmica, ora proposta, surge como uma contribuição para se avaliar problemas definidos de forma lingüistica, isto é, problemas nos quais os parâmetros não são todos bem determinados e

problemas nos quais os parâmetros não são todos bem determinados e precisos, servindo como uma valiosa ferramenta no auxílio à tomada de decisão, no sentido de permitir uma análise de sensibilidade das soluções apresentadas.

6.2 - Recomendações

A dissertação ora desenvolvida, não esgota as possibilidades de futuros trabalhos que visem a utilização de programação dinâmica difusa. Na programação dinâmica, assim como em outras técnicas de pesquisaoperacional, é possível desenvolver outros trabalhos com a utilização da teoria dos conjuntos difusos.

Para novos estudos relacionados ao tema, sugere-se:

- Extensão da técnica apresentada para problemas de programação dinâmica com estágios infinitos;

- Utilização de outras funções de pertinência na determinação dos conjuntos difusos

- Elaboração de um sistema especialista que, a partir de uma descrição verbal de um problema formule e resolva problemas de programação dinâmica, considerando as técnicas propostas neste trabalho.

REFERÊNCIAS BIBLIOGRÁFICAS

[01] Anderson,D.R. et alii,An introduction to management science - quantitative approaches to decision making.4° ed. Minnesota,West Publishing Company,1985.

[02] Beckman,M.J. Dynamic programming of economic decisions. New York, Springer Verlag, 1968.

[03] Bellman,R.E. Dynamic programming. New Jersey, Princeton University Press,1957.

[04] Bellman,R.E. Giertz, M.; On the analytic formalism of the theory of fuzzy sets; Inf. Science, 5,:149-157, 1973.

[05] Blake,I.F. An introduction to applied probability. New York,Joh Wiley, 1970.

[06] Benjamin,R.J. & Cornell,C.A. Probability, statistics and decision for engineeres.

[07] Buffa,E.S. Modern production management. New York,John Wiley, 1969.

[08] Dreyfus,S.E. & Law,A.M. The art and theory of dinamic programming. New York, Academic Press, 1977.

[09] Dubois,D. & Prade,H. Fuzzy sets and statistical data. European journal of Operational Research.25(3):345 -56, 1986.

[10] Dubois,D. & Prade,H. Fuzzy sets and systems,theory and applications. New York,Academic Press,1980.

[11] Feller,W. An introduction to probability theory and its applications. New York, J. Wiley,1957. V.1

[12] Gibra,N. Probability and statistical inference scientists

and engineers. New Jersey, Prentice Hall, 1973.

[13] Gnedenmo,B.V. The theory of probability. Moscou,Mir Publishers, 1964.

[14] Goldeberg,S. Probability an introduction. N.Jersey,Prentice Hall, 1960.

[15] Hastings,N.A., Dynamic programming with management applications. London, Butterworth & Co., 1973.

[16] Howard,R.A. Dynamic programming and markov process. Massachusets, M.I.T. Press, 1970.

[17] Kaufmann,A. Dynamic programming: sequential scientif management. N. York, Academic Press, 1967.

[18] Kaufmann,A. On the relevance of fuzzy sets for operational research. European journal of Operational Research,25 (3):380-35,1986.

[19] Kall,P. Stochastic programming. European journal of Operational Research, 10 (20:125-130,1982.

[20] Mayerle,S.F. Caminhos de mínimo custo em grafos e F-grafos:extensões difusas.UFSC,22-5,1988.

[21] Riggs,J.L. Productions systems: planing, analysis and control. N. York, J.Wiley, 1970.

[22] Saaty,T.L. Scaling the merbership function. European journal of Operational Research,25(3):336-44,1986.

[23] Sugeno,M. Fuzzy measures and integrals :A Survey. IN Gupta et alli., 1977.

[24] White,D.S. Dynamic programming. San Francisco,Aberden University Press, 1969.

[25] Zadeh,L.A. Fuzzy sets. Inference control,8:338-53,1965.

[26] Zadeh,L.A. similarity relations and fuzzy ordering. Inference science,3:177-200,1971.

[27] Zadeh,L.A The concept of lingüistic variable and its applications to approximate reasoning.Inference science,8:301-357,9:43-80,1975.

[28] Zadeh,L.A A fuzzy set theoretic interpretation of lingüistic hedges. Journal Cybernetic,3:04-34,1972.

ANEXOS

```
PROGRAM MARKOV;
{+$u}

{este programa calcula os valores esperados de uma politica de deci
sao,referente ao problema de MARKETING, atraves de programacao dina
mica estocastica difusa}
{atualizado em 20/02/89}

{=============================================================}
{               DECLARACAO DE VARIAVEIS                       }

type matrix = array [0..10,0..10] of real;
type matriz = array [0..5,0..5,0..5] of real;
type vetor  = array [0..30] of real;
type mat    = array [0..5,0..5,0..5,0..5] of real;

var
  P,C,SP,SPC,FPROB                                    :MAT;
  R,F,MAXP,MINP                                       :MATRIZ;
  FAUX ,spdter,vt                                     :MATRIX;
  NIVELCAP,ACAONIVEL ,va1,va2,maxza,y,x,a,b,a1
  ,auxmaxza,auxva1,auxva2                          :VETOR;
  XMAX,XMIN,MINZ,MAXZ,CONT,mu,raux1,raux2,px,mi,auxf
  ,x1,y1,x2,y2,yacao,soma                             :REAL;
  MR,K,L,M,S,D,MAX,MIN,ester,n,nivel,i,j,w,acao,t,u,wt,
  continua ,k1 ,h                                     :INTEGER;

{=============================================================}
{                    DESCRICAO DAS VARIAVEIS                  }
{ P    = probabilidade de transicao entre estado
  C    = retornos de transicao associados aos estados
  SP   = spread da probabilidade de transicao
  SPC  = spread do retorno de transicao
  VT   = valor dos estados terminais
  SPDTER = spread dos valores terminais
  R    = retornos associados aos estados  (calculado)
  F    = valor otimo de um estado
  MAXZ= VALOR MAXIMO OTIMO ESPERADO PARA UM ESTADO
  MINZ= VALOR MINIMO OTIMO ESPERADO PARA UM ESTADO
  NIVELCAP=nivel de capacidade ao longo dos anos
  MU   = grau de pertinencia das imformacoes
  CONT= valor utilizado para discretizar o grau de pertinencia
  ESTER= numero de estados terminais
  N     = numero de  anos
  L    = numero de anos remanescentes
  M    = representa um determinado estado
  S    = acao a ser selecionada
  D    = contador utilizado para armazenar os valores difusos
  NIVEL= numero de niveis do problema
  XMAX = maior valor da probabilidade da acao otima selecionada
  XMIN = menor valor da probabilidade da aco otima selecionada
  *demais variaveis - auxiliares                          }



PROCEDURE DADOSMAR;
  var
     arq : text;
  Begin
      assign (arq,'A:dados2.mkl');
      reset (arq);
  read (arq,ester);
  readln (arq,n);
  {== bimf de notas de
```

```
ad (arq,vt[0,k]);
adln (arq,spdter[0,k]);
;
i:=1 to n do
ad (arq,nivelcap[i]);
ln (arq,nivel);
i:=1 to nivel do
adln (arq,acaonivel[i]);


=1 to n do

;
e m (= nivelcap[1] do
in
=1;
ile s(=acaonivel[m] do
egin
for j:=1 to n do
 begin
  read (arq,p[1,m,j,s]);
  read(arq,sp[1,m,j,s]);
  read (arq,c[1,m,j,s]);
  readln (arq,spc[1,m,j,s]);
 end;
s:=s + 1;
nd;
=m +1;
;

arq);
com dadosmar)


URE FUZILEFT;
procedimento executa operacoes com numeros difusos a esquerda
lor esperado)


u =0
n
gin
culo do retorno do estado    )
[1,m,s]:=(p[1,m,k,s] - sp[1,m,k,s]) * (c[1,m,k,s] - spc[1,m,k,s])
         + (p[1,m,k+1,s] -sp[1,m,k+1,s]) * (c[1,m,k+1,s]-spc[1,m,k+1,s]);
uxf:=0;
or j:=1 to n do
 auxf:=auxf+ f[1-1,j,d] * (p[1,m,j,s] - sp[1,m,j,s]);

aux[1,m]:= r[1,m,s] + auxf;
d
e if mu = 1 then
              begin
               r[1,m,s]:=p[1,m,k,s]*c[1,m,k,s] + p[1,m,k+1,s] * c[1,m,k+1,s];
               auxf:=0;
               for j:=1 to n do
                auxf:= auxf + f[1-1,j,d] * p[1,m,j,s];

                fauxf[1,m]:=r[1,m,s] + auxf;
              end
             else
              begin

               rauxi:=(p[1,m,k,s]-sp[1,m,k,s]* mu)*(c[1,m,k,s]-spc[1,m,k,s]*m
               rauxf:=(p[1,m,k+1,s]-sp[1,m,k+1,s]*mu)*(c[1,m,k+1,s]-spc[1,m
```

```
                    auxf:=0;
                    for j:=1 to n do
                     auxf:=auxf + f[l-1,j,d] * (p[l,m,j,s] - sp[l,m,j,s]*mu);
                    faux[l,m]:= r[l,m,s] + auxf;
                   end;
m,d]:=faux[l,m]; }
[l,m,s,d]:=faux[l,m];



URE FUZIDIR;
procedimento calcula o valor esperado a direita de uma numero difu
ara o exemplo do marketing }



 =0
n
gin
[l,m,s]:=(p[l,m,k,s] +sp[l,m,k,s]) * (c[l,m,k,s] + spc[l,m,k,s])
          + (p[l,m,k+1,s] +sp[l,m,k+1,s]) * (c[l,m,k+1,s]+spc[l,m,k+1,s]);
xf:=0;
 j:=1 to n do
egin
auxf:= auxf + f[l-1,j,d] * (p[l,m,j,s] + sp[l,m,j,s]);
writeln('auxf=',auxf:3:2); }
repeat until keypressed;

d;

uxf[l,m]:=r[l,m,s] + auxf; {valor de estado}

 if mu = 1 then
              begin
               r[l,m,s]:=p[l,m,k,s]*c[l,m,k,s] + p[l,m,k+1,s] * c[l,m,k+1,s];
               auxf:=0;
               for j:=1 to n do
                auxf:=auxf + f[l-1,j,d] * p[l,m,j,s];

               faux[l,m]:= r[l,m,s] + auxf;
              end
             else
              begin
               raux1:=(p[l,m,k,s]+sp[l,m,k,s]* mu)*(c[l,m,k,s]+spc[l,m,k,s]*mu

               raux2:= (P[l,m,k+1,s]+sp[l,m,k+1,s]*mu)* (c[l,m,k+1,s]+spc[l,m,
               r[l,m,s]:=raux1 + raux2;
               auxf:=0;
               for j:=1 to n do
                auxf:= auxf + f[l-1,j,d] *(p[l,m,j,s] + sp[l,m,j,s]*mu);
               faux[l,m]:= r[l,m,s] + auxf;

              end;
,d]:=faux[l,m]; }
l,m,s,d]:=faux[l,m];



RE VERIFICA;
ta o calculo do maior valor de estado entre as duas acoes. Esse calculo
 para avaliar os valores de maximo e minimo, i.e, a esquerda e a direi
s valores esperados}
```

```
    then
     begin
      maxp[1,m,d] :=fprob[1,m,s,d]; max:=2;
      acao:=s;
      f[1,m,d]:=fprob[1,m,s,d];
      minp[1,m,d] :=fprob[1,m,s-1,d]; min:=1;
     end
    else
     begin
      {calcula o maior valor de estado entre as duas acoes}
      maxp[1,m,d]:=fprob[1,m,s-1,d]; max:=1;
      acao:=s-1;
      f[1,m,d]:=fprob[1,m,s-1,d];
      minp[1,m,d]:=fprob[1,m,s,d];    min:=2;
     end;
END;



PROCEDURE VALESPMIN;
 {calcula o valor esperadoa esquerda}
 BEGIN
  if p[1,m,k,max] > p[1,m,k+1,max] then
                                        begin
                                             xmax :=p[1,m,k,max];
                                             mr:=k
                                        end
                                       else
                                          begin
                                           xmax:= p[1,m,k+1,max];
                                           mr:=k+1;
                                          end;
  xmin:= 1 - xmax;
  if xmax + xmin = 1 then minz := maxp[1,m,d]{*xmax + minp[1,m,d]*xmin}
                     else
                      repeat
                        xmax :=xmax - p[1,m,mr,max];
                        xmin:= 1 - xmax;
                      until (xmax + xmin = 1);
                      minz := maxp[1,m,d] {* xmax + minp[1,m,d] * xmin};
END;


PROCEDURE VALESPMAX;

  BEGIN
  if p[1,m,k,max] > p[1,m,k+1,max] then
                                        begin
                                             xmax :=p[1,m,k,max];
                                             mr:=k
                                        end
                                       else
                                          begin
                                           xmax:= p[1,m,k+1,max];
                                           mr:=k+1;
                                          end;
  xmin:= 1 - xmax;
  if xmax + xmin = 1 then maxz := maxp[1,m,d]{*xmax + minp[1,m,d]*xmin}
                     else
                      repeat
                        xmax :=xmax - p[1,m,mr,max];
                        xmin:= 1 - xmax;
                      until (xmax + xmin = 1);
  maxz := maxp[1,m,d]{ * xmax + minp[1,m,d] * xmin};
```

```
iteln; writeln;
D;

ROCEDURE COMPARADOR_1 ;

Begin

                        comparacao                                          )


     continua:=1; mi:=0; i:=1;  k1:=1;
     while mi <= 1 do
      begin
          if continua = 1 then
                if va1[i] >= maxza[k1] then
                     begin
                         continua:=2;
                       { gotoxy(8+6*i , w); write (mi:2); }
                     end;
          i:=i+1;
          k1:=k1+1;
          mi:= mi + cont;
      end;

     if continua = 1 then
      begin

           mi:= mi - cont;
           i:=i - 1;
           k1:=k1-1;
           repeat
                 mi:=mi - cont;
                 k1:=k1-1;
                 i:=i+1;
           until (va1[i] >= maxza[k1]) or (mi = 0);
         if va1[i])=maxza[k1] then
         begin
           k :=0;
           wt:=2;
           x[k]:=va1[i-1];
           x[k+1]:=va1[i];
           y[k]:=mi + cont;
           y[k+1]:=mi;
           x[wt]:=maxza[k1+1];
           x[wt+1]:=maxza[k1];
           y[wt]:=mi + cont;
           y[wt+1]:=mi;

{calculando os valores dos coeficientes angulares e lineares}


           a1[k+1]:=(y[k]-y[k+1])/(x[k]-x[k+1]);
           b[k+1]:=(y[k]-x[k]*a1[k+1]);
           a1[wt]:=(y[wt]-y[wt+1])/(x[wt]- x[wt+1]);
           b[wt]:=(y[wt]-x[wt]*a1[wt]);

              resolvendo o sistema                                          )



           x1:=(-b[k+1] + b[wt]) / (-a1[k+1] + a1[wt])*(-1);
           y1:=a1[wt]*x1 + b[wt];
          end
         else
```

```
        end
           else
               y1 := 1.00;
           {writeln('y1=',y1:3:2);}


END;  {COMPARADOR_1}

{-------------------------------------------------------------------------}


PROCEDURE COMPARADOR_2 ;
var x2,y2          :real;
    passa          :integer;
BEGIN

{                          comparacao                                    }


       passa:=1; mi:=0; i:=1;  k1:=1;
       while mi <= 1 do
        begin
            if passa = 1 then
                  if auxva2[i] >= auxmaxza[k1] then

                              passa:=2;

            i:=i+1;
            k1:=k1+1;
            mi:= mi + cont;
        end;
       if passa = 1 then
        begin

            mi:= mi - cont;
            i:=i - 1;
            k1:=k1-1;
            repeat
                  mi:=mi - cont;
                  k1:=k1-1;
                  i:=i+1;
            until (auxva2[i] >= auxmaxza[k1]) or (mi = 0);
          if va2[i] >= auxmaxza[k1] then
           begin
            k :=0;
            wt:=2;

            x[k]:=va2[i-1];
            x[k+1]:=va2[i];
            y[k]:=mi + cont;
            y[k+1]:=mi;
            x[wt]:=auxmaxza[k1+1];
            x[wt+1]:=auxmaxza[k1];
            y[wt]:=mi + cont;
            y[wt+1]:=mi;

  {calculando os valores dos coeficientes angulares e lineares}


            a1[k+1]:=(y[k]-y[k+1])/(x[k]-x[k+1]);
            b[k+1]:=(y[k]-x[k]*a1[k+1]);
            a1[wt]:=(y[wt]-y[wt+1])/(x[wt]- x[wt+1]);
            b[wt]:=(y[wt]-x[wt]*a1[wt]);
```

```
            x2:=(-b[k+1] + b[wt]) / (-a1[k+1] + a1[wt])*(-1);
            y2:=a1[wt]*x2 + b[wt];
          end
         else y2:=0.0;
        end
        else
              y2:=1.0;

      { writeln ('y2=',y2:4:2);
        repeat until keypressed;}
ND;   { COMPARADOR_2}

-------------------------------------------------------------------------------}


ROCEDURE LEDADOS;
 BEGIN
  writeln ('entre com o numero de estados terminais');
  readln (ester);
  writeln ('entre com o numero de anos');
  readln (n);
{==============================================================================
 ==============================================================================
                        LEITURA DOS VALORES TERMINAIS

 ==============================================================================}


 for k:=1 to ester do
  begin
   writeln ('entre com o valor terminal','[',0,',',k,']');
   readln (vt[1,k]);
   writeln ('entre com o spread terminal','[',0,',',k,']');
   readln (spdter[0,k]);
  end;
 for i:=1 to n do
  begin

   writeln ('entre com o nivel de capacidade para o ano',',',i);
   readln (nivelcap[i]);
  end;
writeln ('entre com o numero de niveis');
readln (nivel);
for i:=1 to nivel do
 begin
  writeln ('entre com o numero de acoes para o nivel',',',i);
  readln (acaonivel[i]);
 end;
==============================================================================

  LEITURA DOS RETORNOS DE ESTADO

==============================================================================}

=1;
r l:=1 to n do
egin
m:=1;
while m <= nivelcap[l] do
 begin
  s:=1;
  while s<=acaonivel[m] do
```

```
      for j:=1 to n do
       begin
        writeln ('entre com o ','P','[',l,',',m,',',j,',',s,']');
        readln (p[l,m,j,s]);
        writeln ('entre com o ','SP','[',l,',',m,',',j,',',s,']');
        readln(sp[l,m,j,s]);
        writeln ('entre com o ','C','[',l,',',m,',',j,',',s,']');
        readln (c[l,m,j,s]);
        writeln ('entre com o','SPC','[',l,',',m,',',j,',',s,']');
        readln (spc[l,m,j,s]);
       end;
      s:=s + 1;
     end;
    m:=m +1;
   end;
 end;
end; {com ledados}


=================================================================================
=================================================================================
                        PROGRAMA PRINCIPAL

=================================================================================
=================================================================================)

egin

 DADOSMAR;

{=================================================================
       CALCULO DOS VALORES DE ESTADOS TERMINAIS
=================================================================}

     clrscr;
gotoxy (1,3);write ('ESTADO'); gotoxy (9,3  );write('ACAO');
gotoxy (18,3 );write ('0.0-');gotoxy (26,3  );write('0.5-');
gotoxy (34,3); write ('1.0');
gotoxy (42,3); write ('0.5+');
gotoxy(50,3); write ('0.0+');
gotoxy (58,3); write('GP');
w:=5;
for m:=1 to ester do
 begin
  l:=0; cont:=0.5;
  d:=1; mu:=0;
  while mu <= 1 do
   begin
    if mu = 0 then
                begin
                 f[l,m,d]:= vt[l,m] - spdter[l,m];
                 d:=d+1;
                 f[l,m,d]:=vt[l,m] + spdter[l,m]
                end
              else  if mu = 1 then
                              begin
                                f[l,m,d]:=vt[l,m];
                                d:=d+1;
                                f[l,m,d]:=vt[l,m]
                              end
                             else
                               begin
                                 f[l,m,d]:=vt[l,m]-spdter[l,m]*mu;
                                 d:=d+1;
```

```
                                                  end;
        mu:=mu + cont;
        d:=d + 1;
       end;
    end;
{*************************************************************************}
{=========================================================================}
{        CALCULO DOS VALORES DOS  ESTADOS INTERMEDIARIOS                  }
{=========================================================================}
  w:=5; h:=1;
  for l:= 1 to n do
   begin
    m:=1; d:=1; k:=1;
    while m  <=nivelcap [l] do
      begin
       cont:=0.5; mu:=0; d:=1; t:=1;u:=4; k:=1;
       while mu <= 1 do
        begin
         s:=1;
         while s<=acaonivel[m] do
          begin

           FUZILEFT;


           if s = 2 then begin
                           va1[t]:=fprob[l,m,s-1,d];
                           va2[t]:=fprob[l,m,s,d];
                           VERIFICA;
                         end;

          s:=s+1;
         end;

{=========================================================================}
{              CALCULO DO VALOR MINIMO OTIMO ESPERADO                     }
{=========================================================================}
         VALESPMIN;
         maxza[t]:=minz;
         d:=d + 1;
         s:=1;
         t:=t+u;
         while s <= acaonivel[m] do
           begin
            FUZIDIR;

            if s= 2 then begin
                           va1[t]:=fprob[l,m,s-1,d];
                           va2[t]:=fprob[l,m,s,d];
                          { writeln ('va1[',t,']=',va1[t]:2);
                           writeln ('va2[',t,']=',va2[t]:2);
                           repeat until keypressed;  }
                           VERIFICA;{calcula maior valor de estado}
                         end;
            s:=s+1; {analisa a proxima acao}
           end;
{=========================================================================}
{              CALCULO DO VALOR MAXIMO OTIMO ESPERADO                     }
{=========================================================================}
         VALESPMAX;
         maxza[t]:=maxz;
         d:=d+1;
         t:=t-u;
       { gotoxy (5,w); write(l:1,',',m:1); gotoxy (17,w); write(MU:1);
```

```
        gotoxy(57,w); write (acao);
        gotoxy(63,w); write (xmax:2); }
        u:=u-2;
        t:=t+1;
        mu:= mu + cont; {proximo grau de pertinencia}
      { w:=w+1; impressao dos resultados      }
       end;

      for i:=1 to 5 do
       begin
           auxva1[i]:=va1[i];
           auxva2[i]:=va2[i];
           auxmaxza[i]:=maxza[i];
        end;


{-------------------------------------------------------------------------}
 {                          comparacao


       continua:=1; mi:=0; i:=1;  k1:=1;
       while mi <= 1 do
        begin
            if continua = 1 then
                  if va1[i] >= maxza[k1] then
                       begin
                           continua:=2;
                       end;
            i:=i+1;
            k1:=k1+1;
            mi:= mi + cont;
        end;

       if continua = 1 then
        begin

             mi:= mi - cont;
             i:=i - 1;
             k1:=k1-1;
             repeat
                    mi:=mi - cont;
                    k1:=k1-1;
                    i:=i+1;
             until (va1[i] >= maxza[k1]) or (mi = 0);
             k :=0;
             wt:=2;
             x[k]:=va1[i-1];
             x[k+1]:=va1[i];
             y[k]:=mi + cont;
             y[k+1]:=mi;
             x[wt]:=maxza[k1+1];
             x[wt+1]:=maxza[k1];
             y[wt]:=mi + cont;
             y[wt+1]:=mi;

   calculando os valores dos coeficientes angulares e lineares


             a1[k+1]:=(y[k]-y[k+1])/(x[k]-x[k+1]);
             b[k+1]:=(y[k]-x[k]*a1[k+1]);
             a1[wt]:=(y[wt]-y[wt+1])/(x[wt]- x[wt+1]);
             b[wt]:=(y[wt]-x[wt]*a1[wt]);

               resolvendo o sistema
```

```
            xi:=(-b[k+1] + b[wt]) / (-a1[k+1] + a1[wt])*(-1);
            y1:=a1[wt]*xi + b[wt];
          end; )
          COMPARADOR_2;

          COMPARADOR_1;

          i:=1;
          while i <= 5 do
           begin
               gotoxy(10,w); write('1');
               gotoxy(3,w+1); write (l:1, ' , ', m:1);
               gotoxy(10,w+2); write ('2');
               gotoxy(10,w+4); write('MAX');
               gotoxy(10+8*i,w); write (auxva1[i]:3:2);
               gotoxy(10+8*i,w+2);  write (auxva2[i]:3:2);
               gotoxy (10+8*i, w+4); write (auxmaxza[i]:3:2);
               i:=i+1;
           end;
           gotoxy (58,w);write (y1:4:2);

           gotoxy(58,w+2); write (y2:3:2);
{-------------------------------------------------------------------------------



      m:=m + 1 ; {proximo estado}
      w:=w+6;
      {writeln; writeln;}
     end;
   end;
end.
```

```
AM DIFUSO;
 programa calcula o maximo valor de estado, utilizando o algoritmo
rog. dinamica, e tambem compara o grau de pertinencia entre dois
ros difusos.)

 matriz = array[0..10,0..10,0..10] of real;
 matrix = array[0..10,0..10] of real;
 vetor  = array[0..100] of real;

                                        :char;
,spdre,f,acaotima,fd,valor,v            :matriz;  (tridimensional)
aux,spdter ,vt                          :matrix;  (bidimensional)
os ,nivelcap,acaonivel,x,y,a1,b         :vetor;
,m,s,j,d,a,ester,n,i,nivel,k,ad,w,t,continua :integer;
u,aux,max,cont,conte,mi,x1,y1           : real;

URE DADOSDET;

:text;

 assign (arq,'A:dados.det');
 reset (arq);
 read (arq,ester);
 readln (arq,n);

 for k:=1 to ester do
  begin
     l:=0;
     read (arq, vt[l,k]);
     readln (arq,spdter[l,k]);
  end;
 for i:=1 to n do
  read (arq,nivelcap[i]);
readln (arq,nivel);
for i:=1 to nivel do
  readln (arq,acaonivel[i]);

 (LEIRURA DOS RETORNOS DE ESTADO)
for l:=1 to n do
  begin
    m:=1;
    while m <=nivelcap[l] do
     begin
       s:=1;
       while s<=acaonivel[m] do
         begin
             read(arq,r[l,m,s]);
             readln (arq,spdre[l,m,s]);
             s:=s + 1;
         end;
       m:=m + 1;
     end;
  end;
ose (arq);
COM DADOSDET);

DURE FUZILEFT; (difuso a esquerda)
e procedimento executa operacoes com numeros difusos a esquer-
 determinando o maximo valor de estado )
N (inicio)
  if mu = 0 then faux[l,m]:=(r[l,m,s] - spdre[l,m,s]) + f[l-1,j,d](-spdter[l-
            else if mu =1 then faux[l,m]:= r[l,m,s] + f[l-1,j,d]
                          else faux [l,m]:=(r[l,m,s] - spdre[l,m,s] * mu) + f
```

```
LDURE FUZIDIR ;{difuso a direita}
te procediemto executa operacoes com numeros difusos a esquerda}
IN
      if mu = 0 then fauxE1,mE:=(rE1,m,sE + spdreE1,m,sE) + fE1-1,j,dE{+spdter
                else faux E1,mE:=(rE1,m,sE + spdreE1,m,sE * mu) + fE1-1,j,dE;
D; {com difusodir}
EDURE VERIFICA; {calculo do maximo valor de estado}
in
 aux:=fauxE1,mE;
 if aux ) max then
   begin
       a:=s;
       max:=aux;
       fE1,m,dE:=max;
       acaotimaE1,m,dE:=a;
   end;
d;
DURE COMPARA;
e procedimento executa operacoes de comparacao entre dois estados
a determinado grau de pertinencia}
n
=100;
r l:= 1 to n do
egin
  m:= 1;
  while m <= nivelcapE1E do
   begin
      d:=1; ad:=1;  t:=10;
      s:=1;
      cont:=0.5; mu:=0;
      while s <= acaonivelEmE do
       begin
         d:=1; ad:=1;
         k:=m+s-1;
         writeln; writeln;
ln('ESTADO','(',1-1,' ',k,')',' ','X',' ','ESTADO','(',1,' ',m,')');
         mu:=0;
         writeln; writeln;
         while mu <= 1 do
           begin
             j:= m + s - 1;
             gotoxy(t,i);write (fdE1-1,j,dE:2:2);
             gotoxy (t,i);write ('      ');
             d:=d + 1;
             mu:=mu + cont;
             t:=t+10;
           end;
         mu:= mu - 2*cont;
         while  mu )= 0 do
          begin
            j:= m + s - 1;
            gotoxy(t,i);write (fdE1-1,j,dE:2:2);
            gotoxy (t,i);write('      ');
            mu:= mu - cont;
            t:=t+10;
            d:= d+1;
          end;


       {compara os valores de estado sucessor}
       BEGIN
        writeln;
        writeln;
        mi:=0;
        conte:=0.5; t:=10;
```

```
        begin
         gotoxy(t,i+2);write (f[1,m,ad] :2:2);
         gotoxy(t,i+2);write('      ');
         ad:=ad + 1;
         mi:=mi + conte; t:=t+10;
        end;
       mi:= mi - 2 * conte;
       while mi >= 0 do
        begin
         gotoxy(t,i+2);write (f[1,m,ad] :2:2);
         gotoxy(t,i+2);write ('      ');
         mi:=mi - conte; t:=t+10;
         ad:= ad + 1;
         end;
      END; (com comparacao sucessor)



   continua :=1;    mu:=0;
   d:=1;  ad:=1;  t:=10;
   while mu <= 1 do
    Begin
      if continua = 1 then
                      if fd[1-1,j,d] >= f[1,m,ad]
                        then
                          begin
                           continua :=2;
                           gotoxy(t,i+4);write('F','(',1-1:1,j:1,')','>=
                           gotoxy(t,i+6); write ('GP','=',1.0:1:2);
                          end;
      d:=d + 1; ad:=ad+1;
      mu:=mu + cont;
     end;

    if continua = 1 then
                    Begin
                    mu:=mu - cont;
                    d:=d-1;
                    ad:=ad-1;
                     Repeat
                       d:=d+1;
                       ad:=ad-1;
                       mu:=mu - cont;
                      Until (fd[1-1,j,d] >=f[1,m,ad]) or (mu =0);
     k:=0; w:=2;
{-------------------------------------------------------------------}

     x[k]:=fd[1-1,j,d-1];     x[k+1]:=fd[1-1,j,d];
     y[k]:=mu+cont;           y[k+1]:=mu;
{-------------------------------------------------------------------}

{-------------------------------------------------------------------}
     x[w]:=f[1,m,ad+1];       x[w+1]:=f[1,m,ad];
     y[w]:=mu+cont;           y[w+1]:=mu;
{-------------------------------------------------------------------}

{CALCULANDO OS VALORES DOS COEFICIENTES ANGULARES E LINEARES        }

[k+1]:=(y[k] - y[k+1] ) / (x[k] - x[k+1]);
+1]:=(y[k] - x[k] * a1[k+1]);
w]  :=(y[w] - y[w+1]) / (x[w] - x[w+1]);
]  :=(y[w] - x[w] * a1[w]);

     RESOLVENDO O SISTEMA                                          }
```

```
y1:= a1*w1 + x1 + b1*w1;
gotoxy(t,i+3);write ('F','(',l-1,j,')', ')=','F','(',l,m,')', '   ', 'GP','='
gotoxy(t,i+4);writeln (' ','X','=',x1:2:2);
end;
            i:=i+10;
            s:=s + 1;
          end;
          m:=m + 1;    {i:=i+10;}
        end;
      end;
      {i:=i+10;}
    end;
{=============================================================================}
{=============================================================================}

PROCEDURE  PERTINENCIA;
{IMPRIMA VALORES DOS GRAUS DE PERTINENCIA}
BEGIN
 writeln ('valores dos graus de pertinencia');
 writeln ('==========================================================================');
 writeln ('**************************************************************************');
 writeln ('==========================================================================');
 cont:=0;
 while cont <= 1 do
  begin
    write ( cont :2);
    cont := cont + 0.5;
  end;
 cont:= 0.5;
 while cont >= 0 do
  begin
    write ( cont :2);
    cont:= cont - 0.5;
  end;
END;
PROCEDURE MAXPERT;
{calcula os valores difusos dos estados correntes mais os retornos,
 com o estado sucussor.                                            }
BEGIN
  For l:=1 to n do
   begin
    m:=1;
    while m <= nivelcap[l] do
     begin
      mu:=0; cont:=0.5;
      d:=1; s:=1;
      while s <= acaonivel[m] do
       begin
        d:=1;
        mu:=0;
        while mu <=1 do
         begin
          j:= m + s - 1;
          fd[l-1,j,d] :=0;
          if mu =0 then fd[l-1,j,d]:=f[l-1,j,d]+(r[l,m,s]- spdre[l,m,s])
                   else if mu=1 then fd[l-1,j,d]:=f[l-1,j,d] + r[l,m,s]
                   else fd[l-1,j,d]:=f[l-1,j,d]+(r[l,m,s]-spdre[l,m,s]*mu);

          d:=d+1;
          mu:=mu + cont;
         end;
        mu:= mu - 2 * cont;
        while mu >= 0 do
         begin
          fd[l-1,j,d]:=0;
```

```
                else fd[l-1,j,d]:=f[l-1,j,d]+(r[l,m,s]+spdre[l,m,s])*mu);
            mu:=mu - cont;
            d:=d + 1;
           end;
          s:=s + 1;
         end;
         m:=m + 1;
        end;
       end;
      end; {com procedimento maxpert}
{=============================================================================}
{=============================================================================}

PROCEDURE SUCESSOR;
{compara os valores de esrado sucessor}
BEGIN
 mi:=0;
 conte:=0.5;
 while mi <=1 do
  begin
   write (f[l,m,ad] :2);
   ad:=ad + 1;
   mi:=mi + 1;
  end;
 mi:= mi - 2 * conte;
 while mi >= 0 do
  begin
   write (f[l,m,ad] :2);
   mi:=mi + conte;
   ad:= ad + 1;
  end;
END; {com procedimento sucessor}

{=============================================================================}
{*****************************************************************************}
{=============================================================================}

PROCEDURE ENTRADA;
BEGIN
  writeln ('entre com o numero de estados terminais');
  readln(ester);

  for k:=1 to ester do
    begin
      l:=0;
      writeln ('entre com o valor terminal','[',l,',',k,']');
      readln( vt[l,k]);
      writeln ('entre com o spread terminal','[',l,',',k,']');
      readln (spdter[l,k]);
    end;
writeln ('entre com o numero de anos');
readln (N);
for i:=1 to n do
 begin
     writeln ('entre com o nivel de capacidade para o ano',',',i);
     readln (nivelcap[i]);
 end;
writeln ('entre com o numero de niveis');
readln (nivel);
for i:=1 to nivel do
 begin
     writeln ('entre com o numero de acoes para o nivel',',',i);
     readln (acaonivel[i]);
 end;
 for i:=1 to n do
```

```
    m:=1;
    while m <= nivelcap[i] do
     begin
         spdter[i,m]:=0; zerar spred dos val. de estados intermediario e fina
         m:=m + 1;
     end;
end;}

EIRURA DOS RETORNOS DE ESTADO}
r l:=1 to n do
 begin
     m:=1;
     while m <=nivelcap[l] do
      begin
          s:=1;
          while s<=acaonivel[m] do
            begin
                writeln('entre com o r','[',',',l,',',m,',',s,']');
                readln(r[l,m,s]);
                writeln ('entre com o spread','[',l,',',m,',',s,']');
                readln (spdre[l,m,s]);
                s:=s + 1;
            end;
          m:=m + 1;
       end;
  end;

; {COM ENTRADA}


================================================================================

                          PROGRAMA PRINCIPAL

================================================================================}

IN
DOSDET; {carrega dados}
rscr;
toxy(5,8);write ('ESTAGIO':7);gotoxy(16,8);write('ESTADO');
nt:=0; k:=30;
ile cont <= 1 do
begin
 gotoxy(k,8);write(cont:2:2);
     cont:= cont + 0.5;
     k:=k+10;
end;
nt:=0.5;
ile cont >= 0 do
egin
   gotoxy(k,8);write (cont:1);
   cont:= cont - 0.5;
   k:=k+10;
nd;
iteln;
mpressao dos valores de estado terminais}
=9;
r m:=1 to ester do
begin
   l:=0; k:=17;w:=30;
   gotoxy(5,i); write (l:2);gotoxy(16,i);write(m:2);
   d:=1;
   mu:=0; cont:=0.5;
   while mu <= 1.0 do
    begin
```

```
                else if mu=1 then f[1,m,d]:= vt[1,m]
                                else f[1,m,d]:= vt[1,m] - spdter[1,m] * mu;
        mu:= mu + cont;
        gotoxy(w,i);write (f[1,m,d] : 2:1);
        d:= d + 1;
        w:=w+10;
    end;
  mu:= mu - 2* cont;
  while mu >= 0 do
   begin
          if mu = 0 then f[1,m,d]:= vt[1,m] + spdter[1,m]
                    else f[1,m,d]:= vt[1,m] + spdter[1,m] * mu;

          gotoxy(w,i);write (f[1,m,d] : 2:1);
          d:= d + 1;
          w:=w+10;
          mu:= mu - cont;
    end;
    writeln;
    i:=i+2;
end;
{calculo dos valores de estado para determinado grau de pertinencia}
:=15;

for 1:=1 to n do
 begin
    m:=1; {estado}
    w:=30;
    while m <=nivelcap[1] do
      begin
          gotoxy(5,t);write (1:2);gotoxy(16,t);write(m:2);
          cont:=0.5; mu:=0;w:=30;
          d:=1;
          while mu <= 1  do
           begin
               s:=1;
               max:= -9999;
               while s<=acaonivel[m] do
                begin
                    j:=m + s - 1; {equacao de transicao}
                    fuzileft; {procedimento}
                    verifica; {procedimento}
                    s:= s + 1;
                 end;

               gotoxy(w,t);write(f[1,m,d]:2:1);
               mu:= mu + cont;
               d:=d + 1; w:=w+10;
            end;
          mu:= mu - cont*2;
          while mu >= 0  do
           begin
               s:=1;
               max:= -9999;
               while s<=acaonivel[m] do
                begin
                    j:=m + s - 1; {equacao de transicao}
                    fuzidir; {procedimento}
                    verifica; {procedimento}
                    s:= s + 1;
                 end;
               gotoxy(w,t);write (f[1,m,d]:2:1);
               mu:= mu - cont;
               d:=d + 1; w:=w+10;
            end;
```

```
                writeln; {proximos valores de estado}
                m:= m + 1;
                t:=t+1; w:=w+10;

                writeln; writeln;writeln;
            end;
        t:=t+2;
    end;
    COMPARA; {compara valores difusos entre estados}

end.
```