UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
DEPARTAMENTO DE TELEMÁTICA

Lógica Nebulosa e Programação Linear Nebulosa Aplicadas a Problemas de Programação Horária de Peças em Células Flexíveis de Manufatura

Este trabalho é parte do requisito para obtenção do título de **Doutor em Engenharia Elétrica** pela Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas

Autor: **Pedro Reumay Romero**
Orientador: **Akebo Yamakami**

Campinas - São Paulo
Julho/1996

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

~~R318L~~

~~Reumay~~ Romero, Pedro
Lógica nebulosa e programação linear nebulosa aplicadas a problemas de sequenciamento de peças em células flexíveis de manufatura / Pedro Reumay Romero.--Campinas, SP: [s.n.], 1996.

Orientador: Akebo Yamakami.
Tese (doutorado) - Universidade Estadual de Campinas Faculdade de Engenharia Elétrica.

1. Conjuntos nebulosos. 2. Programação linear. 3. Produção - sequenciamento. 4. Programação heurística. I. Yamakami, Akebo. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica. III. Título.

Dedico este trabalho a Cristina.

I

Agradecimentos

Agradeço ao Professor Dr. Akebo Yamakami do Departamento de Telamática da Faculdade de Engenharia Elétrica, Unicamp, pela dedicação e incentivo durante a orientação dessa tese.

Gostaria também de agradecer ao Instituto de Matemáticas da Universidade Austral de Chile por ter dado condições ao desenvolvimento desse trabalho.

Agradeço ao CNPq pelo apoio financeiro.

Finalmente, agradeço a minha familia em especial a minha Irmã Cristina que contribuira sensívelmente para a realização desse trabalho.

Resumo

Neste trabalho definem-se as características de programação horária de peças, e sua solução, como problema de otimização combinatorial. É proposto um modelo formal para sua representação em uma célula flexível de manufatura com base na teoria de programação matemática, onde é modelado como um problema de programação linear inteira misto com o objetivo de minimizar o tempo total de processamento.

Apresenta-se também, as idéias básicas da teoria de conjuntos nebulosos e sua contribuição ao desenvolvimento de modelos de tomada de decisão. Obtém-se soluções aproximadas do problema usando a lógica nebulosa e a programação linear nebulosa.

Para a implementação da heurística baseada na lógica nebulosa define-se uma base de regras do tipo if-then associadas as variáveis linguísticas para resolver os gargalos. Para a implementação da heurística baseada na programação linear nebulosa usa-se a extensão de Zimmermann (1976), Werners (1987) e Delgado at al.(1989).

Finalmente, apresenta-se exemplos para testar as heurísticas propostas e analisa-se os resultados comparativamente a programações realizadas segundo regras tradicionais.

Os resultados evidenciam o bom desempenho da programação horária baseada na teoria de conjuntos nebulosos, principalmente no que se refere a uma melhor ponderação do processo de otimização do tempo total da programação horária de peças.

Palavras Chaves: **Lógica Nebulosa, Programação Linear Nebulosa, Sequenciamento de Peças, Células Flexíveis de Manufatura.**

Abstract

In this work, the workpieces scheduling problem is charcterized and modeled as a combinatorial optimization problem. A formal model for flexible manufacturing cell is presented, based on the mathematical programming theory: a mixed integer programming model minimizing the total processing time.

The basic concepts of fuzzy logic and fuzzy numbers and sets is roughly presented and their contributions to the decision making models development are enfatized, using fuzzy logic and fuzzy linear programming.

In the implementation of fuzzy logic based heuristic, a set of rules to linguistic variables are defined, in order to solve the botlenecks. Also, fuzzy linear programming theory extensions of Zimmermann (1976), Werners (1987) and Delgado at al. (1989) has been used to develop another heuristic.

The proposed methodologies are applied in same examples and the results are analised and compared with those found using the traditional methods.

The results showed that our approach using both fuzzy linear programming and fuzzy logic are more realistic in the real world sense and that is computationaly efficient.

Keywords: **Fuzzy Logic, Fuzzy Linear Programming, Scheduling, Flexible Manufacturing Cell.**

Sumário

Lista de Figuras

Lista de Tabelas

Glossário

AGV -	(Automatic Guided Vehicles): dispositivo no qual um pallet é transportado de uma workstation a outra.
BUFFER -	Local para armazenamento das peças
CÉLULA -	Conjunto de workstations.
CIM -	Manufatura integrada por computador
FCFS -	Primeiro que chega é o primeiro a ser executado
FIFO -	Primeiro que chega é o primeiro que sai
FL-	Lógica nebulosa
FLP-	Programação linear nebulosa
FLOW-SHOP -	Produção de peças em uma mesma ordem nas máquinas
FMC -	Célula flexível de manufatura
FMS -	Sistema flexível de manufatura
JOB -	Módulo de tarefas básicas
JOB-SHOP -	Produção de peças em uma ordem qualquer nas máquinas
LAYOUT -	Projeto de espaço físico
LCFS -	Último que chega é o primeiro a ser executado
LPT -	Tempo de processamento máximo
LWKR -	Operação associada ao job com menor quantidade de trabalho restante a ser processado
MAKESPAN -	Tempo total de processamento
MAGAZINE -	Local para armazenamento de ferramentas
MÁQUINA -	Peça do hardware do FMS na qual é executada uma operação de manufatura sobre uma parte.
MOPR -	Operação associada ao job com maior número de operações restantes a serem processadas
MWKR -	Operação associada ao job com maior quantidade de trabalho restante a ser processado
PALLET -	Dispositivo no qual uma parte é posicionada durante todas as operações de manufatura e manuseio de materiais.
SET-UP -	Preparação da máquina
SPT -	Tempo de processamento mínimo
VL -	Conjuntos de valores linguísticos.

Introdução Geral

Os problemas de programação horária surgem quando se depara com situações que precisam da alocação temporal de recursos para um conjunto de atividades. Tipicamente, estes problemas podem ser vistos como da determinição de uma programação horária ótima para as atividades sobre os recursos e vice-versa.

Quando as atividades são determinísticas, estes problemas podem ser formulados como problemas de otimização combinatorial no sentido clássico. Embora por muito tempo tem-se usado métodos de programação horária informais e ainda estão em uso em muitas atividades, modelos de programação horária formais originaram-se durante a primeira guerra mundial com a chegada dos gráficos de Gantt.

Estes modelos de gráficos foram desenvolvidos para selecionar os problemas de programação horária associados com a alocação de cargas nos barcos aliados. Seu uso diminuiu o tempo de volta dos barcos em quase metade. O método de caminhos críticos, sucessores dos gráficos de Gantt, continua sendo uma ferramenta de programação horária amplamente usada.

Modelos matemáticos formais de problemas de programação horária começaram a aparecer na literatura nos meados dos anos cinquenta, aproximadamente quarenta anos depois do trabalho de Gantt.

Desde então, o interesse nestes problemas tem aumentado, como é evidenciado pelas inúmeras publicações que têm aparecido na área de pesquisa operacional e engenharia industrial, tais como Baker(1974), Rinnoy Khan (1976).

Apesar do esforço gasto nas análises destes modelos de programação horária, a teoria unificadora é limitada.

Desenvolvimentos recentes da teoria da complexidade permite a classificação dos problemas de programação horária de acordo com o desempenho dos algoritmos que podem ser construídos para sua solução, como problemas de solução por algoritmos bons ou eficientes no sentido de exigir um número de passos que é limitado

por uma função polinomial da dimensão do problema.

Muitos de tais problemas podem ser formulados na forma de problemas de otimização em redes bem resolvidos (tais como caminho mais curto, alocação, fluxo de custo mínimo, caminhos críticos, etc) ou na forma de problemas de ordenamento simples.

A maioria dos problemas de programação horária são da classe de problemas combinatoriais NP completos (Lenstra and Rinnooy Khan 1978), para os quais o desenvolvimento de algoritmo eficientes fracassa.

É interessante notar que muitos dos resultados disponíveis na teoria da complexidade representa o pior caso e, quando caracteriza o comportamento do caso médio, necessita de algoritmos com heurísticas.

A diversidade das contribuições em programação horária está na necessidade de resolver estes problemas numa ampla variedade de contextos, sejam eles NP completos ou não.

Atualmente há uma grande motivação para desenvolver técnicas especializadas que sejam eficientes para uma dada classe de problemas mas não são necessàriamente transferíveis e aplicáveis a outras classes. Na prática, problemas de programação horária são resolvidos confiando na intuição, ou seja, usando heurísticas ou alguma forma de enumeracão explícita ou implícita.

Apesar de que tais métodos, via de regra, apresentam limitações óbvias, eles fornecem soluções satisfatórias para muitas aplicações práticas. As dependências em heurísticas têm aberto caminhos em linhas de pesquisa que tratam de responder à seguinte questão: *"Se uma heurística é usada, qual é o pior erro que pode resultar de sua aplicação?"* Esta questão pode ajudar na identificação de heurísticas eficientes e robustas, e na eliminação de outras.

Os trabalhos sobre este tema refletem as áreas de interêsse em pesquisa de programação horária que temos descrevido acima. Eles incluem resultados relativos a complexidade para classificação de problemas, análises de heurísticas, algoritmos e-

numerativos baseados em métodos branch-and-bound e programação dinâmica, bem como novos modelos de programação horária que surgem de situações reais ou hipotéticas.

Uma teoria de decisão é uma teoria que incorpora minimização de alguma função de custo ou maximização de uma função de utilidade. Pela importância desta área, há um grande número de teorias sobre o assunto, podendo classificá-las do ponto de vista matemático, em quatro categorias:

- tomada de decisão vista como uma otimização da utilidade esperada,
- tomada de decisão vista como uma otimização com restrições,
- otimização multi-pessoal que se refere à otimização do número de pessoas (planejadores),
- otimização multi-critério.

Esta divisão é pragmática pois atinge uma divisão nas teorias matemáticas de decisão. A correspondência entre estas quatro categorias e a teoria de decisão convencional é a seguinte:

- Maximização da utilidade esperada - Teoria estatística de decisão.
- Otimização com restrições - Programação matemática.
- Otimização multi-pessoal - Teoria de jogos.
- Otimização multi-critério - Tomada de decisão multicritério.

A tomada de decisão em ambiente nebuloso descreve a situação de decisão vista como um problema de otimização com restrições. Para isto separa a situação de decisão em variáveis de decisão, restrições e metas, aplicando a noção de nebulosidade às duas últimas e considerando as variáveis de decisão como sendo determinísticas (no máximo probabilísticas).

Na teoria de Bellman, and Zadeh (1970), tanto a função objetivo como as restrições são caraterizadas pelas suas funções de pertiência. A meta é satisfazer objetivos e restrições.

Este trabalho têm como objetivos principais:

- Propor uma solução factível ao problema de programação horária usando uma metodologia baseada na lógica nebulosa.
- Propor uma solução factível ao problema de programação horária usando uma metodologia baseada na programação linear nebulosa.
- Considerar metodologias que permitam o tratamento de variações no tempo de transporte e de execução de tarefas e de situações conflitantes entre si.
- Testar as metodologias através de exemplos.

No capítulo 1, define-se o problema de programação horária, descreve-se sua complexidade e aborda-se as características dos ambientes industriais onde a programação horária ocorre.

Em seguida apresenta-se, de forma sucinta, os conceitos básicos sobre conjuntos nebulosos, lógica nebulosa e programação linear nebulosa.

No capítulo 2 apresenta-se os conceitos básicos sobre as células flexíveis de manufatura (FMC) e é também apresentado um modelo matemático para o problema de programação horária a ser utilizado ao longo deste trabalho.

No capítulo 3 são apresentadas as nossas propostas de heurísticas baseadas na lógica nebulosa e programação linear nebulosa.

No capítulo 4 são apresentados exemplos nos quais as heurísticas foram aplicadas e sugere-se uma análise interativa, por parte do planejador, usando os resultados fornecidos em tabelas, com o objetivo de atingir uma melhor apreciação da aproximação dos resultados esperados.

O capítulo 5 contém as conclusões do trabalho e sugestões para trabalhos futuros.

Capítulo 1
Fundamentos matemáticos e metodológicos

Neste capítulo é apresentado uma rápida revisão bibliográfica sobre problemas de sequenciamento de peças e de programação horária. Também são apresentados os conceitos básicos sobre lógica nebulosa e sobre programação linear nebulosa, como subsídio para os capítulos que seguem.

1.1 Programação horária

Esta seção apresenta um breve levantamento bibliográfico na área de programação horária em Células Flexíveis de Manufatura (FMC) e Sistemas Flexíveis de Manufatura (FMS). A programação horária consiste na alocação de *recursos* no tempo para a execução de uma série de *tarefas*. No vocabulário de manufatura os *recursos* são as máquinas, as tarefas são chamados "jobs" e as tarefas elementares das quais está constituída uma tarefa (job) são referidas como *operações ou estágios*.

Em vários artigos tais como Ashour(1970), Bestwick and Hastings (1976), Cunto (1973), Gonzalez and Sahni (1978), Gupta (1975), Inman and Jones (1993), Johnson (1954), Lageweg at al. (1976), Lageweg at al. (1978), Lenstra and Rinnooy Kan (1978), Morton and Smunt (1986), Reumay and Yamakami (1995), encontra-se uma descrição do problema de programação horária dentro do seguinte cenário: dado um conjunto de n tarefas (peças, jobs, commodities, que denotamos por $I = \{1, \ldots, n\}$) e dado um conjunto de m máquinas (facilities, que denotamos por $J = \{1, \ldots, m\}$), um problema que ocorre frequentemente é o de encontrar uma ordem na qual as tarefas podem ser processadas nas máquinas de forma a otimizar uma função objetivo definida.

Se as peças são processadas de tal maneira que a ordem tecnológica de todas as peças sobre todas as máquinas é a mesma, então o problema é dito ser um

problema de programação horária do tipo Flowshop. Se a ordem do processamento é restrita ser a mesma sobre cada máquina, neste problema existem $n!$ diferentes sequências de job possíveis para cada máquina (flowshop de permutação) e, portanto, $(n!)^m$ diferentes soluções a serem examinadas.

Se as peças são processadas numa ordem tecnológica qualquer, então o problema é dito ser um problema de programação horária do tipo Jobshop. A ordem das peças, neste caso, não precisa ser a mesma para cada máquina e, portanto, é possível ter $(m!(n!)^m)$ combinações de soluções.

Para tornar estes problemas solúveis, é preciso introduzir simplificações. As suposições mais comuns são:

1. Os tempos de processamento dos jobs são quantidades determinísticas (conhecidas, sem variações).

2. As máquinas podem processar um job de cada vez e uma vez que começa o processo, deve terminar.

3. O tempo de transporte e de setup ou são ignorados ou são incluídos nos tempos de processamento e não dependem da ordem.

4. No início, todos os jobs estão prontos para processamento e todas as máquinas estão prontas para processá-los.

5. Todos os jobs têm igual prioridade e são independentes.

6. Há somente uma máquina de cada tipo.

7. Os jobs aceitos para o processamento nas máquinas não podem ser cancelados.

As técnicas de programação horária buscam determinar a programação horária que minimiza ou maximiza uma medida bem definida. A medida mais comum para representar a programação horária é o *makespan* , definido como o

tempo total no qual todos os jobs completam seu processo sobre todas as máquinas. No entanto, cabe lembrar que esta não é a única medida possível.

A dificuldade para desenvolver técnicas exatas de otimização para a solução do problema geral de programação horária têm levado vários autores a considerarem problemas com estruturas especiais.

Por exemplo, Johnson (1954) considera o problema especial de três máquinas onde uma condição sobre o tempo de vários jobs relacionados permite desenvolver um algoritmo de otimização simples e eficiente para a obtenção da programação horária desejada. Lageweg at al. (1978) consideram Enumeração Implícita para programação horária de jobshop, Ashour(1970) desenvolve um Algoritmo Branch -and-Bound para flowshop.

Baker (1974) afirma que a teoria sobre programação horária existente se caracteriza por uma abordagem quantitativa obtida através de modelos matemáticos. Para classificar os modelos de programação horária é necessário caracterizar a configuração das máquinas e o comportamento das tarefas.

No desenvolvimento de modelos de programação horária mais gerais do que o modelo de uma máquina, Johnson (1954) desenvolveu um algoritmo de programação horária ótimo aplicável a uma extensa classe de flowshop.

Garey et al. (1978) afirma que os problemas de programação horária são basicamente problemas de otimização da seguinte forma: dada uma coleção de tarefas para serem programadas em um sistema de processamento particular sujeito a várias restrições, encontrar uma programação horária factível que minimize (ou em alguns casos que maximize) o valor de uma dada função objetivo. Assim, não é surpreendente que muitos dos trabalhos da teoria de programação horária tenham sido desenvolvidos para o projeto e análise de algoritmos de otimização que constróem uma programação horária factível para uma dada instância.

Desafortunadamente, a meta de desenvolver algoritmos de otimização eficientes tem-se evidenciado mais difícil de atingir. De fato, todos os problemas de otimização de programação horária são considerados sem solução ótima, exceto para

problemas de pequeno porte especialmente estruturados. Este pessimismo têm sido confirmado por resultados que mostram que a maioria dos problemas de programação horária pertencem à classe de problemas NP-completo (Lenstra and Rinnooy Kan, (1978)).

Para muitos problemas de programação horária existem algoritmos heuristícos simples que determinam rapidamente um sequenciamento factível. Neste ponto Garey et al (1978) introduz o seguinte formalismo: um problema de programação horária consiste de duas partes, um *modelo* e uma *função objetivo*. O *modelo* descreve o sistema incluindo a classe de tarefas e as restrições em seu processamento, os tipos de processadores e seu número e todas as outras propriedades necessárias para especificar a programação horária factível. A *função objetivo* designa um valor a cada programação horária factível. Uma amostra de tal problema é somente uma especificação de um sistema particular, conjunto de tarefas e conjunto de restrições conformando o modelo. Dado um caso I, o modelo expressa a forma da programação horária factível para I e a função objetivo expressa quais são seus valores.

Lenstra and Rinnoy Kan (1978) estudam a complexidade do problema de scheduling sob restrições de precedência. Identificam como $\mathcal{P}$ uma classe de problemas para os quais existe um algoritmo bom (ou polinomialmente limitado), enquanto que todos os problemas que podem ser resolvidos por um algoritmo não determinístico de tempo polinomial, são identificados como $\mathcal{NP}$. Afirmam que é claro que $\mathcal{P} \subset \mathcal{NP}$, e a questão que surge é se esta inclusão é própria ou se, pelo contrario, $\mathcal{P} = \mathcal{NP}$. Apesar de que esta questão permaneça aberta, a igualdade de $\mathcal{P}$ e $\mathcal{NP}$ é considerada muito improvável pois $\mathcal{NP}$ contém muitos problemas combinatórios notáveis para os quais não têm encontrado algoritmos eficientes até agora.

Além disso, a relação entre $\mathcal{P}$ e $\mathcal{NP}$ é obtida introduzindo os seguintes conceitos:

"O problema P' é redutível ao problema P (notação: $P' \propto P$) se dado qualquer modelo para P' existe um algoritmo de tempo polinomial que resolvendo

P resolve também P'''.

$$P \text{ é NP-completo se } P \in \mathcal{NP} \text{ e } P' \propto P \ \ \forall P' \in \mathcal{NP}$$

Informalmente, a redutibilidade de P' para P implica que P' pode ser considerado como um caso especial de P; o fato de P ser NP-completo, indica que P é em algum sentido, o problema mais difícil em $\mathcal{NP}$.

Gonzalez and Sahni (1978) fazem um estudo da complexidade e aproximação para o problema de scheduling tipo flowshop e jobshop. Definem o flowshop e jobshop como sendo conjuntos ordenados de $m \geq 1$ processadores, $(P_1, P_2, \ldots, P_m)$ e define também um schedule de *tempo final ótimo* - OFT (que têm o menor tempo final entre todos os factíveis) e o schedule com *fluxo de tempo médio ótimo* - OMFT (o que têm o menor fluxo de tempo médio entre todos os schedules factíveis). Considera também para ambos modelos, flowshop e jobshop, schedules com prioridades e sem prioridades. Prova que o problema de obter schedules OFT e OMFT para flowshop e jobshop é NP-completo para o caso de schedules sem prioridades, e estende estes resultados para o caso de schedules com prioridades.

Stecke(1983) escreve que gerenciar a produção em um Sistema Flexivel de Manufatura é mais difícil do que em linhas de produção ou em jobshop porque cada máquina é capaz de executar várias operações diferentes. Assim, o sistema pode fabricar vários tipos de peças em forma simultânea e cada peça pode ter rotas alternativas do sistema. Estas aptidões adicionais e opções de planejamento aumentam o número de variáveis de decisão.

Bensana at al. (1988) apresentam um sistema baseado em conhecimentos múltiplos para scheduling de jobshop industrial, OPAL. O sistema OPAL usa várias ferramentas e entidades conceituais para considerar diferentes classes de conhecimento envolvidos na formulação do problema, descrição de informações e métodos de solução. De uma maneira geral é um software software para scheduling de uma fábrica de tamanho médio ou pequena, dedicada à produção de partes. Porém, não trata problemas de alocação de operações nas máquinas. Sua arquitetura é formada

por três módulos principais monitorado por um supervisor o qual dirije o processo de busca como segue:

1. A base de dados contém a descrição da fábrica (estações de trabalho, tipo de partes, etc.), descrição do problema de scheduling (produção, exigências, datas de entrega, disponibilidade de máquinas, etc.) e o estado atual do projeto de scheduling.

2. Um módulo de análise baseado nas restrições (CBA) calcula as consequências das restrições de tempo no sequenciamento de operações. Pode gerar novas relações de precedência entre operações e propagar tais relações.

3. Um módulo de suporte a decisão (DS) da informação sobre o sequenciamento de operações, baseado na prática ou conhecimento heurístico.

4. Um supervisor controla o diálogo entre CBA e DS e constrói o schedule passo a passo de acordo com as decisões tomadas por esses módulos.

Chryssolouris at al. (1994) afirmam que em manufatura existem uma grande quantidade de problemas de tomada de decisões nas áreas de desenho de produto, controle de processos de manufatura e controle da produção e que a tomada de decisão em manufatura é caracterizada pelo tempo necessário para que uma decisão seja tomada, o número de decisões a serem tomadas em um certo período de tempo e o impacto que a decisão terá sobre o sistema de manufatura. Baseado nos valores destes atributos, as decisões de manufatura podem ser classificados como estratégicas, operacionais ou decisões pormenorizadas, que estão relacionadas de uma maneira hierárquica. No nível estratégico, o tempo disponível para a tomada de decisões é usualmente atingido e as decisões têm um grande impacto na organização da manufatura. Um exemplo desta classe é a decisão para construir uma nova planta de manufatura.

Vários procedimentos têm sido desenvolvidos para abordar o problema de programação horária em tempo real. Uma aproximação que têm sido extensamente

pesquisada é o uso de regras de despacho (Panwalker and Iskander (1977), Stecke and Solberg (1981), Blackstone at al. (1982)). Daniels and Mazzola (1994) tratam o problema da melhoria na eficiencia em manufatura que pode ser atingida ampliando o alcance do sequenciamento da produção incluindo o sequenciamento do trabalho e a coordenação dos recursos de entrada necessários para executar o trabalho. Admitindo que alguns recursos são inerentemente flexíveis e assim podem ser realocados dinâmicamente aos centros de trabalho quando necessário, e que o tempo de processamento dos jobs são comumente função das quantidades de recursos dedicados a operações específicas, formulam o problema de sequenciamento de recursos flexíveis com o objetivo de determinar simultaneamente a sequência de jobs, uma política de alocação de recursos e o tempo de início das operações que otimize o sistema de execução.

De Matta and Gugnard (1994), tratam o problema de sequenciamento da produção com capacidade orientada que pode ser descrito como sendo a alocação de produtos em vários níveis e com linhas de produção capacitadas sobre um número finito de períodos. Usam técnicas Lagrangeanas com soluções que podem gerar schedules de produção factíveis, realocando sistemàticamente linhas de produtos cuja produção exede a demanda, para produtos com demanda insatisfeita.

Koulamas (1994) explora o problema de atraso e propõe uma base unificada, resumindo a literatura relacionada com uma máquina, máquinas paralelas, cenários flowshop e jobshop. Define o problema DES como sendo o problema de sequenciamento determinístico cujos principais elementos são: a configuração de máquinas, as características dos jobs e a função ojetivo. Define o atraso de um job como sendo $T_i = \max\{0, c_i - d_i\}$ onde d_i é a data de entrega do job e c_i o tempo de conclusão do job. De acordo com este critério, não existe proveito completando os jobs mais cedo e a penalização é proporcional à demora. O objetivo é minimizar a penalização de todos os jobs $T_1 + T_2 + \ldots + T_n$.

Bilge and Ulusoy (1995), exploram a interação entre o sequenciamento de máquinas e o sequenciamento do sistema de manuseio de materiais em um FMS.

A transferência do material entre as máquinas é executada por veiculos guiado automaticamente (AGVs), e a formulação do problema consiste de um conjunto de restrições de um subproblema de sequenciamento de máquinas e um subproblema para o sequenciamento do veículo o qual interage através de um conjunto de restrições de janelas de tempo para o manuseio de materiais.

Wang and Liu (1995) desenvolveram uma aproximação usando técnicas de pesquisa operacional aliadas às técnicas de inteligência artificial, para o problema de sequenciamento dinâmico chamado de SAOSS (System-Attribute-Oriented knowledge-based Scheduling-System), com capacidade de aprendizagem indutiva para gerar sequenciamento inteligente com respeito às mudanças no sistema em tempo real.

Em Células Flexíveis de Manufatura, os equipamentos de manuseio de materiais influenciam na flexibilidade da célula, pois podem existir várias máquinas precisando os serviços do equipamento produzindo às vezes, conflitos nos quais uma máquina terá que esperar enquanto a outra é servida. A espera por parte de uma ou mais máquinas pode afetar o fluxo do sistema. Logo, deve ser manejado apropriadamente para não afetar o desempenho total da célula. Desta forma, é importante levar em consideração os movimentos dos equipamentos de manuseio das peças nos modelos matemáticos de programação horária.

Dentro deste contexto, consideramos células constituídas de algumas máquinas e um robô que transporta as peças de uma máquina para outra. Propomos heurísticas baseadas na lógica nebulosa e em programação linear nebulosa para a programação horária de peças e robô numa célula flexível de manufatura (Reumay, and Yamakami (1995)), dentro da tentativa de, simultaneamente, levar em consideração aspectos de incertezas nos tempos de processamento das peças pelas máquinas e nos tempos de transporte das mesmas pelo robô. Este tipo de enfoque, que consideramos extremamente adequado uma vez que estas incertezas existem nos processos reais, não foi encontrado na literatura revisada e, portanto, acreditamos ser uma importante contribuição nestas áreas de pesquisa.

1.2 Programação horária de um Flow Shop.

Em um flowshop, podemos ter n jobs e cada um com m tarefas $T_{1i}, T_{2i}, \ldots, T_{mi}$ $1 \leq i \leq n$ para serem executadas seguindo a mesma ordem nos processadores(máquinas).

A tarefa T_{ji} é executada sobre o processador P_j, $j \in \mathbb{N}_m$. Aqui $\mathbb{N}_m = \{1, 2, \ldots, m\}$. O tempo necessário para completar a tarefa T_{ji} é t_{ji}. Uma programação horária para n jobs é uma designação de tarefas em intervalos de tempo sobre os processadores. A tarefa T_{ji} deve ser designada ao processador P_j. Nenhum processador pode ter mais de uma tarefa designada para o mesmo intervalo de tempo. Além do mais, para qualquer job i o processamento da tarefa T_{ji}, $j > 1$ não pode ser iniciado até que a tarefa $T_{j-1,i}$ tenha sido completada.

1.3 Programação horária de um Jobshop .

O problema clássico de scheduling de um jobshop difiere do problema de scheduling de um flowshop em que o fluxo de trabalho não é unidirecional.

Os n jobs para serem programados precisam completar várias (m), tarefas. O tempo de execução da j-ésima tarefa para o job i é $t_{k,j,i}$. A tarefa j, $j \in \mathbb{N}_m$ é para ser executada no processador P_k. As tarefas para qualquer job i, $i \in \mathbb{N}_n$ são levadas a cabo numa ordem qualquer. A tarefa j não pode começar até que a tarefa $(j-1)$ (se $j > 1$) tenha sido completada. É possível que um job tenha várias tarefas para serem executadas no mesmo processador. Em uma programação horária sem prioridades, uma tarefa é processada de começo até o fim sem interrupções. O problema de obter o tempo final mínimo de uma programação horária com prioridades ou sem prioridades é NP-completo mesmo quando $m = 2$ (Horowitz (1978)).

Neste trabalho consideramos o problema de programação horária de peças

de um jobshop com rota tecnológica flexível.

No caso de processadores idênticos existe a seguinte regra que gera schedules muito perto do schedule ótimo com tempo final $\sum t_i/m$.

Definição. Uma programação horária LPT é aquela que resulta de um algoritmo onde, sempre que uma máquina fica livre, associa-se a esta máquina a tarefa cujo tempo é o maior daquelas tarefas ainda não alocadas. Os empates em geral são resolvidos de maneira arbitrária (Horowitz (1978)).

Exemplo. Seja $m = 3, n = 6$ e $(t_1, t_2, t_3, t_4, t_5, t_6) = (8, 7, 6, 5, 4, 3)$. Em uma programação horária LPT as tarefas 1, 2, 3 são designadas aos processadores 1, 2, 3, respectivamente. As tarefas 4, 5 e 6 são designadas aos processadores 3, 2, 1 respectivamente. Se t_{ji} é o tempo necessário para completar a terefa t_i no processador P_j a programação horária LPT é:

$$0 \leq t_{11} \leq 8, \quad 8 \leq t_{61} \leq 11$$
$$0 \leq t_{22} \leq 7, \quad 7 \leq t_{52} \leq 11$$
$$0 \leq t_{33} \leq 6, \quad 6 \leq t_{43} \leq 11$$

e o tempo final é 11. Como $\sum t_i/3 = 11$, a programação horária é ótima .

Teorema 1. (Graham) Seja $F^*(I)$ o tempo final de uma programação horária ótima de m processadores idênticos para o modelo I do problema de scheduling de tarefas. Seja $\hat{F}(I)$ o tempo final de uma programação horária LPT para o mesmo modelo. Então

$$|F^*(I) - \hat{F}(I)|/F^*(I) \leq \frac{1}{3} - \frac{1}{3m}$$

(Prova em Horowitz(1978).)

1.4 Conjuntos nebulosos: definições

Dado U o universo do discurso que pode ser discreto ou contínuo, apresentamos as seguintes definições básicas:

1. Um conjunto nebuloso F é caraterizado pela sua função de pertinência denotada por μ que associa a cada ponto em U um número real no intervalo [0, 1]

$$\mu_F : U \rightarrow [0,1]$$

onde $\mu_F(x)$ representa o grau de compatibilidade de x com F (Klir and Folger (1988), Yager at al. (1987), Zadeh(1965)). Zadeh (1965) representa um conjunto nebuloso F em U como um conjunto de pares ordenados $F = \{(x, \mu_F(x))|x \in U\}$ denotado por $F = \{\mu_F(x)/x|x \in U\}$. Quando U é discreto, o conjunto nebuloso F é representado como:

$$F = \mu_F(x_1)/x_1 + \ldots + \mu_F(x_n)/x_n$$

e quando U é contínuo F pode ser expresso na forma:

$$F = \int_U \mu_F(x)/x;$$

2. Suporte de $F = \{x \in U/\mu_F(x) > 0\}$;

3. O elemento $x \in U$ para o qual $\mu_A(x) = 0.5$ é denominado ponto de corte;

4. Um conjunto nebuloso unitário é um conjunto cujo suporte é um único ponto em U;

5. Se A e B são subconjuntos nebulosos, $A \subset B$ se e somente se $\mu_A(x) \leq \mu_B(x), x \in U$;

6. A união de dois subconjuntos nebulosos A e B é definida pela sua função de pertinência $\mu_{A \cup B}(x) = \vee\{\mu_A(x), \mu_B(x)\}$, $\vee$ denota o máximo;

7. $A \cap B$ é definida por $\mu_{A \cap B}(x) = \wedge\{\mu_A(x), \mu_B(x)\}$, $\wedge$ denota o mínimo;

8. O complemento $\overline{A}$ de A é definido ponto a ponto para todo $x \in U$ por $\mu_{\overline{A}}(x) = 1 - \mu_A(x)$;

9. Se $A_1, \ldots, A_n$ são subconjuntos nebulosos de $U_1, \ldots, U_n$ respectivamente, o produto cartesiano de $A_1, \ldots, A_n$ é um conjunto nebuloso no espaço n-dimensional $U_1 \times \ldots \times U_n$ com função de pertinência expressa por:

$$\mu_{A_1 \times \ldots \times A_n}(x_1, x_2, \ldots, x_n) = \wedge\{\mu_{A_1}(x_1), \ldots, \mu_{A_n}(x_n)\},$$

10. Uma relação nebulosa é um conjunto nebuloso definido no produto Cartesiano de conjuntos $X_1, X_2, \ldots, X_n$ onde as n-tuplas $(x_1, x_2, \ldots, x_n)$ podem ter vários graus de pertinência na relação, expressos por um número real no intervalo $[0, 1]$.

Exemplo. Dada a relação $R(X_1, X_2, \ldots, X_n)$, seja $[R \downarrow Y]$ a projeção de R que descarta todas as variáveis em X exceto as do conjunto $Y = \{X_j / j \in J \subset \mathbb{N}_n\}$. Então, $[R \downarrow Y]$ é uma relação nebulosa cuja função de pertinência é definida no produto cartesiano de conjuntos em Y pela equação:

$$\mu_{[R \downarrow Y]}(y) = \bigvee_{x \succ y} \mu_R(x)$$

onde μ_R é a função de pertinência da relação R e $y \prec x$ denota que y é subsequência de x. Consideremos por exemplo os conjuntos $X_1 = \{u, v\}$, $X_2 = \{a, b\}$, $X_3 = \{\alpha, \beta\}$ e a relação nebulosa ternaria $R(X_1, X_2, X_3) = 0.7/(u, a, \alpha) + 0.9/(u, b, \alpha) + 1/(v, a, \beta) + 0.4/(v, a, \alpha) + 0.8/(v, b, \beta)$. Seja $R_{i,j} = [R \downarrow \{x_i, x_j\}], R_i = [R \downarrow X_i], i, j \in \mathbb{N}_3$.

Então: $R_{1,2} = 0.7/(u, a) + 0.9/(u, b) + 1/(v, a) + 0.8/(v, b)$
$R_{1,3} = 0.9/(u, \alpha) + 0.4/(v, \alpha) + 1/(v, \beta)$
$R_{2,3} = 0.7/(a, \alpha) + 1/(a, \beta) + 0.9/(b, \alpha) + 0.8/(b, \beta)$
$R_1 = 0.9/u + 1/v$
$R_2 = 1/a + 0.9/b$

$$R_3 = 0.9/\alpha + 1/\beta,$$

11. Composição sup $*$.

Sejam R uma relação nebulosa em $U \times V$, S uma relação em $V \times W$. A composição de R e S é uma relação nebulosa denotada por $S \circ R$ e definida por

$$S \circ R = \{[(u,w), \sup_v(\mu_R(u,v) * \mu_S(v,w))],\ u \in U, v \in V, w \in W\}$$

onde $*$ é um operador da classe das normas triangulares tais como, mínimo, produto algebrico, produto drástico ou produto limitado (Lee 1990).

Exemplo: Sejam $P(X,Y) = 0.7/(1,a) + 0.5/(1,b) + 1/(2,a) + 1/(3,b) + 0.4/(4,b) + 0.3/(4,c)$, $Q(Y,Z) = 0.6/(a,\alpha) + 0.8/(a,\beta) + 1/(b,\beta) + 0.9/(c,\beta)$ e

$R(X,Z) = P(X,Y) \circ Q(Y,Z)$, relações nebulosas. O operador mais comum de composição é o max-min. A operação de composição é definida por $\mu_{PoQ}(x,z) = \max_{y \in Y} \min(\mu_p(x,y), \mu_Q(y,z))\ \forall x \in X, \forall z \in Z$. Por exemplo:

$$\begin{aligned}
\mu_R(1,\alpha) &= \max_{y \in Y} \min(\mu_p(1,a), \mu_Q(a,\alpha)) = \max \min(0.7, 0.6) = 0.6 \\
\mu_R(1,\beta) &= \max_{y \in Y} \{\min(\mu_p(1,a), \mu_Q(a,\beta)), \min(\mu_p(1,\beta), \mu_Q(b,\beta))\} \\
&= \max(\min(0.7, 0.8), \min(0.5, 1)) \\
&= 0.7,
\end{aligned}$$

$R(X,Z)$ está representada pela função de pertinência especificada na Tabela 1.1.

Composição		$R = PoQ$
x	z	$\mu_R(x,z)$
1	α	0.6
1	β	0.7
2	α	0.6
2	β	0.8
3	β	1
4	β	0.4

Tabela 1.1 Composição de relações

12. Número nebuloso. Pode ser definido em todo conjunto totalmente ordenado como $\mathbb{R}, \mathbb{R}^+, \mathbb{Z}$ ou $\mathbb{N}$.

Seja A um subconjunto nebuloso de $\mathbb{R}$. A é convexo se e somente se todo subconjunto ordinário $A_\alpha = \{x / \mu_A(x) \geq \alpha\}, \alpha \in [0,1]$ é convexo, isto é, se e somente se

$$\forall x_1, x_2 \in \mathbb{R}, \mu_A(\lambda x_1 + (1-\lambda)x_2) \geq (\mu_A(x_1) \wedge \mu_A(x_2)), \forall \lambda \in [0,1].$$

A é normal se e somente se $\forall x, \ \vee \mu_A(x) = 1$. Esse máximo pode ou não ser único.

Um número nebuloso em $\mathbb{R}$ é um subconjunto nebuloso de $\mathbb{R}$ que é convexo e normal.

13. Variáveis linguísticas. Uma variável linguística é caracterizada por uma quintupla (x, T, U, G, S) onde x é o nome da variável; $T(x)$ é o conjunto de termos de x, isto é, o conjunto de valores linguísticos de x sendo cada valor um número nebuloso definido em U; G uma regra sintática para gerar os nomes dos valores de x; S uma regra semântica para associar a cada valor o seu significado.

Exemplo. x = temperatura
$T(x) = \{$baixa, média, alta$\}$
$U = [0, 100]$

O conhecimento da língua portuguesa e dos fenômenos físicos permite selecionar o conjunto de termos que caracterizam uma grandeza (regra sintática) e determinar o significado de cada um dos termos (regra semântica).

Obs. (1) Cada valor linguístico em $T(x)$ é um conjunto nebuloso definido em U.

(2) A temperatura baixa pode ser interpretada como temperaturas abaixo de 40°C, média em torno a 60°C e alta acima de 100°C.

(3) A representação gráfica da variável linguística x = temperatura com as funções de pertinência caracterizam seus valores (Figura 1).

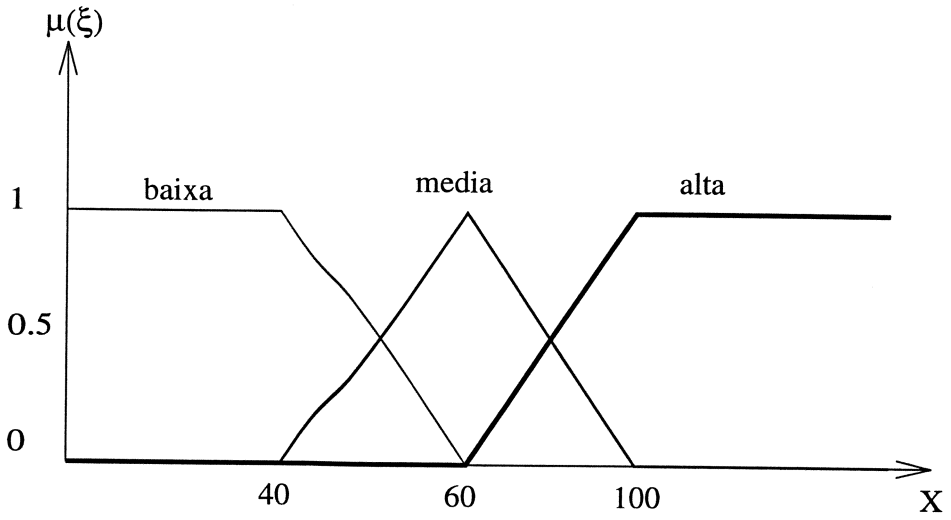


Figura 1: **Exemplos de termos da variável linguística temperatura .**

1.5 Lógica nebulosa e raciocínio aproximado

Vários autores, tais como Baldwing (1979), Baldwing and Guild (1980), Dubois at al. (1991), Gaines(1977), Giles(1977), Lee(1990), Yager(1980), etc., tratam da questão de lógica nebulosa e raciocínio aproximado da seguinte forma. Uma proposição nebulosa consiste na atribuição de um valor linguístico a uma variável linguística. Com o uso das proposições nebulosas pode-se construir uma estrutura básica de representação do conhecimento composta de regras nebulosas. Uma regra nebulosa é uma relação nebulosa expressa por:

Se x é A_i E y é B_i Então z é C_i

Esta regra é implementada por uma relação nebulosa R_i com expressão:

$$\mu_{R_i} \triangleq \mu_{(A_i \text{ e } B_i \rightarrow C_i)}(u, v, w)$$
$$\mu_{R_i} = (\mu_{A_i}(u) \text{ e } \mu_{B_i}(v)) \rightarrow C_i(w)$$

onde: a proposição A_i e B_i é um conjunto nebuloso $A_i \times B_i$ em $U \times V$; a implicação nebulosa $(A_i \text{ e } B_i) \rightarrow C_i$ é uma relação nebulosa R_i em $U \times V \times W$.

Exemplo. Modus ponens generalizado.

V é A'

Se V é A então w é B

Então w é B'

onde A, $A' \subsetneq X$, $B \subsetneq Y$, $B' = A' o D_{B|A}$ isto é, (aqui $\subsetneq$ denota subconjunto nebuloso.)

$$B'(y) = \max\{A'(x) \wedge D_{B|A}(x, y)\}$$

e se v é A então w é B induz uma relação nebulosa sobre $X \times Y$, por exemplo como aquela definida por Zadeh(1987):

$$D_{B|A}(x, y) = \min\{1, 1 - \mu_A(x) + \mu_B(y)\}.$$

Definição: Regra de composição.

Seja R uma relação nebulosa em $U \times V$, $A' \subsetneq U$. O conjunto $B' \subsetneq V$ induzido por A' é definido como $B' = A' o R$, onde $A' o R$ é a composição sup $*$, Lee(1990).

Uma *base de regras nebulosas* consiste em uma coleção de regras nebulosas *se - então*, da seguinte forma:

$$R^{(\ell)} : \text{Se } x_1 \text{ é } A_1^\ell \text{ e } x_2 \text{ é } A^\ell \text{ e } \ldots \text{ e } x_n \text{ é } A_n^\ell, \text{ então } y \text{ é } B^\ell \qquad (1.1)$$

onde $A_i^\ell \subsetneq U_i \subset R$ e $B^\ell \subsetneq V \subset R$, $\overline{x} \in U_1 \times \ldots \times U_n$, $y \in V$ são variáveis linguísticas. Se M é o número das regras nebulosas *se - então* da forma (1.1) na base de regras, então $\ell = 1, 2, \ldots, M$.

Existem duas abordagens empíricas para se formular as regras nebulosas que dependem basicamente da origem do conhecimento. O mais comum formaliza o conhecimento do especialista. Uma outra abordagem baseia-se na elaboração de uma série de entrevistas com especialistas ou operadores, usando questionários preparados cuidadosamente. Assim é possível montar um protótipo de base de regras para um domínio de aplicação. Não existe um procedimento padrão para se decidir pelo número de regras considerado ótimo, pois diversos fatores influenciam o processo

de definição das regras, tais como: o desempenho do sistema, a eficiência computacional, a experiência e o conhecimento de especialistas e a escolha das variáveis linguísticas.

Para obter F_i^ℓ e G^ℓ há também duas maneiras principais, dependendo da origem das regras. Se as regras são geradas via especialistas, então as funções de pertinência de F_i^ℓ e G^ℓ seriam especificadas pelos própios especialistas porque estas funções são parte integrada do conhecimento . Se as regras são determinadas por dados numéricos então $\mu_{F_i^\ell}$ e μ_{G^ℓ} são em geral assumidas Gaussianas, triangular ou trapezeidal, por definição.

Função de Fuzzificação. Uma Função de Fuzzificação é uma função

$$\alpha : U \to U, \quad \alpha(x) = A', \quad x \in U, \quad A' \tilde{\subseteq} U$$

Relações possíveis para esta função:

(1) Função de Fuzzificação unitario: A' é um conjunto unitario nebuloso, isto é,
$\mu_{A'}(x') = \begin{cases} 1 & \text{se} \quad x' = x \\ 0 & \text{se} \quad x' \neq x \end{cases}$

(2) Função de Fuzzificação não unitario: $\mu_{A'}(x) = 1$ e $\mu_{A'}(x')$ decresce quando x' se afasta de x, por exemplo,
$\mu_{A'}(x') = \exp\left\{-\dfrac{(x'-x)^T(x-x)}{\sigma^2}\right\}$, onde σ^2 é um parâmetro que caracteriza a forma de $\mu_{A'}(x')$.

Função de Defuzzificação. Uma Função de Defuzzificação é uma função
$\gamma : V \to V, \quad \gamma(A) = y, \quad A \tilde{\subseteq} V, \quad y \in V.$
Há três relações possíveis para esta função :

(1) Função de Defuzzificação máxima, definida por:
$y = \arg\sup_{y \in V}\{\mu_{B'}(y)\}, \quad \text{onde} \quad \mu_{B'}(y) = \mu_{B^1}(y) \overset{\circ}{+} \ldots \overset{\circ}{+} \mu_{B^M}(y).$
e onde $\overset{\circ}{+}$ é uma T co-norma (Klir (1992)).

(2) Função de Defuzzificação média dos centros, definido por
$y = \frac{\sum_{\ell=1}^{M} \bar{y}^{\ell} \cdot \mu_{B^{\ell}}(\bar{y}^{\ell})}{\sum_{\ell=1}^{M} \mu_{B^{\ell}}(\bar{y}^{\ell})}$, onde $\bar{y}^{\ell}$ é o centro do conjunto nebuloso G^{ℓ} e $\mu_{B^{\ell}}(y) = \sup_{x \in U}\{[\mu_{(A \to B)}(x, y) \star A'(x)]\}$
$A = F_1^{\ell} \times \ldots \times F_n^{\ell}$ $B = G^{\ell}$, sendo $\star$ uma T-norma.

(3) Função de Defuzzificação média dos centros modificado, definido por
$y = \frac{\sum_{\ell=1}^{M} \bar{y}^{\ell} \cdot \mu_{B^{\ell}}(\bar{y}^{\ell})/\delta^{\ell}}{\sum_{\ell=1}^{M} \mu_{B^{\ell}}(\bar{y}^{\ell})/\delta^{\ell}}$, onde δ^{ℓ} é um parâmetro que caracteriza a forma de $\mu_{G^{\ell}}(y)$ de modo que a forma mais estreita de $\mu_{G^{\ell}}(y)$ é δ^{ℓ}; por exemplo, se $\mu_{G^{\ell}}(y) = \exp\left(-\left(\frac{y - \bar{y}^{\ell}}{\delta^{\ell}}\right)^2\right)$, então δ^{ℓ} é esse parâmetro.

1.6 Procedimento de inferência

$$
\begin{array}{rl}
\text{Condição :} & x \text{ é } A' \text{ e } y \text{ é } B' \\
\text{Regras :} & R_1 : \text{ se } x \text{ é } A_1 \text{ e } y \text{ é } B_1 \text{ então } z \text{ é } C_1 \\
 & \vdots \\
 & \underline{R_n : \text{ se } x \text{ é } A_n \text{ e } y \text{ é } B_n \text{ então } z \text{ é } C_n} \\
\text{Conclusão :} & z \text{ é } C'
\end{array}
$$

onde x, y são variáveis linguísticas de condição e z é uma variável linguística conclusão; A_i, B_i, C_i são valores linguísticos das variáveis linguísticas x, y, z nos universos $U, V, W, i = 1, \ldots, n$.

Exemplo. Para ilustrar o procedimento de inferência, suponhamos um sistema nebuloso composto das duas regras:

Regra 1: Se x é A_1 e y é B_1, então z é C_1.

Regra 2: Se x é A_2 e y é B_2, então z é C_2.

Consideremos os valores de entrada $A' = u_0$ e $B' = v_0$. Os valores númericos u_0 e v_0 são considerados precisos, logo devem ser convertidos em conjuntos nebulosos unitários.

Aqui $(\alpha_1 = x$ é A_1 e y é $B_1)$ é expresso por

$\alpha_1 = \mu_{A_1}(x_0) \wedge \mu_{B_1}(y_0)$

$(\alpha_2 = x$ é A_2 e y é $B_2)$ é expresso por

$\alpha_2 = \mu_{A_2}(x_0) \wedge \mu_{B_2}(y_0)$

onde $\mu_{A_1}(x_0), \mu_{A_2}(x_0), \mu_{B_1}(y_0), \mu_{B_2}(y_0)$ indicam o grau de pertinência dos dados de entrada no conjunto das condições definidas na Regra 1 e Regra 2 respectivamente. A conclusão é expressa por $\mu_c(z) = \mu_{c_1'}(z) \vee \mu_{c_2'}(z)$

$$\mu_c(z) = \{\alpha_1 \wedge \mu_{c_1}(z)\} \vee \{\alpha_2 \wedge \mu_{c_2}(z)\}$$

O processo de raciocínio nebuloso deste exemplo é ilustrado na Figura 2 (Lee (1990)).

1.7 Programação linear nebulosa - FLP.

Zadeh (1965) sugere o conceito de conjuntos nebulosos como uma forma de modelagem das relações e fenômenos incertos os quais não são de natureza estocástica. Bellmam and Zadeh (1970) sugerem modelos para tomada de decisões em ambientes nebulosos. Sobre a base da teoria de Zadeh vários pesquisadores têm desenvolvido e aplicado modelos de programação linear nebulosos (Zimmermann(1976), Zimmermann (1984), Tanaka and Asai (1984-a), Tanaka and Asai (1984-b), Werners(1987), Wiedely and Zimmermann (1987), Chanas(1983), Lai and Hwang (1992), Reumay, and Yamakami (1995)), etc..

Os modelos FLP podem ser classificados como segue:

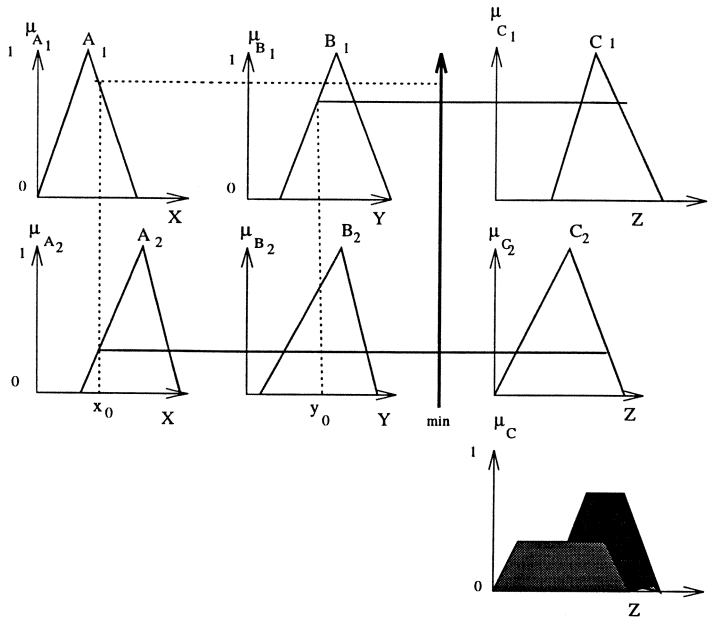


Figura 2: **Gráfico do procedimento de Inferenência do exemplo**

(i) Problemas com restrições nebulosas, permite-se pequenas violações na satisfação das restrições.

(ii) Problemas com função objetivo nebulosa, onde a informação sobre os coeficientes da função objetivo é imprecisa. Esses coeficientes podem ser definidos por números nebulosos.

(iii) Problemas com coeficientes nebulosos, onde podem-se usar elementos de $F(\mathbb{R})$ em ambos membros das restrições, onde $F(\mathbb{R})$ denota o conjunto dos número nebulosos.

É possível, com certeza, a combinação natural das três situações.

Uma classe particular de tomada de decisão pode ser vista como um problema de otimização sob restrições, definido como segue.

Dados:

- um conjunto de variáveis de decisão,

- um conjunto de restrições sob estas variáveis de decisão,
- uma função objetivo a qual ordena as alternativas,

encontrar a solução ótima .

Em outras palavras, o problema pode ser separado nas variáveis de decisão, as restrições e as metas(ou objetivos). A noção de incertezas pode ser introduzida em todos estes elementos básicos. A teoria de Programação Matemática Nebulosa - FMP restringe-se às incertezas das restrições e metas, pois elas constituem classes de alternativas cujos contornos não estão definidos claramente. As variáveis de decisão são consideradas determinísticas. De acordo a teoria de Bellman and Zadeh(1970) a função objetivo nebulosa assim como as restrições nebulosas são caracterizadas por funções de pertinência. O objetivo é satisfazer metas e restrições; portanto uma decisão nebulosa é consdierada como a intersecção das restrições nebulosas e as metas nebulosas. Uma característica importante da teoria é a simetria entre metas e restrições. Os dois conceitos são essencialmente similares, o que torna mais simples a relação dos conceitos de decisão nebulosa com as metas e restrições. A base geral da teoria de programação matemática nebulosa pode ser proposta como segue:

1 - Seja X um conjunto de alternativas possíveis. Uma meta nebulosa G e uma restrição nebuloa C são subconjuntos nebulosos de X caracterizados pelas suas funções de pertinência
$\mu_G : X \rightarrow [0,1], \mu_C : X \rightarrow [0,1]$, respectivamente.

A decisão nebulosa D resultante da meta nebulosa e restrição nebulosa é a interseção de ambos conjuntos nebulosos, $D = G \cap C$, caracterizado pela sua função de pertinência

$$\mu_D(x) = \min\{\mu_G(x), \mu_C(x)\}\, x \in X.$$

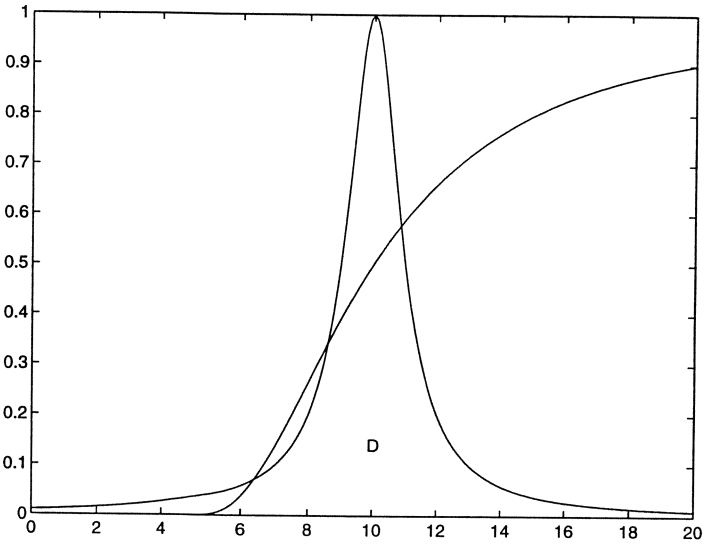


Figura 3: **Gráfico da decisão nebulosa D como intersecção da meta G e a restrição C**

Geralmente a decisão resultante de n metas nebulosas $G_1, \ldots, G_n$ e m restrições nebulosas $C_1, \ldots, C_n$ é definida por:

$$\mu_D(x) = \min\{\mu_{G_i}(x), \mu_{C_j}(x)\}\, x \in X, i = 1, \ldots, n,\ j = 1, \ldots, m.$$

Por exemplo suponhamos que uma meta G e uma restrição C são expressas por:

G: x deve ser muito maior do que 5.

C: x deve estar perto de 10.

$$\mu_G(x) = \begin{cases} 1 - (1 + (0.2(x-5))^2)^{-1} & \text{se} \quad x > 5 \\ 0 & \text{se} \quad x \leq 5 \end{cases}$$

$$\mu_C(x) = (1 + (x-10)^2)^{-1}$$

A decisão nebulosa D é a intersecção, $D = G \cap C$ (Figura 3).

2 - As metas e as restrições são conceitos matematicamente idênticos. Ainda que não se faz menção de uma função objetivo mas de uma meta nebulosa, é claro

que a função de pertinência, $\mu_G(x)$, serve ao mesmo propósito.

Com respeito a uma função objetivo limitada $f : X \rightarrow \mathbb{R}^+$ onde X pode ser $\mathbb{R}^n(f(x) \leq M)$, se C é uma restrição nebulosa em X com
$\mu_c : X \rightarrow [0,1]$, então tomando $\mu_G(x) = f(x)/M$ temos
$\mu_G : X \rightarrow [0,1]$ transformando a função objetivo f em uma meta nebulosa. O problema de programação matemática consiste em determinar o máximo da decisão nebulosa: $\sup_{x \in X} \mu_D(x) = \sup_{x \in X}\{\mu_C(x) \wedge \mu_G(x)\}$. Em Tanaka et al.(1974), e Negoita and Ralescu(1975) prova-se que sob certas condições este problema reduz-se ao problema de programação matemática convencional de encontrar o máximo:

$$\sup_{x \in X} \mu_D(x) = \sup_{x \in A} \mu_G(x)$$

onde $A = \{x \in X | \mu_c(x) \wedge \mu_G(x) \geq 0\}$

Com respeito a programação linear, a primeira extensão foi apresentada por Zimmermann(1976), Ele fuzzifica o problema:

$$\begin{aligned} \min\ & z = c^T x \\ \text{s.a.}\ & Ax \leq b \\ & x \geq 0 \end{aligned}$$

na versão nebulosa

$$\begin{aligned} cx &\lesssim z_0 \\ Ax &\lesssim b \\ x &\geq 0 \end{aligned}$$

onde A, b, c são conhecidos. O simbolo $\lesssim$ têm a interpretação "não substancialmente maior que" ou "essencialmente menor ou igual a" e z_0 tem o significado de nível de aspiração dado pelo planejador.

Este conceito de desigualdade nebulosa é operacionalizado tomando como função de pertinência, por exemplo, uma função linear f que aumente gradualmente

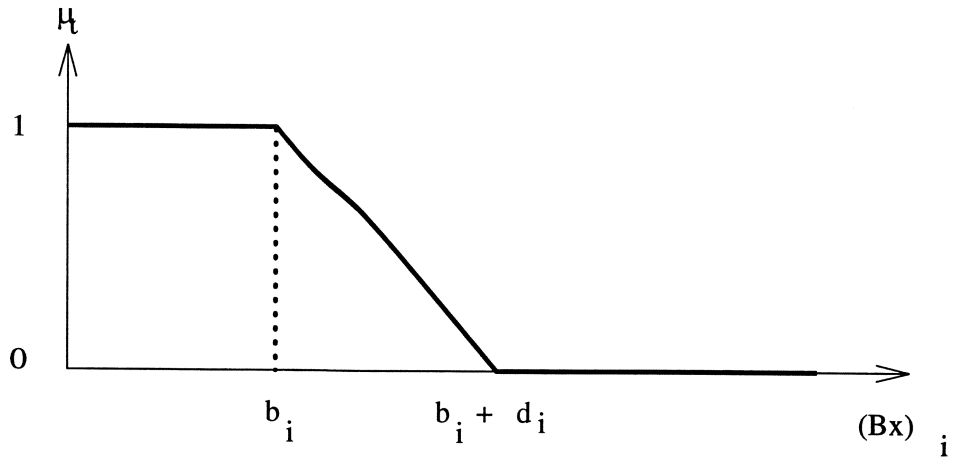


Figura 4: **Exemplo de Função de Pertinência** μ_i

de zero a um, entre seus valores extremo $cx \lesssim Z$ como dois conceitos matematicamente idênticos e definindo a decisão nebulosa como a interseção delas, temos a seguinte função de pertinência (Figura 4) para o conceito de desigualdade nebulosa:

$$f(Bx) = \min_i f_i((Bx)i)$$

$$\text{com } f_i((Bx)i) = \begin{cases} 1 & \text{se } B(x)_i \leq b_i \\ \dfrac{d_i + b_i - (Bx)_i}{d_i} & \text{se } b_i < (Bx)_i \leq b_i + d_i \\ 0 & \text{se } (Bx)_i > b_i + d_i \end{cases}$$

onde $B = \binom{c}{A}, b = \binom{b_0}{b} e f_i((Bx)_i)$ é a função da i-ésima linha do sistema Bx. As constantes d_i são as violações admissíveis das restrições. A função final $\min_i f_i((B_x)_i)$ é a decisão nebulosa do problema correspondendo ao conceito de interseção de metas e restrições.

Pode-se provar facilmente que o problema de encontrar a decisão máxima $\max_x \min_i f_i((Bx)_i)$ é equivalente ao problema LP:

$$\begin{aligned} &\max \lambda \\ &s.a. \qquad \lambda \leq b_i - (Bx)_i \quad , x \geq 0. \end{aligned}$$

Com efeito, substituindo $b_i'' = \frac{b_i}{d_i}, B_i' = \frac{B_i}{d_i}$ nas respectivas componentes de $f_i(Bx)$ temos:

$\frac{d_i + b_i' - (Bx)_i)}{d_i} = 1 + b_i'' - (B'x)_i$ onde podemos eliminar o "1" pois o problema não muda, então chegamos ao problema:

$$\max_x \min_i f_i((Bx)_i) = \max_x \min_i (b_i'' - (B'x)_i)$$

ou $\max_x \mu_D(x)$, onde $\mu_D(x)$ é a função de pertinência do conjunto de decisões D.

Por outro lado, se $\lambda = \mu_D(x)$ temos o problema equivalente[1]

$$\begin{aligned} & \max \lambda \\ s.a. \quad & \lambda \leq b_i'' - (B'x)_i \quad , i = 0, 1, \ldots, m, \\ & x \geq 0 \end{aligned}$$

O problema nebuloso $\begin{aligned} & \widetilde{\max} \quad c^T x \\ & \text{sa} \quad (Ax)_i \lesssim b_i, \ \forall i \\ & x \geq 0 \end{aligned}$

(Aqui $\widetilde{\max}$ têm o significado de incerteza na função objetivo),

é equivalente a:

$$\begin{aligned} & \widetilde{\max} \quad c^T x \\ & \text{s.a.} \quad (Ax)_i \leq b_i + \theta P_i \\ & x \geq 0 \end{aligned}$$

onde A, b, c, P são conhecidos. Quando o planejador não tem certeza de uma meta b_0 nem de uma tolerância máxima γ_0, para estabelecer a função de pertinência da função objetivo nebulosa, Werners (1987) propõe dois pontos extremos possíveis definidos como segue:

$$W^0 = \inf(\max_{x \in X} c^T x) = Z^*(\theta = 0)$$

[1] No sentido que a solução ótima para $\max \mu_D(x)$ é também ótima para $\max \lambda$

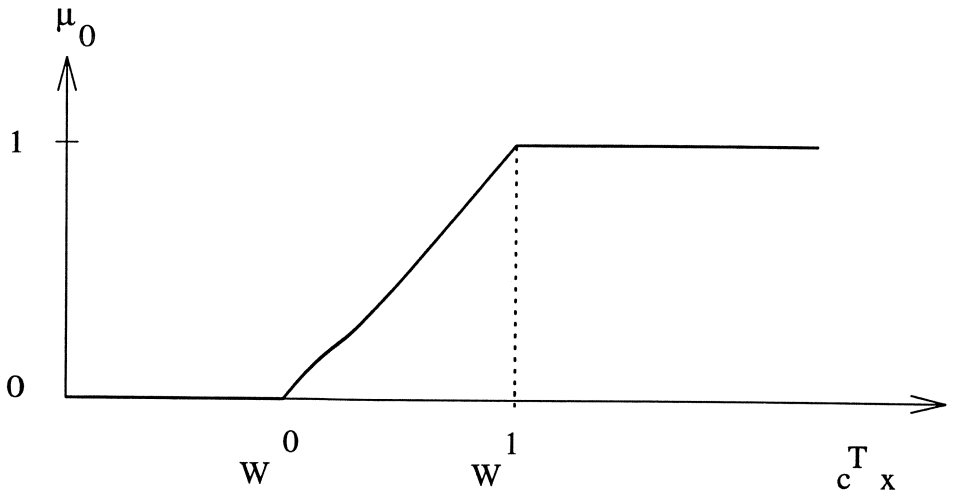


Figura 5: **Função de pertinência** μ_0

$$W^1 = \sup(\max_{x \in X} c^T x) = Z^*(\theta = 1)$$

onde $X = \{x | (Ax)_i \leq b_i + \theta P_i, \forall i, \theta \in [0,1], x \geq 0\}$.

A função de pertinência da função objetivo (Figura 5) fica:

$$\mu_0 = \begin{cases} 1 & \text{se } c^T x > Z^1 \\ \dfrac{c^T x - W^0}{W^1 - W^0} & \text{se } W^0 \leq c^T x \leq W^1 \\ 0 & \text{se } c^T x < W^0 \end{cases}$$

A função de pertinência das restrições é dada por;

$$\mu_i = \begin{cases} 1 & \text{se } (Ax)_i < b_i \\ \dfrac{b_i + P_i - (Ax)_i}{P_i} & \text{se } b_i \leq (Ax)_i \leq b_i + P_i \\ 0 & \text{se } (Ax)_i > b_i + P_i \end{cases}$$

O espaço D de decisões é definido pela sua função de pertinência, usando o operador min (Bellman and Zadeh(1970)):

$$\mu_D = \min(\mu_0, \mu_1, \ldots, \mu_m)$$

A solução ótima do problema de programação linear nebuloso é a decisão onde μ_D é máxima, portanto temos que o problema pode ser escrito como:

$$\begin{aligned} \max \quad & \mu_D \\ \text{s.a.} \quad & \mu_0 \geq \mu_D \\ & \mu_i \geq \mu_D \end{aligned}$$

$\mu_D, \mu_0, \mu_i \in [0,1] \forall_i, x \geq 0$

Se $\mu_D = 1 - \theta$, então $\mu_0 \geq \mu_D \Rightarrow c^T x \geq W^1 - \theta(W^1 - W^0)$

$$\mu_i \geq \mu_D \Rightarrow (Ax)_i \leq b_i + P_i\theta$$

substituindo, temos o problema equivalente:

$$\begin{aligned} \min \quad & \theta \\ \text{s.a.} \quad & c^T x \geq W^1 - \theta(W^1 - W^0) \\ & (Ax)_i \leq b_i + \theta P_i \quad , \forall_i \\ & \theta \in [0,1] \,, \; x \geq 0 \end{aligned}$$

onde A, b, c, P são conhecidos.

Se a meta b_0 é dada, como também a sua tolerância máxima γ_0, então o problema nebuloso é equivalente a determinar x tal que

$$C^T x \gtrsim b_0$$

$$(Ax)_i \lesssim b_i \,, \quad \forall i$$

$$x \geq 0$$

com função de pertinência das restrições definida por:

$$\mu_i = \begin{cases} 1 & \text{se} \quad (Ax)_i < b_i \\ \dfrac{P_i + b_i - (Ax)_i}{P_i} & \text{se} \quad b_i \leq (Ax)_i \leq b_i + P_i \\ 0 & \text{se} \quad (Ax)_i > b_i + P_i \end{cases}$$

com função de pertinência da função objetivo definida por:

$$\mu_0 = \begin{cases} 1 & \text{se} \quad c^T x > b_0 \\ \dfrac{c^T x - \gamma_0 - b_0}{\gamma_0} & \text{se} \quad b_0 - \gamma_0 \leq c^T x \leq b_0 \\ 0 & \text{se} \quad c^T x < b_0 - \gamma_0 \quad , \end{cases}$$

Repetindo o processo concluímos que o problema nebuloso é equivalente a:

$$\begin{aligned} &\min\ \theta \\ &\text{s.a.} \quad c^T x \geq b_0 - \theta\gamma_0 \\ &\qquad (Ax)_i \leq b_i + \theta P_i \ \ \forall_i \\ &\qquad \theta \in [0,1] \ \ x \geq 0 \end{aligned}$$

onde A, b, c, P, γ_0 são dados $\forall i$, Zimmermann(1984).

Pode-se apresentar as incertezas tanto na função objetivo, nas restrições, os coeficientes do vetor econômico, os coeficientes do vetor de recursos e os coeficientes da matriz tecnológica. O modelo geral proposto por Delgado et al.(1989) pode se escrever como segue:

$$\begin{aligned} \text{Max} \quad & Z = \textstyle\sum_{j=1}^{n} \tilde{c}_j x_j \\ \text{sa.} \quad & \textstyle\sum_{j=1}^{n} \tilde{a}_{ij} x_j \lesssim \tilde{b}_i \ \ i \in I\!N_m \\ & x_j \geq 0, \ \ j \in I\!N_n \end{aligned} \tag{1.2}$$

onde os elementos nebulosos considerados são definidos por:

1. Para cada restrição, existe $\mu_i \in F(I\!R)$, $\mu_i : I\!R \to [0,1]$, $i \in I\!N_m$, que define os numeros nebulosos $\tilde{b}_i$

2. Para cada $i \in I\!N_m, j \in I\!N_m$, existe $\mu_{ij} \in F(I\!R)$, $\mu_{ij} : I\!R \to [0,1]$ define os números nebulosos que compoem a matriz tecnológica.

3. Para cada $j \in I\!N_n$, existe $\mu_j \in F(I\!R)$, $\mu_j : I\!R \to [0,1]$ define os números nebulosos na função objetivo

4. Para cada restrição, existe $\mu^i \in F(F(I\!R))$, $\mu^i : F(I\!R) \to [0,1]$, fornece o *grau de satisfação* do número nebuloso $\sum_{j=1}^{n} \tilde{a}_{ij} x_j, i \in I\!N_m$, isto é, a adequação entre esse número nebuloso e o $\tilde{b}_i$ correspondente, por medio do número nebuloso $\tilde{t}_i$ que expressa a margem que o planejador tolera para a violação da i-ésima restrição em (1.2).

1.8 Programação linear nebulosa 0 - 1

Zimmermann and Pallatscheck (1984) afirmam que nos modelos de programação linear 0 - 1 a hipótese usual é que todos os parâmetros do modelo são conhecidos com precisão e podem ser expressos como números do sistema real $\mathbb{R}$. Porém, nas aplicações, esta hipótese pode ser uma aproximação grosseira do problema real. Por exemplo, em modelos de investimento, o planejador pode exigir que a soma do fluxo de caixa de todos os projetos "não exceda um certo limite" porque será extremamente difícil obter fundos para assegurar liquidez. Em problemas de produção pode-se desejar "uma boa utilização das capacidades" e em gestão de pessoal pode-se lutar por uma "atribuição razoável das pessoas às tarefas".

As idéias sobre modelos de pogramação linear nebulosos são extendidas a modelos de programação linear onde as variáveis de decisão devem ser 0 ou 1. Esses modelos são da forma:

$$\begin{aligned} &\text{Encontrar } x_j, j = 1 \ldots, r \quad , x_j \in \{0,1\} \\ &\text{tal que } \sum_{j=1}^{n} a_{ij} x_j \lesssim b_i \quad , i = 1, \ldots, n \end{aligned} \tag{1.3}$$

Aqui, o símbolo $\lesssim$ têm a interpretação "não substancialmente menor que". Para modelar esta desigualdade nebulosa cada linha i do problema (1.2) é interpretada como um conjunto nebuloso representado por sua função de pertinência $\mu_i(x)$. Utilizando conjuntos nebulosos normalizados os seguintes tipos de funções de pertinência podem ser definidas:

$$\mu_i(x) = \begin{cases} 1 \text{ se } u_{i1}x_1 + \ldots + a_{in}u_n \leq b_i \text{ é satisfeita} \\ \\ 0 \text{ se } u_{i1}x_1 + \ldots + a_{in}u_n \leq b_i \text{ é bastante violada} \end{cases} \tag{1.4}$$

"bastante violada" significa que a restrição é violada tal que se d_i é o "intervalo de

tolerância", então

$$a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n \geq b_i + d_i \tag{1.5}$$

Se admitimos que a função de pertinência têm crescimento linear no intervalo de tolerância, então a função de pertinência completa pode se definir por:

$$\mu_i(x) = \begin{cases} 1 & \text{se } \sum_{j=1}^{n} a_{ij}x_j \leq b_i \\ 1 - \dfrac{a_{ij}x_1 + \ldots + a_{in}x_n - b_i}{d_i} & \text{se } b_i \leq \sum_{j=1}^{n} a_{ij}x_j \leq b_i + d_i \\ 0 & \text{se } \sum_{j=1}^{n} a_{ij}x_j \geq b_i + d_i \end{cases} \tag{1.6}$$

Seguindo Bellmann and Zadeh (1972), uma decisão é a intersecção de todos os conjuntos nebulosos que representam ou funções objetivo ou restrições e que a função de pertinência da intersecção é calculada aplicando o operador min às funções de pertinência de todos os conjuntos nebulosos envolvidos. A função de pertinência do problema de *decisão* (1.3) é

$$\mu_D = \min\{\mu_1(x), \mu_2(x), \ldots, \mu_m(x)\}. \tag{1.7}$$

A solução ótima para o problema (1.3) será a solução com o maior grau de pertinência em (1.7). Agora, se $\lambda = \mu_D$, o modelo para determinar μ_D é:

$$\begin{aligned} &\text{Max } \lambda \\ &\text{s.a. } \lambda \leq 1 - \frac{a_{i1}x_1 + \ldots + a_{in}x_n - b_i}{d_i} \quad i = 1, \ldots, m \\ &\qquad x_j \in \{0,1\} \quad j = 1, \ldots, n \\ &\qquad \lambda \in [0,1] \end{aligned} \tag{1.8}$$

Se existe uma solução, x^* e λ^* para o problema (1.8), x^* maximizará (1.7). Se (1.8) não têm solução, não existem valores para x_j que indicará um valor positivo

para μ_D. Assim $\mu_D = \emptyset$. Neste sentido o problema (1.8) é equivalente ao problema (1.2) sob a hipótese (1.6).

1.9 Comentários.

Considerando que neste trabalho pretende-se abordar problemas de programação horária que levam em conta as incertezas nos tempos de processamento e de transporte, no início do capítulo foi apresentada uma breve revisão bibliográfica na área de sistemas de manufatura e em seguida foram descritas as principais de características dos sistemas de programação horária. Logo a seguir, foram apresentadas as idéias básicas sobre a teoria de con nebulosos, programação linear nebulosa e programação linear inteira nebulosa 0 - 1.

Os modelos apresentados levam em conta diferentes aspectos que são encontrados na resolução de problema: conhecimento de um objetivo incerto e de sua tolerância máxima e restrições nebulosas; desconhecimento destes atributos e restrições nebulosas; incertezas nos coeficientes da matriz tecnológica e nos elementos do vetor de recursos junto com restrições nebulosas. Estas teorias permitem tratar uma ampla variedade de situações em que as incertezas podem fazer parte dos problemas.

Finalmente, os seguintes pontos devem ser enfatizados:

- Para problemas combinatoriais que geralmente pertencem à classe NP-completo a combinação de métodos matemáticos com heurísticas é adequada.
- Os procedimentos como de planejamento, decisão e controle, via de regra apresentam inerentemente aspectos de incerteza, os quais, muitas vezes, são obrigatórios e devem ser considerados.
- A teoria de conjuntos nebulosos surge como uma ferramenta adequada em termos de tratamentos das questões de incerteza acima colocadas, tanto em

termos da utilização das lógicas como suas extensões no campo de programação matemática.

Capítulo 2
Formulação do problema

Neste capítulo, vamos apresentar a nossa proposta de modelo para o problema de programação horária de n peças em m máquinas em células flexíveis de manufatura, com o objetivo de minimizar o tempo total de processamento. Vamos considerar dois modelos baseados na regra LPT, com finalidade de aplicar técnicas baseadas na lógica fuzzy para resolvê-los. Apresentamos um modelo simples e um outro mais elaborado.

2.1 Programação horária em FMC.

A programação horária em FMC têm sido tradicionalmente vista como uma otimização restrita de uma função de tomada de decisões que pode ser formulada em termos de uma função objetivo explícita e restrições explícitas.

Quatro tipos de metas parecem ser predominantes:

- utilização eficiente de recursos,
- configuração para descrever datas de encerramento,
- medidas de custo dos sistemas, e
- respostas rápidas à demandas.

Duas classes de restrições são comumente encontradas:

- limite de capacidade e limite dos recursos disponíveis,
- regras de precedência sobre a ordem das operações.

Uma solução para o problema de programação horária é qualquer solução factível destes dois tipos de restrições que permite tomar decisões sobre alocação de recursos e sequenciamento de operações.
Uma programação horária de um jobshop é caracterizada por sequenciamento de operações tanto em série como em paralelo, entre um número limitado de recursos.

Ainda que o problema geral de programação horária de um jobshop com n operações e m-máquinas possa ser formulado precisamente, sob um certo conjunto de suposições sobre as operações, as máquinas e as restrições, como um problema de otimização de multicritério, situações práticas não podem ser gerenciadas eficientemente pois o algoritmo de geração de schedule usando programação inteira, programação dinâmica ou técnicas de busca branch-and-bound são de tipo enumerativos.

Métodos heurísticos usando ferramentas avançadas da inteligência artificial ou regras de despacho simples reduzem o espaço de busca de todas as programações horárias possíveis, e são capazes de produzir soluções sub-ótimas com menor exigência computacional. Relaxando a exigência de atingir soluções ótimas, o problema original $NP-$completo de programação horária chega a ser tratável, mas os algoritmos usados, todavia, não são capazes de reagir a um ambiente de mudanças rápidas de um jobshop aberto (rota variável das partes).

A produção de programação horária de tempo real para manufatura pode ser atingida usando programação horária dinâmica, provisto de um sistema computacional multi-tarefas controlado por um algoritmo adaptativo inteligente.

Neste desenho, as operações de manufatura são emuladas por tarefas computacionais executando concorrentemente e cooperativamente as restrições de manufatura, tais como datas de entrega, disponibilidade de recursos, precedencia de operações, etc., e são expressas diretamente dentro das tarefas que dirigem as operações. A função de adaptação pode ser fornecida por um sistema, baseado em conhecimento (especialista).

Dada uma seqüência de operações para cada máquina, existe somente uma solução de programação horária, na qual nenhum deslocamento local à esquerda no gráfico de Gantt pode ser feito e o conjunto destas soluções é chamado o conjunto de soluções semi-ativas e é equivalente ao conjunto de todas soluções que não contém tempo vazio supérfluo. O tipo de ajustamento no qual alguma operação é iniciada mais cedo sem atrasar qualquer outra operação é chamado deslocamento global à esquerda e o conjunto de todas soluções, nas quais nenhum deslocamento deste tipo pode ser feito, é chamado o conjunto de soluções de programação horária ativas.

Schedules ativos podem ser gerados diretamente por técnicas de enumeração explícita ou implícita. Os métodos de enumeração implícita são mais eficientes e usam três ferramentas matemáticas básicas: programação linear inteira, branch-and-bound, programação dinâmica (Merabet (1986)).
Na programação linear inteira, primeiro o problema é transformado em um programa linear e então resolvido usando algum algoritmo padrão. Os outros dois enfoques enfrentam o problema de otimização combinatorial diretamente. O enfoque branch-and-bound está baseado na idéia de enumeração das soluções factíveis inteligentemente. Elas determinam a solução ótima particionando sucessivamente o espaço de soluções. A programação dinâmica está baseada no princípio de otimalidade (qualquer subpolítica de uma política ótima é também ótima).

O método mais amplamente usado em problemas de schedules de jobshop é o " branch-and-bound ". Além disso, os procedimentos são agrupados em dois: aqueles que geram um schedule a partir de uma sequência de schedules parciais e aqueles que começam com um schedule completo (possivelmente infactível) e o modificam.

Os métodos pertencentes ao primeiro grupo são chamados métodos de geração de schedules e os segundos são chamados de métodos de modificação de schedules. Os métodos baseados nos três enfoques gerais mencionadas acima, produzem schedules ótimos. Estes métodos são chamados exatos e são conhecidos por serem penosamente lentos. Em muitas situações, não se encontra um schedule ótimo mas

sim, um schedule bom. Os métodos correspondentes são chamados aproximados e as soluções produzidas são também aproximadas. Além disso estes métodos podem ser classificados como garantidos (produzindo resultados dentro de alguma tolerância da solução ótima) ou heurísticos (produzindo uma solução baseada em alguma regra de decisão com sentido comum).

Métodos exatos e garantidos para geração de schedules, em geral consomem tempo, muitas vezes proibitivamente grandes até encontrar os resultados e não podem reagir em tempo real (reprogramação horária) a mudanças externas dos parâmetros. Estes schedules são comumente usados de uma maneira preditiva .

Devido ao problema da dimensionalidade de todas estas estratégias de enumeração, sómente casos simples de problemas de programação horária de jobshop são manuseados desta forma. Podem-se, também, desenvolver algoritmos eficientes usando regras de tomada de decisões.

2.2 Regras de despacho.

Do ponto de vista prático, o problema de sequenciamento trabalha com abordagens heurísticas que geram soluções sub-ótimas. Foi desenvolvido um grande número de heurísticas denominadas regras de despacho ou regras de prioridades. Coloca-se muitas vezes, a questão de se saber qual regra de despacho irá atingir um bom desempenho.

Para o sequenciamento de um jobshop PanWalkar and Iskander(1977) e Blackstone at al. (1982) sugerem ,entre outras, as seguintes regras:

LPT (*Longest Processing Times*): Seleciona a operação com o maior tempo de processamento.

SPT (*Shortest Processing Times*): As tarefas são executadas na ordem crescente de tempo de processamento.

R (*Random*): Seleciona qualquer tarefa com igual probabilidade. Esta regra

é usada como comparação com outras regras.

FCFS (*First Come First Served*): As tarefas são processadas na ordem de chegada à célula.

LWR (*Least Work Remaining*): Seleçao da operação associada com o *job* que têm a menor quantidade de trabalho restante a ser processado.

MWR (*Most Work Remaining*): Seleção da operação associada com o *job* que têm a maior quantidade de trabalho restante a ser processado.

MOPR (*Most Operations Remaining*): Seleção da operação associada com o *job* que têm o maior número de operações restantes a serem processadas.

LOPR (*Least Operations Remaining*): Considera o *job* com o menor número de operações restante .

Vamos apresentar agora, o nosso modelo de programação matemática para o problema de programação horária em FMC

O problema de programação horária em uma célula flexível de manufatura é modelado em geral, como um problema de programação linear inteiro mixto (Reumay and Yamakami (1995), Teixeira and Yamakami (1990)) e o objetivo é determinar a seqüência dos jobs nas máquinas e a seqüência de movimentos executados pelo robô para transportar as peças desde o buffer inicial às máquinas e destas ao buffer final, que minimize o makespan de um conjunto de jobs.

O modelo de célula considerado contém m máquinas processadoras e um robô para manipulação do material, um buffer de entrada e um buffer de saída (Figura 6).

Nesta configuração, o robô deve efetuar três tipos de tarefas:

1. mover uma peça desde a fila de entrada da célula até a fila de entrada de uma máquina.

2. descarregar jobs de uma máquina e levá-los à fila de entrada de uma outra

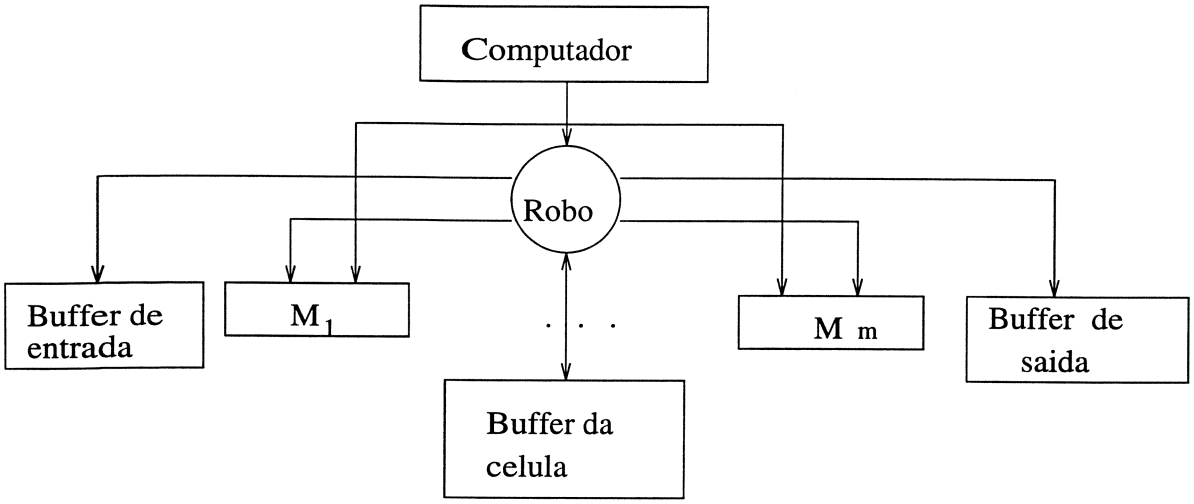


Figura 6: **Célula flexível de manufatura.**

máquina.

3. descarregar jobs de uma máquina e movê-los até o buffer de saída da célula.

Certamente, são possíveis outras formas de manipulação e armazenagem de jobs nas células. Neste trabalho, consideramos a situação onde o robô move-se em sequências de tempos independentes. Uma vez que os jobs são sequenciados na fila de entrada, eles fluem através da célula sem manobras pois as máquinas são carregadas automaticamente em ordem FCFS.

O tempo total(makespan) de término da programação horária de um conjunto de peças pode ser determinado como a soma dos tempos das tarefas do robô mais os tempos determinísticos no processamento das peças nas máquinas. O modelo é formulado como um PLI - programação linear inteira mista.

O objetivo é determinar a programação horária de peças e robô que minimize o makespan.

Este estudo é interessante pois fornece uma visão sobre como a programação horária do sistema de manuseio de materiais pode afetar as metas de produção de

uma célula flexível da manufatura.

2.3 Restrições do problema de programação horária

Notação:

$I = \{i | i = 1, 2, \ldots, n\}$ n peças (jobs),
$J = \{j | j = 1, 2, \ldots, m\}$ m máquinas,
t_j; tempo de transporte para o estágio j
y_{ij}; instante final do processamento do job i no estágio $j-1$, que coincide com o instante em que o job i está disponível para iniciar o transporte para o estágio j
x_{ij}; instante em que o job i está pronto para ser processado no estágio j,
$x_{ij} - t_j$; instante inicial da tarefa j do job i,
P_{ij}; tempo de processamento do job i no estágio j.

O conceito de *estágio* será largamente explorado na definição do modelo matemático para o problema que estamos tratando. Assumimos a definição de estágio como sendo uma janela de tempo, onde um transporte e processamentos ocorrem. Para podermos utilizar convenientemente este conceito, dois vetores e duas matrizes auxiliares precisam ser, também, definidos:

1. Um vetor J contendo a ordem das peças, ou seja, j_i = peça k.

2. Uma matriz M de dimensão nm onde cada elemento m_{ij} é a identificação da máquina que processa o job i (j_i) no estágio j.

3. Uma segunda matriz P, contendo os tempos de processamento, onde cada elemento p_{ij} representa o tempo de processamento do job i no estágio j. Cabe reforçar que job i é a i-ésima peça do vetor J e p_{ij} é o tempo de processamento dessa peça na máquina m_{ij} da matriz M.

4. Um segundo vetor T contendo os tempos de transporte das peças pelo robô. Rigorosamente T deve ser uma matriz, sendo t_{ij} o tempo de transporte da peça correspondente ao job i no estágio j. Cabe considerar que os tempos de transporte são, em geral, função da peça, de um passo anterior e do passo atual (onde estava imediatamente antes de "pegar" a peça atual e para qual máquina vai levar). Neste trabalho, por simplicidade, vamos assumir que é função apenas do estágio, simplificando para um vetor T, cujos elementos t_i representam tempos de transporte das peças (indiscriminadamente) no estágio i.

Logo, cada coluna das matrizes M ou P define um estágio e em uma mesma janela de tempo pode existir um ou mais estágios.

A seguir, vamos analisar estratégia para equacionar situações de conflito entre máquinas. Para tanto, vamos considerar dois jobs u e v quaisquer e dois estágios j e m. Na situação de conflito de máquinas e robô apresentada na Figura 10a, *o estágio j é depois do estágio m*, vemos que se satisfazem as seguintes desigualdades de tempo. No caso de *conflito do robô*, temos que:

$$y_{vm} < x_{uj} \text{ e } y_{uj} < x_{vm} \Leftrightarrow x_{vm} - t_m < x_{uj} \text{ e } x_{uj} - t_j < x_{vm}$$

Logo, para evitar este tipo de conflito, precisamos colocar no modelo a seguinte exigência:

$$x_{vm} \leq y_{uj} \Leftrightarrow x_{vm} \leq x_{uj} - t_j$$

No caso de *conflito de máquinas* temos que:

$$x_{v,m-1} < y_{uj} \text{ e } x_{u,j-1} < y_{vm} \Leftrightarrow y_{vm} - P_{v,m-1} < y_{uj} \text{ e } y_{uj} - P_{u,j-1} < y_{vm}$$

Logo, para evitar este tipo de conflito, precisamos colocar no modelo, a seguinte exigência:

$$y_{vm} \leq x_{u,j-1} \Leftrightarrow x_{vm} - t_m \leq y_{uj} - P_{u,j-1}$$

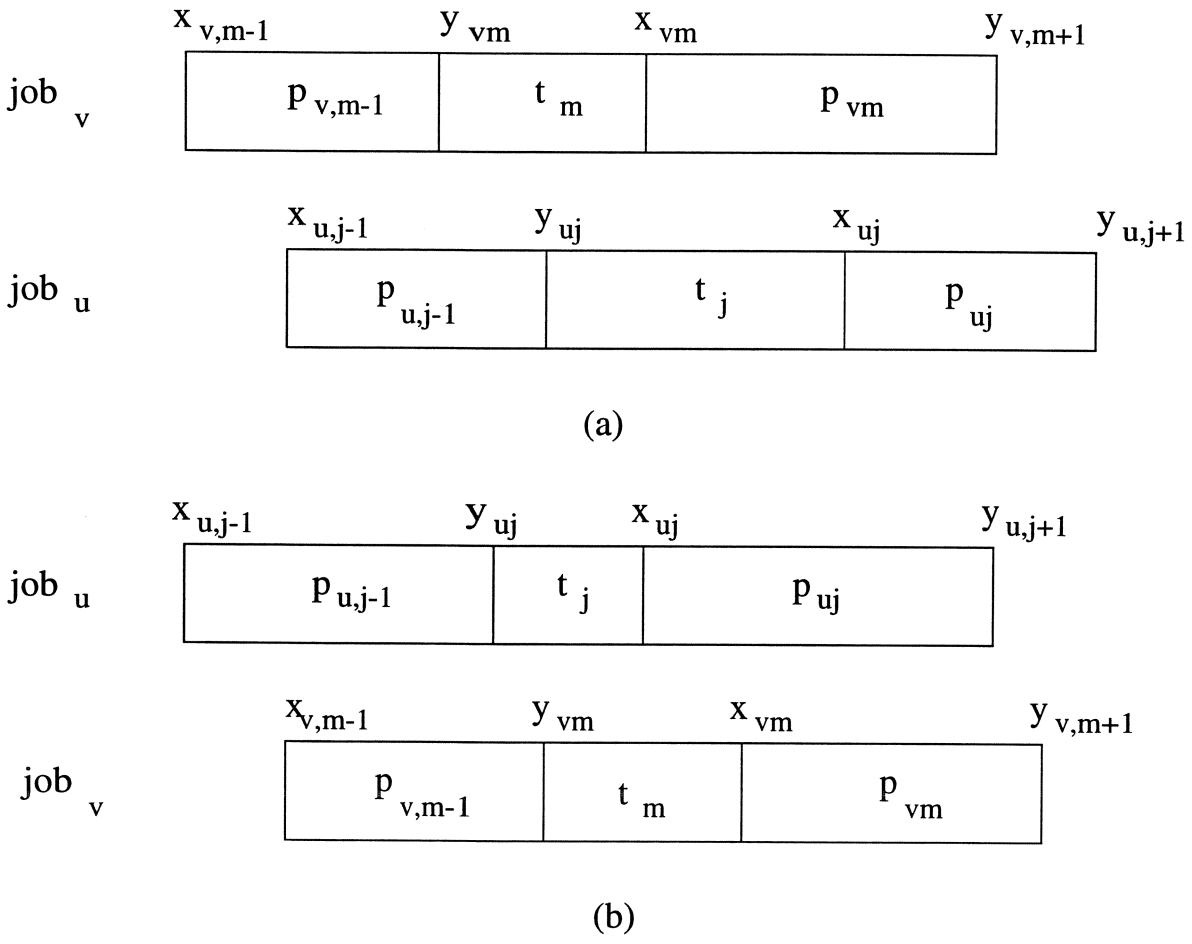


Figura 7: **Programação horária dos jobs u e v nos casos (a) j depois de m, (b) m depois de j.**

No caso em que *o estágio m é depois do estágio j* podemos ter a seguinte situação de conflito (Figura 7 - (b)).

Na situação de conflito do robô temos:

$$y_{uj} < x_{vm} \text{ e } y_{vm} < x_{uj} \Leftrightarrow x_{uj} - t_j < x_{vm} \text{ e } x_{vm} - t_m < x_{uj}$$

Assim, para evitar este tipo de conflito, deve-se anexar no modelo, a exigência:

$$y_{vm} \geq x_{uj} \Leftrightarrow x_{vm} - t_m \geq x_{uj}.$$

Na situação de conflito de máquinas temos:

$$x_{u,j-1} < y_{vm} \text{ e } x_{v,m-1} < y_{uj} \Leftrightarrow y_{uj} - P_{u,j-1} < y_{vm} \text{ e } y_{vm} - P_{v,m-1} < y_{uj}$$

Novamente, para evitar este conflito, deve-se impôr:

$$y_{uj} \leq x_{v,m-1} \Leftrightarrow x_{uj} - t_j \leq y_{vm} - P_{v,m-1}$$

Agora considerando o sequenciamento do job j e do robô em cada uma das m máquinas, seguem-se as seguintes restrições:

$$y_{uj} \geq x_{u,j-1} + P_{u,j-1}$$
$$x_{uj} - t_j \geq y_{uj}$$

A primeira desigualdade estabelece que o instante em que o job fica pronto para ser transportado ao estágio j é maior ou igual que o instante em que o job u finaliza seu processamento no estágio $j-1$ e a segunda desigualdade estabelece que o instante de início do transporte do job u para o estágio j é maior o igual que o instante em que o job u fica pronto para ser transportado nesse estágio. Logo, o problema de programação horária em células flexíveis de manufatura pode ser modelado como o seguinte problema de programação linear inteiro mixto:

Min Z

s.a. 1. $x_{uj} \geq y_{uj} + t_j \quad \forall u, \forall j$

2. $y_{uj} \geq x_{u,j-1} + P_{u,j-1} \quad \forall u, \forall j > 1$

3. $(y_{uj} - y_{vm} - P_{u,j-1})\delta'_{ujvm} + (y_{vm} - y_{uj} - P_{v,m-1})(1 - \delta'_{ujvm}) \geq 0$

para cada maquina com $\delta'_{ujvm} = \begin{cases} 1 & \text{se } j \text{ ocorre depois de } m \\ 0 & \text{se } m \text{ ocorre depois de } j \end{cases}$

4. $(x_{uj} - x_{vm} - t_j)\delta_{ujvm} + (x_{vm} - x_{uj} - t_m)(1 - \delta_{ujvm}) \geq 0$

com $\delta_{ujvm} = \begin{cases} 1 & \text{se } j \text{ ocorre depois de } m \\ 0 & \text{se } m \text{ ocorre depois de } j \end{cases}$

$x_{uj} \geq 0,\ x_{vm} \geq 0,\ y_{uj} \geq 0,\ y_{vm} \geq 0,\ \forall u \neq v,\ \forall j, \forall m.$

onde $Z = \sum_{i=1}^{n} \sum_{j=1}^{m} (x_{ij} + y_{ij}) + \sum_{i=1}^{n} y_{i,m+1}$

Quanto à restrição 3., a configuração das variáveis δ'_{ujvm}, que são em número máximo de $(n(n+1))/2$ para cada máquina define uma matriz M (ou seja, são no máximo $m * (n(n+1))/2$ variáveis que precisam ser definidas coerentemente). Variações de modelamento e métodos de resolução são perfeitamente possíveis de serem feitas sobre o modelo apresentado, visando obter melhores soluções. Neste trabalho, adotamos o seguinte modelo e estratégia:

Minimizar $Z = \sum_{i=1}^{n} \sum_{j=1}^{m} (x_{ij} + y_{ij}) + \sum_{i=1}^{n} y_{i,m+1}$

sujeito a:

$$x_{ij} - t_j \geq y_{ij} \quad \forall i, j \tag{2.1}$$

$$y_{ij} - x_{i,j-1} \geq P_{i,j-1}, \quad \forall i, \forall j > 1 \tag{2.2}$$

e utilizando as restrições:

$$(y_{uj} - y_{vm} - P_{u,j-1})\delta'_{ujvm} +$$
$$(y_{vm} - y_{uj} - P_{v,m-1})(1 - \delta'_{ujvm}) \geq 0 \quad (2.3)$$
$$(x_{uj} - x_{vm} - t_j)\delta_{ujvm} + (x_{vm} - x_{uj} - t_m)(1 - \delta_{ujvm}) \geq 0 \quad (2.4)$$

$$x_{ij}\ y_{ij} \geq 0, \forall i, j$$

$$\forall u \neq v\ \ \forall j, m$$

na ocorrência de conflitos, respectivamente de máquinas e de robô.

O seguinte algoritmo pode ser usado para a resolução de um problema de programação horária.

Algoritmo

Passo 1. Ordenar a matriz P de tempos de processamento segundo a regra LPT

Passo 2. Resolver o problema usando as restrições (2.1) e (2.2).

Passo 3. Existe algum tipo de conflito?

Se não, fim.

Se sim, ir resolvendo de "esquerda para a direita" e de "cima para baixo", da seguinte maneira:

a) Se têm conflito de máquinas, escolher a melhor alternativa entre o resultado obtido efetuando troca da ordem e anexando a restrição (2.3) e o resultado obtido com simples anexação da restrição (2.3).

b) Se têm conflito do robô anexar a restrição (2.4)

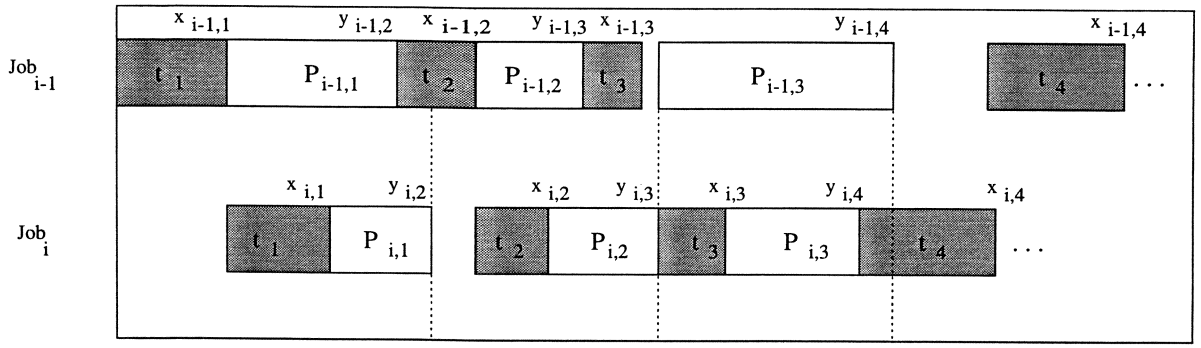


Figura 8: **Programação horária dos jobs i-1 e i sobre m máquinas.**

Passo 4. Resolver o novo problema e voltar ao *Passo 3.*

Com o objetivo de obter um modelo mais simples do problema, podemos considerar $u = i, v = i-1, m = j$ (Figura 8). Neste caso o problema de programação horária pode ser formulado como segue:

$$\text{Minimizar } Z = \sum_{i=1}^{n} \sum_{j=1}^{m} (x_{ij} + y_{ij}) + \sum_{i=1}^{n} y_{i,m+1}$$

sujeito a:

$$x_{ij} - t_j \geq y_{ij} \quad \forall i, j \tag{2.5}$$

$$y_{ij} - x_{i,j-1} \geq P_{i,j-1}, \quad \forall i, \quad \forall j > 1 \tag{2.6}$$

$$y_{ij} - y_{i-1,j} \geq P_{i,j-1}, \quad \forall i, \forall j > 1 \tag{2.7}$$

$$(x_{uj} - x_{vm} - t_j)\delta_{ujvm} + (x_{vm} - x_{uj} - tm)\,(1 - \delta_{ujvm}) \geq 0 \tag{2.8}$$

onde $\delta_{ujvm} = \begin{cases} 1 & \text{se } j \text{ ocorre depois de } m \\ 0 & \text{se } m \text{ ocorre depois de } j \end{cases}$

$$x_{ij} \; y_{ij} \geq 0, \forall i, j$$

$$\forall u \neq v \;\; \forall j, m$$

Neste modelo, força-se a ocorrência dos mesmos estágios para diferentes jobs numa mesma janela de tempo. Em geral, a solução deste problema pode ser

obtida usando técnicas de programação linear inteira mista (Taha, 1975)) com uso de regras de prioridades tais como SPT, LPT, FIFO, FCFS, MWR, LWR, etc.

2.4 Comentários.

Neste capítulo foram apresentados os conceitos básicos da programação horária de peças em FMC e a forma como este tipo de problema de complexidade NP-completo pode ser tratado através de heurísticas combinadas com regras de despacho .

Introduziu-se o conceito de *estágio* com a finalidade de utilizar os conceitos de instante de término do tempo de transporte de uma peça para um estágio e o instante de término do tempo de processamento de uma peça num estágio. A partir destes conceitos, apresentou-se dois modelos de programação linear inteira mista para o problema de programação horária de peças. O primeiro modelo geral considera a possibilidade de reordenar as peças de modo a permitir a redução do tempo total de processamento, programando as peças nas máquinas ociosas, permitindo sobreposição de diferentes estágios de diferentes jobs numa mesma janela de tempo definindo um único estágio de algum job. O segundo modelo é mais pobre, no sentido que em casos de coflito, soluciona o problema programando a peça para o instante de tempo em que a máquina em questão fica livre, prolongando assim, o tempo total, mas têm a vantagem de ser muito mais fácil de implementar, além de ser computacionalmente muito eficiente.

Capítulo 3
Heurísticas propostas

Neste capítulo propomos a solução do problema de programação horária aplicando uma metodologia baseada na lógica nebulosa, uma aproximação da solução baseada na programação liner nebulosa e uma aproximação baseada em métodos de ordenamento de números nebulosos, com aplicação da regra LTP, de modo a permitir o tratamento de objetivo não bem definido.

3.1 Heurística baseada na lógica nebulosa

A solução que propomos aqui baseia-se na lógica nebulosa correspondendo à uma solução onde todas as peças são processadas em cada máquina, analisando as sobreposições das tarefas de atendimento pelo robô e pelas máquinas. Se não houver sobreposição, esta solução é a aproximada do problema. Caso contrário usa-se a base da regras da lógica nebulosa e a regra LPT para reordenar o schedule de modo que não tenha sobreposições.

A necessidade de aplicar a teoria de conjuntos nebulosos para a resolução do problema de programação horária surge ante os dados que apresentam incertezas. Com a inclusão de valores linguísticos do tipo "pequeno, medio, grande" para os tempos de processamento e os tempos de transporte das peças, pretendemos definir um procedimento para programação horária que seja confiável e eficiente. A idéia básica para obter a programação horária consiste em ordenar os tempos de processamento das peças usando a regra LPT e gerar regras para efetuar as mudanças necessárias quando detectar máquinas conflitantes entre si. Estes conflitos podem ser detectados olhando a matriz de máquinas M que foi gerada junto com a matriz de tempos de processamento P. Se em uma coluna da matriz de máquinas M aparece repetida uma máquina k após ter aplicado a regra LPT, é assumido que há um conflito de

máquinas, devendo proceder a uma permutação entre os elementos da matriz M, e, portanto, também na matriz P, visando eliminar o conflito. Este procedimento deve ser feito de "cima para baixo", ou seja, no sentido de crescimento da ordem dos jobs, e da "esquerda para direita", ou seja, no sentido de crescimento do estágio. Ele é adequado para ser aplicado nos casos onde os tempos de processamento são valores similares, isto é, o tempo de processamento de uma peça em um estágio não difere muito do tempo de processamento de outra peça no mesmo estágio. No entanto, também é aplicável mesmo em outras situações, com eventuais prejuízos na qualidade das soluções obtidas.

A base de funções do sistema da forma de decisões para a programação horária é composta pelas grandezas que expressam as variações nos tempos de processamento representadas no sistema pelas variáveis de decisão y_{ij} que corresponde ao tempo de término do processo da peça i no estágio $j-1$ e x_{ij} que corresponde ao tempo de término do transporte da peça i para o estgio j.

Os valores para estas variáveis são fornecidos nos termos linguísticos *pequeno, medio, grande.* expressos pelos símbolos $G2$, $G4$, $G6$, respectivamente, por exemplo.

Para estes valores linguísticos, neste trabalho, foram criadas 7 partições nebulosas expressando variações: $G1$, $G2$, $G3$, $G4$, $G5$, $G6$, $G7$.

A Figura 9 representa as funções de pertinência dos valores linguísticos $G1, G2, G3, G4, G5, G6, G7$ da variável linguística *tempo de processamento.* A variação de processamento $G1$, têm o seu ponto de amplitude máxima entre 0 e $T/8$ minutos, onde T representa o maior tempo de processamento de uma tarefa. Os conceitos de variações pequena, baixa, média, alta e grande atingem seus valores máximos em $T/4$, $3T/8$, $T/2$, $5T/8$ e $6T/8$ minutos, respectivamente. O conceito de variação de processamento $G7$ atinge seu valor máximo com variações a partir de $7T/8\,\text{min.}$, sendo $T\,\text{min.}$ a variação máxima. Esta partição se extende seguindo o mesmo tipo de regra que para os valores que seguem em ordem de magnitude, chamados de $G8$, $G9$, $G10$, $G11$ e assim sucesivamente, até atingir todos os valores ne-

cessários para obter a programação horária das peças. Esta extensão é útil por causa dos diferentes valores linguísticos das somas dos tempos de processamento. Por exemplo, se consideramos os valores linguísticos representados por números nebulosos triangulares simétricos, isto é; números nebulosos da forma $A = [a, a-1, a+1]$ que podem-se representar simplesmente pelo intervalo $[a_1, a_2]$ onde $a_1 = a-1, a_2 = a+1$, então para o nível de certeza $\alpha \in [0,1]$, os números nebulosos $A_\alpha = [a_1(\alpha), a_2(\alpha)]$ e $B_\alpha = [b_1(\alpha), b_2(\alpha)]$ têm por soma o número nebuloso.

$$A_\alpha + B_\alpha = [a_1(\alpha) + b_1(\alpha), a_2(\alpha) + b_2(\alpha)].$$

Aqui consideramos números nebulosos triangulares simétricos da forma $G_i = [i-1, i+1]$ para os valores da partição. Assim, se $G7 = [6,8]$ e $G11 = [10,12]$ com funções de pertinência,

$$\mu_{G7}(x) = \begin{cases} 0 & \text{se } x \leq 6 \\ x-6 & \text{se } 6 \leq x \leq 7 \\ -x+8 & \text{se } 7 \leq x \leq 8 \\ 0 & \text{se } x \geq 8 \end{cases} \qquad \mu_{G11}(x) = \begin{cases} 0 & \text{se } x \leq 10 \\ x-10 & \text{se } 10 \leq x \leq 11 \\ -x+12 & \text{se } 11 \leq x \leq 12 \\ 0 & \text{se } x \geq 12 \end{cases}$$

então para $\alpha \in [0,1]$ temos:

$$\begin{aligned} \alpha &= g7_1(\alpha) - 6, \ \alpha = -g7_2(\alpha) + 8 \\ \alpha &= g11_1(\alpha) - 10, \ \alpha = -g11_2(\alpha) + 12 \end{aligned}$$

$$\begin{aligned} G7_\alpha &= [g7_1(\alpha), g7_2(\alpha)] \ ; & G11_\alpha &= [g11_1(\alpha), g11_2(\alpha)] \\ &= [\alpha + 6, -\alpha + 8] & &= [\alpha + 10, -\alpha + 12] \end{aligned}$$

$$\begin{aligned} G7_\alpha + G11_\alpha &= [g7_1(\alpha) + g11_1(\alpha), g7_2(\alpha) + g11_2(\alpha)] \\ &= [2\alpha + 16, -2\alpha + 20] \end{aligned}$$

A Tabela 3.1 (b) mostra os valores das somas destes valores linguísticos.

A base de regras do sistema é formada por um conjunto de regras de tipo *if then*, com a atribuição de valores linguísticos. Essa atribuição foi realizada como o

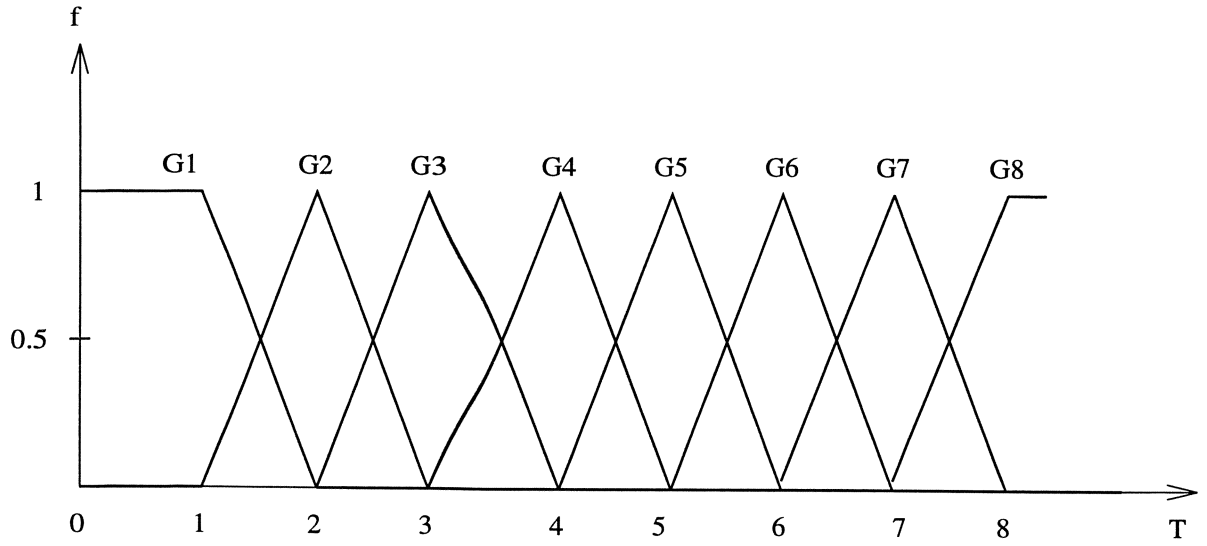


Figura 9: **Funções de pertinência dos valores linguísticos.**

preenchimento de tabelas da forma *Tabela 3.1 (a)*, que cobrem todas as combinações possíveis. Nesta mesma tabela damos os valores linguísticos das somas das diferentes combinações.

Para qualquer $\tilde{a}$ no conjunto VL={G1, G2, G3, G4, G5, G6, G7, G8, ...} definimos:

$$\mu_{\tilde{a}}(x) = \begin{cases} 1 & \text{se} \quad x = 1 \\ \dfrac{x - \underline{a}}{a - \underline{a}} & \text{se} \quad \underline{a} \leq x < a \\ \dfrac{\overline{a} - x}{\overline{a} - a} & \text{se} \quad a < x \leq \overline{a} \end{cases}$$

onde $\tilde{a} = [a, \underline{a}, \overline{a}]$ onde $\underline{a}, \overline{a}$ são os limites inferior e superior do $\tilde{a}$, respectivamente.

A seguir, baseado nestas regras apresentamos duas propostas de algoritmo.

t_{ij} $i = 1, \ldots, n$ $j = 1, \ldots, m$	$G1$	$G2$	t_{ik} $k = 2, \ldots, m$ $k > j$ $G3$	$G4$	$G5$	$G6$	$G7$
$G1$	t	$G2$	$G3$	$G4$	$G5$	G	
$G2$	$G2$	t	$G3$	$G4$	$G5$	$G6$	$G7$
$G3$	$G3$	$G3$	t	$G4$	$G5$	$G6$	$G7$
$G4$	$G4$	$G4$	$G4$	t	$G5$	$G6$	$G7$
$G5$	$G5$	$G5$	$G5$	$G5$	t	$G6$	$G7$
$G6$	$G6$	$G6$	$G6$	$G6$	$G6$	t	$G7$
$G7$	$G7$	$G7$	$G7$	$G7$	$G7$	$G7$	t

onde $t = \begin{cases} t_{ij} & \text{se } t_{ij} \geq t_{ik} \\ t_{ik} & \text{cc.} \end{cases}$

(a)

+	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	...
G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	...
G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	...
G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	...
G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	G14	...
G5	G6	G7	G8	G9	G10	G11	G12	G13	G14	G15	...
G6	G7	G8	G9	G10	G11	G12	G13	G14	G15	G16	...
G7	G8	G9	G10	G11	G12	G13	G14	G15	G16	G17	...
G8	G9	G10	G11	G12	G13	G14	G15	G16	G17	G18	...
G9	G10	G11	G12	G13	G14	G15	G16	G17	G18	G19	...
⋮					⋮						

(b)

Tabela 3.1 Valores linguísticos.

Algoritmo 3.1

Passo 1 Ler todos os dados de entrada, tais como o número de peças no buffer de entrada, o número de peças em processamento na FMC, o conjunto orde-

nado de máquinas, o tempo de processamento de cada peça em cada máquina (fornecidos através de valores linguísticos), o tempo de transporte das peças pelo robô por estágio (também fornecido em valores linguísticos), o instante de início da programação horária.

Passo 2 Criar uma matriz P de tempos de processamento das peças nas máquinas, criar uma matriz M das máquinas nas quais processam-se as peças na mesma ordem que os tempos de processamento e criar o vetor J dos jobs identificando as peças. Caso tenha mais jobs do que estágios (número de máquinas), definir estágios subsequentes (cada grupo de estágio deve ter no máximo m jobs).

Passo 3 Determinar um sequenciamento provisório usando a regra LPT. As matrizes M e P e o vetor J devem acompanhar as mudanças de linhas e de colunas, das peças e de máquinas.

Passo 4 Determinar as sobreposições do schedule atual:

- Se a sobreposição é zero ir para o passo 5.

- Caso contrário, na ordem crescente de estágios, efetuar permuta entre tarefas entre estágios subsequentes até eliminar o conflito. Novamente, este procedimento deve ser refletido nas matrizes M e P Repetir o passo 4.

Passo 5 Determinar a programação horária das peças sincronizando o robô e as máquinas, utilizando o vetor J e as matrizes M e P, assim como o vetor T.

Passo 6 Determinar a matriz A de tempos finais dos processos das peças nas máquinas, criar o vetor de tempos finais.

Passo 7 Determine o sequenciamento do robô, utilizando o resultado do sequenciamento, a matriz M e o vetor J.

Passo 8 Escreva os dados de saída com as medidas de representação e os gráficos de Gantt.

O segundo algoritmo é inspirado no primeiro, mas procurando melhorar os resultados, ampliou-se a utilização do conceito de estágios, permitindo agora, sobreposição de diferentes estágios de diferentes jobs. Este procedimento permite a ocorrência de mais de um estágio de um job dentro da janela de tempo de um estágio de outro job, procurando preencher ociosidades das máquinas e robô.

Algoritmo 3.2

Passo 1 Ler todos os dados de entrada, tais como o número de peças no buffer de entrada, o número de peças em processamento na FMC, o conjunto ordenado de máquinas, o tempo de processamento de cada peça em cada máquina (fornecidos através de valores linguísticos), o tempo de transporte das peças pelo robô por estágio (também fornecido em valores linguísticos), o instante de início da programação horária.

Passo 2 Criar uma matriz P de tempos de processamento das peças nas máquinas, criar uma matriz M das máquinas nas quais processam-se as peças na mesma ordem que os tempos de processamento, criar o vetor T de tempos de transporte e criar o vetor J dos jobs identificando as peças. Caso tenha mais jobs do que estágios (número de máquinas), definir estágios subsequentes (cada grupo de estágio deve ter no máximo m jobs).

Passo 3 Determinar um sequenciamento provisório usando a regra LPT na soma dos tempos de processamento de cada job. As matrizes M e P e o vetor J devem acompanhar as mudanças de linhas e de colunas, das peças e de máquinas.

Passo 4 Efetuar os seguintes procedimentos até o último estágio:

- Escolher o maior tempo de processamento t_{max} (valor linguístico) das tarefas que ainda não foram sequenciadas.
- Definir a janela de tempo do atual estágio baseado no t_{max} mais o tempo de transporte.
- Na sequência crescente dos jobs, sequenciar as tarefas cujas somas de tempos de processamentos mais os de transportes (valores linguísticos) sejam em torno de t_{max}, e que não conflitem com as tarefas já programadas no estágio atual. Caso nenhuma soma satisfaça a condição acima, escolher tarefa de maior tempo de processamento que não conflite com os jobs já programados no atual estágio.

Passo 5 Determinar a programação horária das peças sincronizando o robô e as máquinas, utilizando o vetor J e as matrizes M e P, assim como o vetor T.

Passo 6 Determinar a matriz A de tempos finais dos processos das peças nas máquinas, criar o vetor de tempos finais.

Passo 7 Determine o sequenciamento do robô, utilizando o resultado do sequenciamento, a matriz M e o vetor J.

Passo 8 Escreva os dados de saída com as medidas de representação e os gráficos de Gantt.

Para ilustrar este algoritmo, vamos considerar um problema composto de seis peças e cinco máquinas com tempos de processamento dados na matriz P que segue. Na notação adotada para matriz P, o elemento Gi/Mj significa que o job i tem uma peça $J(i)$ com tempo de processamento Gi na máquina M_j. Além disso, esta matriz já está ordenada segundo a regra LPT, isto é, o job 1 têm a peça com o maior soma de tempos de processamento e seus dados na ordem decrescente, o job 2 têm a peça com segundo maior tempo de processamento e seus dados em ordem

decrecente, e assim por diante até o job 6.

$$P = \left\{ \begin{array}{lllll} G9/M_2 & G7/M_1 & G6/M_4 & G5/M_3 & G4/M_5 \\ G8/M_2 & G6/M_4 & G5/M_5 & G2/M_1 & G3/M_3 \\ G8/M_4 & G5/M_2 & G5/M_5 & G3/M_3 & G2/M_1 \\ G6/M_2 & G5/M_1 & G4/M_5 & G2/M_3 & G1/M_4 \\ G5/M_1 & G5/M_2 & G4/M_3 & G3/M_5 & G1/M_4 \\ G5/M_2 & G5/M_3 & G3/M_5 & G1/M_1 & G1/M_4 \end{array} \right\}$$

A programação horária deste problema obtida aplicando o algoritmo 3.2 se encontra tabulado no quadro que segue, onde à direita dos jobs encontram-se as programações das máquinas nos diferentes estágios. Observe que em cada coluna estão programadas as cinco máquinas, critério usado para dar por completada a programação desse estágio; além disso, para as peças 5 e 6 foi criado um sexto estágio pois não tinham programado todas as suas tarefas nas máquinas.

Job 1	$G9/M_2$	$G7/M_1$	$G6/M_4$	$G5/M_3$	$G4/M_5$	
Job 2	$G6/M_4$ $G3/M_3$	$G8/M_2$	$G5/M_5$	$G2/M_1$		
Job 3	$G5/M_4$ $G2/M_1$	$G8/M_4$	$G5/M_2$ $G3/M_3$			
Job 4		$G4/M_5$ $G2/M_3$	$G5/M_1$	$G6/M_2$	$G1/M_4$	
Job 5				$G3/M_5$ $G1/M_4$	$G4/M_3$	$G5/M_1$ $G5/M_2$
Job 6					$G1/M_1$ $G5/M_2$	$G5/M_3$ $G3/M_5$ $G1/M_4$

Os instantes de tempos finais das peças em cada estágio estão calculados a seguir, onde as somas dos valores liguísticos foram determinados usando a tabela 3.1 (b). $G1$ é o tempo de transporte.

$$\text{Peça 1} = \left\{ \begin{array}{c} G1 + G9 = G10 \\ G1 + G10 + G7 = G18 \\ G1 + G18 + G6 = G25 \\ G1 + G25 + G5 = G31 \\ G1 + G31 + G4 = G36 \end{array} \right.$$

$$\text{Peça } 2 = \begin{cases} G2 + G6 + G1 + G3 = G12 \\ G1 + G12 + G8 = G21 \\ G1 + G21 + G5 = G27 \\ G1 + G27 + G2 = G30 \end{cases}$$

$$\text{Peça } 3 = \begin{cases} G3 + G5 + G1 + G2 = G11 \\ G1 + G11 + G8 = G20 \\ G1 + G20 + G5 + G1 + G3 = G30 \end{cases}$$

$$\text{Peça } 4 = \begin{cases} G8 + G1 + G4 + G1 + G2 = G16 \\ G1 + G16 + G5 = G22 \\ G1 + G22 + G6 = G29 \\ G1 + G29 + G1 = G31 \end{cases}$$

$$\text{Peça } 5 = \begin{cases} G31 + G1 + G3 + G1 + G1 = G37 \\ G1 + G37 + G4 = G42 \\ G1 + G42 + G5 + G1 + G5 = G54 \end{cases}$$

$$\text{Peça } 6 = \begin{cases} G31 + G1 + G1 + G1 + G5 = G39 \\ G1 + G39 + G5 + G1 + G3 + G1 + G1 = G51 \end{cases}$$

Este algoritmo não têm restrições com respeito as diferenças nos tempos de processamento das peças nas máquinas, pois vai resolvendo o problema de programação horária estágio por estágio, analizando todas as possibilidades de programação em cada um deles. Em outras palavras, quando completamos a programação em um estágio este não têm influência na programação do estágio posterior.

O exemplo da figura 10 ilustra a diferença de comportamento entre os dois algoritmos 3.1 e 3.2.

3.2 Heurística baseada na FLP

No capítulo 2, foi proposto o modelo 2 matemático para o nosso problema de programação horária e que, se assumirmos que o robô passa para o estágio seguinte somente após completar as tarefas do estágio atual, resulta no seguinte problema de programação linear clássico: Minimizar $Z = \sum_{i=1}^{n} \sum_{j=1}^{m} (x_{ij} + y_{ij}) + \sum_{i=1}^{n} y_{i,m+1}$

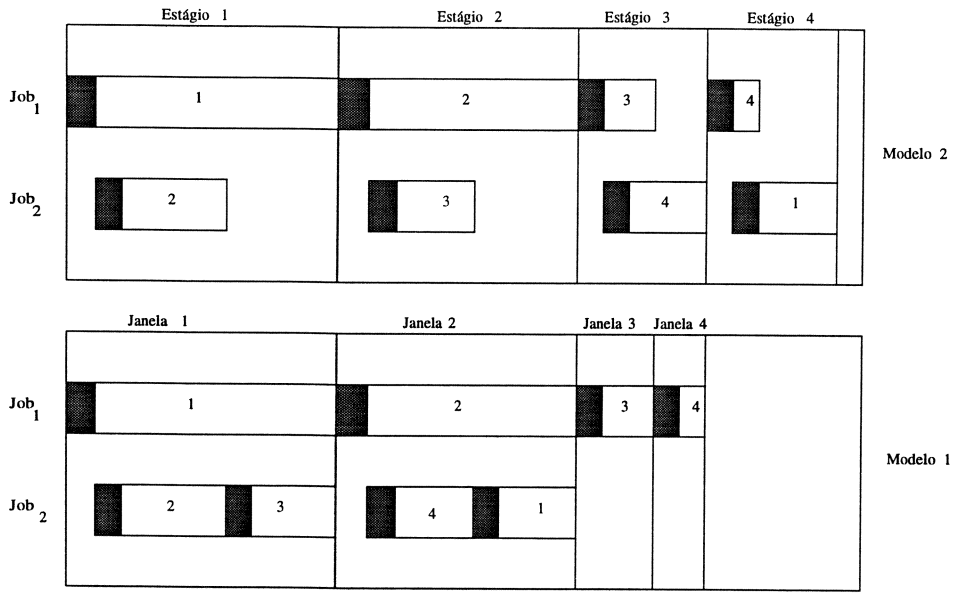


Figura 10: **Algoritmos 3.1 e 3.2 em um exemplo.**

sujeito a:

$$x_{ij} - t_j \geq y_{ij} \ \forall i,j \tag{3.1}$$

$$y_{ij} - y_{k,j-1} \geq P_{i,j-1}, \ \forall i,j > 1, \forall k \tag{3.2}$$

$$y_{ij} - x_{i,j-1} \geq P_{i,j-1}, \forall i, \forall j > 1 \tag{3.3}$$

$$x_{ij} - x_{i-1,j} - t_j \geq 0, \ \forall i > 1, \ \forall j \tag{3.4}$$

$$x_{1j} - x_{k,j-1} - t_j \geq 0, \ \forall j > 1, \forall k \tag{3.5}$$

$$\tag{3.6}$$

$$x_{ij} \ y_{ij} \geq 0, \forall i,j$$

Esta formulação foi concebida, tendo como pano de fundo o conceito de estágio que foi introduzido e utilizado no algoritmo 3.1, para permitir a aplicação da lógica fuzzy na sua abordagem linguística. Tem caráter experimental, considerando que este tipo de tratamento não foi encontrado na literatura e o grande objetivo é pemitir-nos análises e aprendizado.

Como pode se ver, é um problema de programação linear com variáveis reais, e portanto, de fácil solução. Introduzindo incertezas nos valores de $P_{i,j}$ e de t_j, ficamos com problema de programação linear fuzzy, o que traz a necessidade de trabalharmos com mais duas grandezas, quais sejam, uma matriz δP contendo as tolerâncias máximas nos tempos de processamentos das peças pelas máquinas e um vetor δT contendo as tolerâncias máximas nos tempos de transporte das peças pelo robô. Este problema pode ser resolvido, utilizando o algoritmo que segue.

Algoritmo 3.3

Passo 1 Ler todos os dados de entrada, tais como o número de peças no buffer de entrada, o número de peças em processamento na FMC, o conjunto ordenado de máquinas, o tempo de processamento de cada peça em cada máquina, as tolerâncias máximas nos tempos de processamento das peças pelas máquinas, o tempo de transporte das peças pelo robô por estágio, as tolerâncias máximas nos tempos de transporte das peças pelo robô, o instante de início da programação horária.

Passo 2 Criar uma matriz P de tempos de processamento das peças nas máquinas, criar uma matriz δP de tolerâncias máximas nos tempos de processamento das peças nas máquinas, criar uma matriz M das máquinas nas quais processam-se as peças na mesma ordem que os tempos de processamento, criar um vetor T de tempos de transporte das peças pelo robô, criar um vetor δT de tolerâncias máximas de tempos de transporte das peças pelo robô, criar o vetor J dos jobs identificando as peças. Caso tenha mais jobs do que estágios (número de máquinas), definir estágios subsequentes (cada grupo de estágio deve ter no máximo m jobs).

Passo 3 Determinar um sequenciamento provisório usando a regra LPT. As matrizes M, P e δP e os vetores T, δT e J devem acompanhar as mudanças de linhas e de colunas.

Passo 4 Determinar as sobreposições do schedule atual:

- Se a sobreposição é zero ir para o passo 5.

- Caso contrário, na ordem crescente de estágios, efetuar permuta entre tarefas entre estágios subsequentes até eliminar o conflito. Novamente, este procedimento deve ser refletido nas matrizes M, P e δP. Repetir o passo 4.

Passo 5 Resolver o problema de programação linear fuzzy:

Min Z

s.a.

$$\begin{aligned}
&(1)\ x_{ij} - y_{ij} \gtrsim \quad t_j \ \forall i,j \\
&(2)\ y_{ij} - y_{k,j-1} \gtrsim \quad P_{i,j-1} \ \forall i, \forall k, \ \forall j > 1 \\
&(3)\ y_{ij} - x_{i,j-1} \gtrsim \quad P_{i,j-1}, \ \forall i, \ \forall j > 1 \\
&(4)\ x_{ij} - x_{i-1,j} \gtrsim \quad t_j \ \forall j, \ \ \forall i > 1 \\
&(5)\ x_{1j} - x_{k,j-1} \gtrsim \quad t_j \ \forall j > 1, \forall k \\
&\quad x_{ij}, y_{ij} \geq 0
\end{aligned} \tag{3.7}$$

utilizando os vetores T, δT, J e as matrizes M, P e δP.

Passo 6 Determinar a matriz A de tempos finais dos processos das peças nas máquinas, criar o vetor de tempos finais.

Passo 7 Determine o sequenciamento do robô, utilizando o resultado do sequenciamento, a matriz M e o vetor J.

Passo 8 Escreva os dados de saída com as medidas de representação e os gráficos de Gantt.

Este algoritmo fornece uma solução sub-ótima mas têm a característica de ser computacionalmente eficiente, no sentido da convergência e rapidez em obtenção de resultados. O resultado da aplicação deste algoritmo pode ser visto mais claramente no exemplo que segue, onde as matrizes P e M já estão prontas, como pode ser observado na matriz M de máquinas cujas colunas não têm elementos repetidos. Isto significa que não têm conflitos de máquinas. A solução está mostrada no gráfico de Gantt da Figura 11.

$$P = \begin{pmatrix} 8 & 6 & 5 & 4 & 3 \\ 6 & 5 & 3 & 7 & 2 \\ 5 & 7 & 3 & 4 & 1 \end{pmatrix}; \qquad M = \begin{pmatrix} 3 & 5 & 1 & 4 & 2 \\ 2 & 1 & 5 & 3 & 4 \\ 4 & 3 & 2 & 1 & 5 \end{pmatrix}$$

Com a finalidade de definir um algoritmo para resolver o problema de programação horária utilizando o modelo 1 do capítulo 2, por simplicidade e sem perda de generalidade, neste trabalho vamos considerar o caso em que o deslocamento das tarefas efeita no estágio j que ocorre depois do m, significando que na eliminação de conflitos, não se atua sobre jobs anteriores. Vamos utilizar também, a regra LPT para inicializarmos o algoritmo. Desta forma, ficamos com o problema:

Min Z

s.a. 1. $x_{ij} \geq y_{ij} + t_j \qquad \forall i, \forall j$

2. $y_{ij} \geq x_{i,j-1} + P_{i,j-1} \qquad \forall i, \forall j > 1$

3. $(y_{uj} - x_{vm} + t_m - P_{u,j-1}) \geq 0$

4. $(x_{uj} - x_{vm} - t_j) \geq 0$

$$x_{ij} \geq 0,\ x_{vm} \geq 0,\ y_{ij} \geq 0,\ y_{vm} \geq 0,\ \forall u \neq v,\ \forall j,\ \forall i, \forall m.$$

com $Z = \sum_{i=1}^{n} \sum_{j=1}^{m} (x_{ij} + y_{ij}) + \sum_{i=1}^{n} y_{i,m+1}$ e a restrição 3. válida para cada máquina.

Este problema, novamente somente em variáveis reais, têm seu comportamento quanto à qualidade de solução, fortemente dependente da configuração da

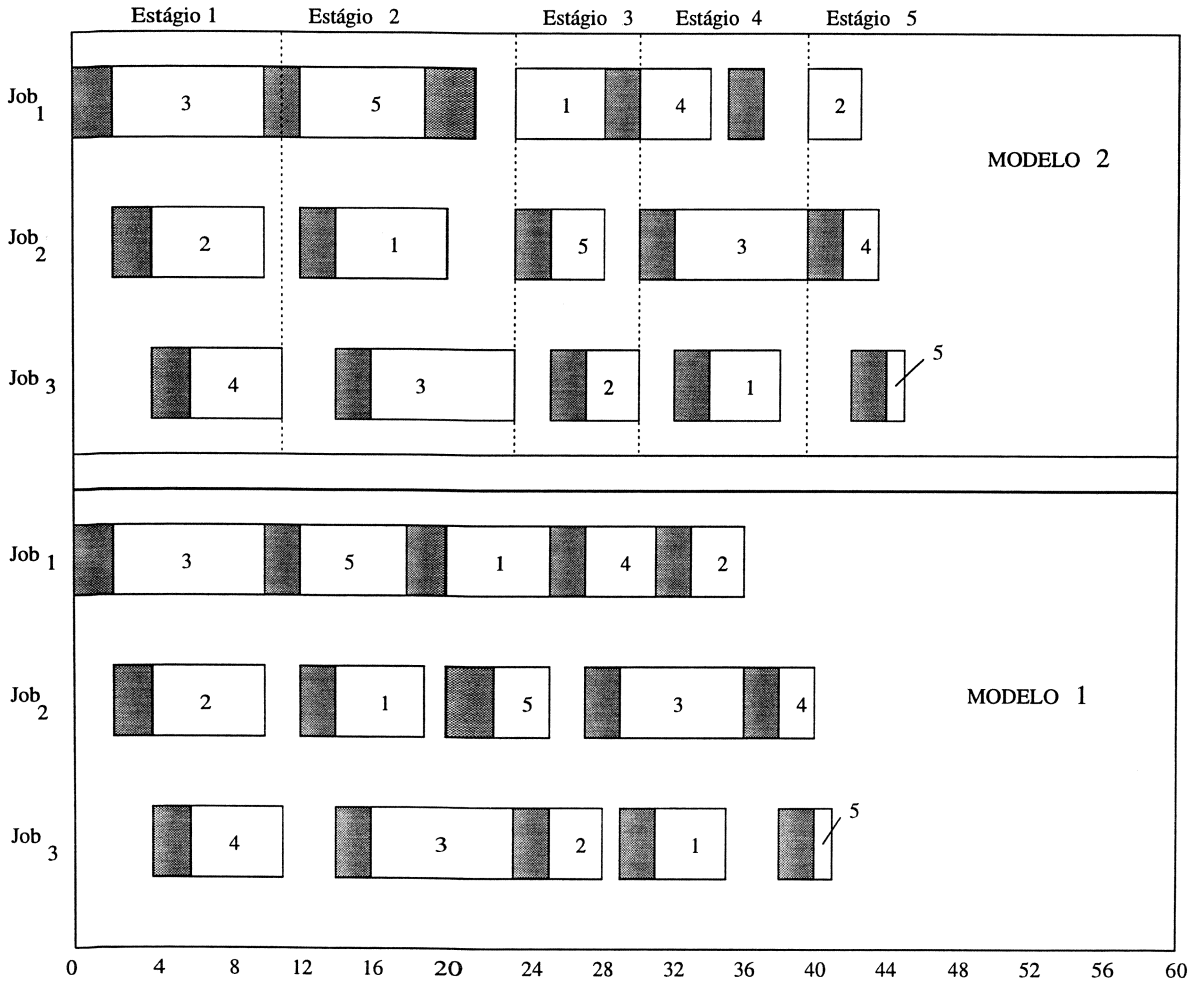


Figura 11: **Gráfico de Gantt da programação horária do exemplo usando o modelo 2 e o modelo 1, utilizando algoritmos 3.3 e 3.4**

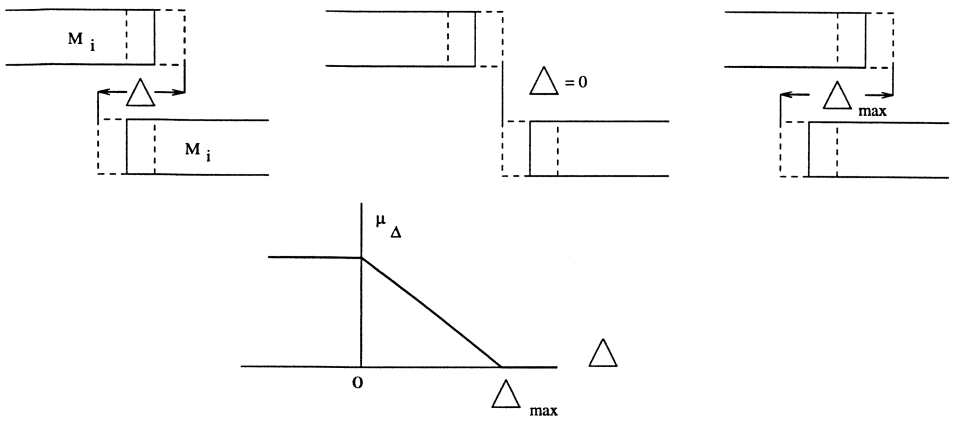


Figura 12: **Limiar de conflito.**

matriz M das máquinas. Esta configuração será definida passo a passo, utilizando alguma heurística adequada. Neste trabalho, vamos utilizar a regra LPT na soma dos tempos de processamento e de transportes por job em primeiro plano e, depois, sequencialmente de "cima para baixo", vamos resolvendo os conflitos com troca de posições das tarefas e/ou utilizando a terceira e a quarta restrições. A inicialização do processo pode ser feita relaxando estas duas últimas restrições, utilizando somente a primeira e a segunda. Cabe notar que neste caso, existe a necessidade de identificação de ocorrência ou não de conflitos para decidir sobre introdução ou não de restrições do tipo 3 ou 4. O problema de identificação de existência ou não destes conflitos, pode ser resolvido definindo um limiar de pertinência para a variável inteira que define a existência ou nào de conflito. Esta função de pertinência pode ser definida como na figura 12. Nesta figura, $\Delta > 0$ representa ocorrência de conflito e μ_Δ o nível deste conflito.

Algoritmo 3.4

Passo 1 Ler todos os dados de entrada, tais como o número de peças no buffer de entrada, o número de peças em processamento na FMC, o conjunto ordenado de máquinas, o tempo de processamento de cada peça em cada máquina, as

tolerâncias máximas nos tempos de processamento das peças pelas máquinas, o tempo de transporte das peças pelo robô por estágio, as tolerâncias máximas nos tempos de transporte das peças pelo robô, o instante de início da programação horária.

Passo 2 Criar uma matriz P de tempos de processamento das peças nas máquinas, criar uma matriz δP de tolerâncias máximas nos tempos de processamento das peças nas máquinas criar uma matriz M das máquinas nas quais processam-se as peças na mesma ordem que os tempos de processamento, criar um vetor T de tempos de transporte das peças pelo robô, criar um vetor δT de tolerâncias máximas de tempos de transporte das peças pelo robô, criar o vetor J dos jobs identificando as peças. Caso tenha mais jobs do que estágios (número de máquinas), definir estágios subsequentes (cada grupo de estágio deve ter no máximo m jobs).

Passo 3 Determinar um sequenciamento provisório usando a regra LPT na soma dos tempos de processamento de cada job. As matrizes M, P e δP e os vetores T, δT e J devem acompanhar as mudanças de linhas e de colunas,

Passo 4 Resolver o problema sem as restrições 3 e 4.

Passo 5 Sequencialmente, em ordem decrescente obtido no passo anterior, determinar a melhor solução parcial, fazendo combinações das tarefas do job em consideração e utilizando um valor pré-escolhido de valor de pertinência para Δ na identificação de existencia de conflitos. Para cada configuração, deve-se introduzir restrições que eliminam os conflitos ao problema e resolver:

Min Z

s.a.

$$(1)\ x_{ij} - y_{ij} \gtrsim t_j \quad \forall i, j$$

$$(2)\ y_{ij} - x_{i,j-1} \gtrsim P_{i,j-1}, \ \forall i, \ \forall j > 1$$

para cada máquina e para cada j que ocorre depois de m:

$$(3)\ y_{uj} - x_{vm} \gtrsim \quad P_{u,j-1} - t_m \quad \forall u, v,\ \forall j > 1,$$

$$(4)\ x_{uj} - x_{vm} \gtrsim \quad t_j \quad \forall u, v$$

$$x_{ij}, y_{ij} \geq 0$$

Passo 5 Determinar a programação horária das peças sincronizando o robô e as máquinas, utilizando o vetor J e as matrizes M e P, assim como o vetor T.

Passo 6 Determinar a matriz A de tempos finais dos processos das peças nas máquinas, criar o vetor de tempos finais.

Passo 7 Determinar o sequenciamento do robô, utilizando o resultado do sequenciamento, a mtriz M e o vetor J.

Passo 8 Escrever os dados de saída com as medidas de representação e os gráficos de Gantt.

Uma visão do comportamento de estes algoritmos no progresso da programação horária pode ser ilustrado como da Figura 12.

As restrições (3) e (4), portanto, serão usadas apenas nos casos de conflito de máquinas e robô. O resultado de aplicação deste algoritmo no mesmo exemplo do caso anterior, pode ser visualizado na Figura 12, em comparação com o algoritmo 3.3.

Se os tempos de transporte das peças pelo robô e os tempos de processamento das peças nas máquinas são "maiores ou iguais que" $j = 1, 2, \ldots, m$ respectivamente e com tolerâncias máximas correspondentes a α_i e β_{ij}, então o modelo (2) é enunciado como um problema linear fuzzy com uma função objetivo e um conjunto de restrições nebulosas definidas pelas suas funções de pertinência. Isto é:

Min Z

s.a.

$$(1)\ x_{ij} - y_{ij} \gtrsim \quad t_j; \quad \forall i, j$$

$$
\begin{aligned}
&(2)\ y_{ij} - y_{k,j-1} \gtrsim \quad P_{i,j-1};\ \ \forall i, \forall k,\ \forall j > 1 \\
&(3)\ y_{ij} - x_{i,j-1} \gtrsim \quad P_{i,j-1};\ \ \forall i,\ \forall j > 1 \\
&(4)\ x_{ij} - x_{i-1,j} \gtrsim \quad t_j;\ \ \forall j,\ \ \forall i > 1 \\
&(5)\ x_{1j} - x_{k,j-1} \gtrsim \quad t_j;\ \ \forall j > 1,\ \forall k \\
&\quad x_{ij}, y_{ij} \geq 0
\end{aligned}
\tag{3.8}
$$

Assim, as funções de pertinência μ_i para a i-ésima restrição nebulosa são:

$$
\mu_1(x) = \begin{cases} 1 & \text{se} \quad g_1(x) > t_j \\ \dfrac{g_1(x) - t_j + \alpha_j}{\alpha_j} & \text{se} \quad t_j - \alpha_j \leq g_1(x) \leq t_j \\ 0 & \text{se} \quad g_1(x) < t_j - \alpha_j \end{cases} \tag{3.9}
$$

$g_1(x) = x_{ij} - y_{ij},\ \forall i, j.$

$$
\mu_2(x) = \begin{cases} 1 & \text{se} \quad g_2(x) > P_{i,j-1} + \beta_{i,j-1} \\ \dfrac{g_2(x) - (P_{i,j-1} + \beta_{i,j-1})}{\beta_{i,j-1}} & \text{se} \quad P_{i,j-1} - \beta_{i,j-1} \leq g_2(x) \leq P_{i,j-1} \\ 0 & \text{se} \quad g_2(x) < P_{i,j-1} - (\beta_{i,j-1}) \end{cases}
$$

$g_2(x) = y_{ij} - y_{k,j-1} \quad \forall i, k,\ \forall j > 1.$

$$
\mu_3(x) = \begin{cases} 1 & \text{se} \quad g_3(x) > P_{i,j-1} \\ \dfrac{g_3(x) - P_{i,j-j} + \beta_{i,j-1}}{\beta_{i,j-1}} & \text{se} \quad P_{i,j-1} - \beta_{i,j-1} \leq g_3(x) \leq P_{i,j-1} \\ 0 & \text{se} \quad g_3(x) < P_{i,j-1} - \beta_{i,j-1} \end{cases}
$$

$g_3(x) = y_{ij} - x_{i,j-1} \quad \forall i, \forall j > 1.$

$$
\mu_4(x) = \begin{cases} 1 & \text{se} \quad g_4(x) > t_j \\ \dfrac{g_4(x) - t_j + \alpha_j}{\alpha_j} & \text{se} \quad t_j - \alpha_j \leq g_4(x) \leq t_j \\ 0 & \text{se} \quad g_4(x) < t_j - \alpha_j \end{cases}
$$

$g_4(x) = x_{ij} - x_{i-1,j}\ \forall j,\ \forall i > 1.$

$$\mu_5(x) = \begin{cases} 1 & \text{se} \quad g_5(x) > t_j \\ \dfrac{g_5(x) - t_j + \alpha_j}{\alpha_j} & \text{se} \quad t_j - \alpha_j \leq g_4(x) \leq t_j \\ 0 & \text{se} \quad g_5(x) < t_j - \alpha_j \end{cases}$$

$g_5(x) = x_{1j} - x_{k,j-1},\ \forall k,\ \forall j > 1.$

Este problema é equivalente ao problema:

$$\begin{aligned} &\text{Min} \quad Z \\ &\text{s.a.} \quad (1)\ g_1(x) \geq t_j - \alpha_j\theta \ \ \forall i,j \\ &\qquad (2)\ g_2(x) \geq P_{i,j-1} - (\beta_{i,j-1})\theta, \ \ \forall i,j > 1 \\ &\qquad (3)\ g_3(x) \geq P_{i,j-1} - \beta_{i,j-1}\theta, \ \ \forall i, \ \ \forall j > 1 \\ &\qquad (4)\ g_4(x) \geq t_j - \alpha_j\theta, \ \forall j, \ \forall i > 1 \\ &\qquad (5)\ g_5(x) \geq t_j - \alpha_j\theta, \ \forall j > 1 \end{aligned} \tag{3.10}$$

$x_{ij}, y_{ij} \geq 0$ e θ sendo um parâmetro em [0, 1].

Portanto usando as técnicas da programação linear paramétrica pode-se resolver este problema obtendo uma solução x^*, z^* que pode ser tabulada para diferentes valores do parâmetro $\theta \in [0, 1]$. Esta tabela pode ser utilizada para determinar W^0 e W^1 da aproximação de Werners(1987), com o objetivo de determinar uma solução do problema (3, 8) com função objetivo fuzzy, onde a meta b_0 da função objetivo nebulosa não é dada.

De acordo com a tabela obtida como solução de (3.10), $W^0 = Z^*(\theta = 1), W^1 = Z^*(\theta = 0)$ e o problema (3.8) pode ser formulado agora na forma:

$\widetilde{\text{Min}}\ Z$

s.a (3.10)

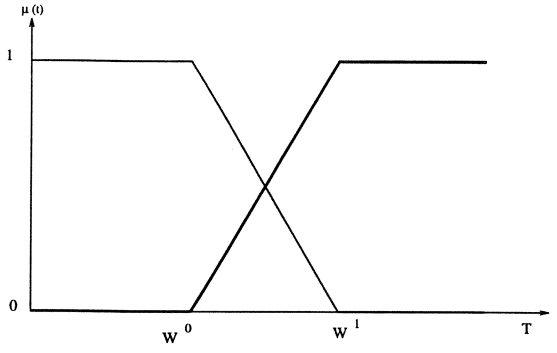


Figura 13: **Funções de pertinência da aproximação de Werners.**

com as funções de pertinência das restrições definidas em (3.9) e a função de pertinência da função objetivo fuzzy definida como (Figura 13):

$$\mu_0(x) = \begin{cases} 1 & \text{se } Z < W^0 \\ \dfrac{W^1 - Z}{W^1 - W^0} & \text{se } W^0 \leq Z \leq W^1 \\ 0 & \text{se } Z > W^1 \end{cases} \tag{3.11}$$

Este problema é equivalente a:

$$\begin{aligned} &\text{Min} \quad Z \\ &\text{sa.} \quad (1)\ Z \leq W^0 + \theta(W^1 - W^0) \\ &\quad (2)\ g_1(x) \geq t_j - \theta\alpha_j \ \ \forall i, j \\ &\quad (3)\ g_2(x) \geq P_{i,j-1} - \theta(\beta_{i,j-1})\ \forall i > 1, \forall j > 1 \\ &\quad (4)\ g_3(x) \geq P_{i,j-1} - \beta_{i,j-1}\theta\ \forall i, \forall j > 1 \\ &\quad (5)\ g_4(x) \geq t_j - \theta\alpha_j,\ \forall j,\ \forall i > 1 \\ &\quad (6)\ g_5(x) \geq t_j - \theta\alpha_j,\ \forall j > 1 \\ &\qquad x \geq 0,\ \theta \in [0, 1]. \end{aligned} \tag{3.12}$$

Resolvendo este problema, obtemos uma solução ótima associada. No entanto, é possível tabular resultados para diferentes valores do parâmetro θ, com finalidade

de fornecer maior visão da situação ao planejador, fornecendo maior opção. Além disso, se desta tabela podemos escolher uma meta b_0 para a solução e uma tolerância máxima γ_0 para b_0, então podemos usar a aproximação de Zimmermann (1984) para obtermos uma nova solução que tenha um grau de pertinência maior, no conjunto de soluções aceitáveis, do que a solução dada pela aproximação de Werner.

Seja então $b_0 = Z^*(\theta = \theta_0)$ e γ_0 tal que $0 \leq \gamma_0 \leq Z^*(\theta = \theta_0)$. O problema FLP fica:

"Determinar x tal que $Z \lesssim b_0$ com x satisfazendo as restrições (3.8), com funções de pertinência das restrições fuzzy dadas em (3.9) e a sua função de pertinência μ_0:

$$\mu_0(x) = \begin{cases} 1 & \text{se } Z < b_0 \\ \dfrac{Z - b_0 - \gamma_0}{-\gamma_0} & \text{se } b_0 \leq Z \leq b_0 + \gamma_0 \\ 0 & \text{se } Z > b_0 + \gamma_0 \end{cases} \tag{3.13}$$

da função objetivo."

Então, o problema precedente pode ser escrito na forma:

$$\begin{aligned} &\text{Min} \quad Z \\ &\text{sa.} \quad (1)\ Z \leq b_0 + \theta\gamma_0 \\ &\qquad (2)\ g_1(x) \geq t_j - \theta\alpha_j \quad \forall i, j \\ &\qquad (3)\ g_2(x) \geq P_{i,j-1} - (\beta_{i,j-1})\theta \ \forall i > 1, \forall j > 1 \\ &\qquad (4)\ g_3(x) \geq P_{i,j-1} - \beta_{i,j-1}\theta, \ \forall i, \forall j > 1 \\ &\qquad (5)\ g_4(x) \geq t_j - \theta\alpha_j, \ \forall j, \ \forall i > 1 \\ &\qquad (6)\ g_5(x) \geq t_j - \theta\alpha_j, \ \forall j > 1 \\ &\qquad\quad x \geq 0, \ \theta \in [0,1]. \end{aligned} \tag{3.14}$$

No caso em que a tolerância máxima γ_0 para b_0 não seja dada, podemos calcular a solução para diferentes valores de γ_0 pois $0 \leq \gamma_0 \leq Z^*(\theta = 0) - Z^*(\theta = \theta_0)$ e mostrar os resultados numa tabela para a escolha da melhor solução pelo planejador.

3.3 Algoritmo baseado em métodos de ordenamento de números nebulosos reais.

Algoritmo 3.5 Aplicação de métodos de ordenamento de números nebulosos reais.

Vamos considerar o FLP:

$$\begin{aligned} \textbf{Max } & Z = c^T x \\ \textbf{sa. } & \tilde{a}_i x \lesssim \tilde{b}_i \;\; i \in I\!N_m \\ & x \geq 0 \end{aligned} \tag{3.15}$$

$\tilde{a}_i = (\tilde{a}_{i1}, \tilde{a}_{i2}, \ldots, \tilde{a}_{in})$

O vetor de custos é suposto conhecido pelo planejador.

Por motivos de simplicidade e sem perda de generalidade, vamos supor números nebulosos triangulares denotados por:

$$\tilde{a} = (a, \underline{a}, \overline{a})$$

onde $\underline{a}$ e $\overline{a}$ denotam o limite inferior e superior do número nebuloso $\tilde{a}$.

Para esta classe de números nebulosos (Figura 14), e de acordo com o princípio de extensão, pode-se provar (Tanaka 1984) que a equação linear com coeficientes nebulosos $\tilde{a}_i x, \tilde{a}_i \in F(I\!R), x \in I\!R^n$ é outro número nebuloso com função de pertinência:

$$\mu_{\tilde{a}_i}(y) = \begin{cases} (y - \underline{a}_i)/(a_i - \underline{a}_i) & \text{se} \quad \underline{a}_i \leq y \leq a_i \\ (\overline{a}_i - y)/(\overline{a}_i - a_i) & \text{se} \quad a_i \leq y \leq \overline{a}_i \\ 0 & \text{caso contrário} \end{cases}$$

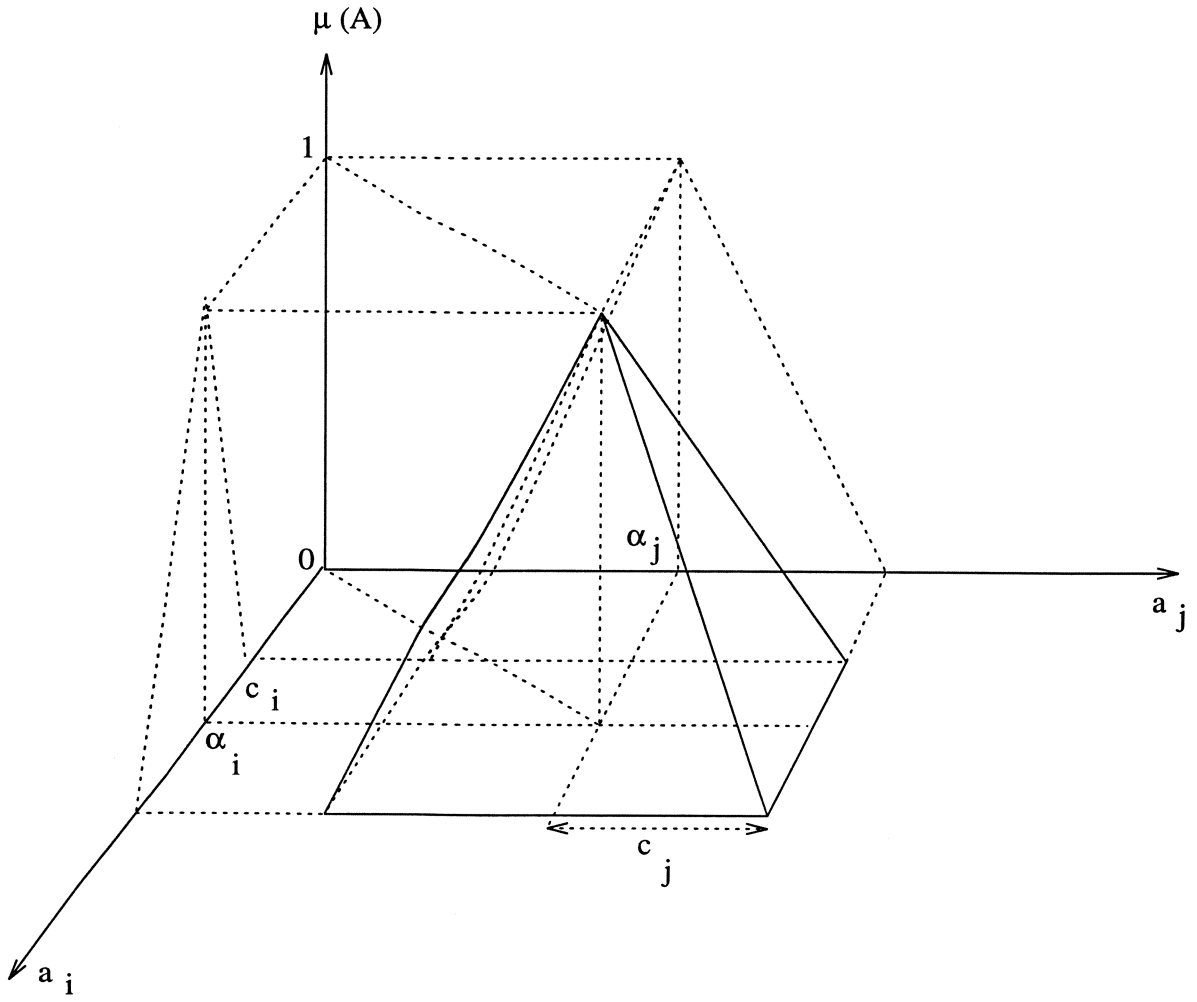


Figura 14: **Representação de $\tilde{a}$.**

Um método de resolução para o problema de programação linear nebuloso foi proposto em Delgado at al (1989). A aproximação consiste na substituição do conjunto de restrições nebulosas pelo seguinte conjunto convexo nebuloso:

$$\tilde{a}_i \leq_P \tilde{b}_i + \tilde{t}_i(1-\alpha), i \in I\!N_m, \alpha \in (0,1]$$

onde $\tilde{t}_i$ é um número nebuloso fixado pelo planejador e $\leq_P$ é qualquer relação entre os números nebulosos que devem ser escolhidos de modo que preservem a ordem quando os números nebulosos são multiplicados por um escalar positivo.

O problema, então, fica:

$$\begin{array}{lll} \text{Max} & Z = c^T x & \\ \text{sa.} \quad \tilde{a}_i & \leq_P \quad \tilde{b}_i + \tilde{t}_i(1-\alpha), i \in I\!N_m & (3.16) \\ & x \geq 0, \alpha \in (0,1] & \end{array}$$

e pode-se obter uma solução nebulosa para (3.15).

De (3.16), podem-se desenvolver modelos de programação linear auxiliares, considerando o fato de que $\leq_P$ é dado por qualquer método para ordenar número nebulosos. Esses métodos para ordenar números nebulosos são gerados usando diferentes aproximações. Por exemplo:

(I) usando função de ordenamento.

(II) usando grau de conformidade para o qual a restrição pode ser considerada verdadeira. Este grau é escolhido a priori pelo planejador.

(I) Sejam $\tilde{u}_i, i \in I\!N_n$, n conjuntos nebulosos convexos normais. Um método para ordenar os $\tilde{u}_i$ consiste na definição de uma função de ordenamento F que

mapeia cada conjunto nebuloso de $I = [0,1]$ em $\mathbb{R}$, onde existe uma ordem natural

$$\tilde{u}_i = \{x, \mu_{\tilde{u}_i}(x)\}, x \in \mathrm{S_i} \subset I$$

Seja $F : \mathcal{P}(I) \rightarrow \mathbb{R}$ tal que:

$$F(\tilde{u}_i) < F(\tilde{u}_j) \Rightarrow \tilde{u}_i < \tilde{u}_j$$
$$F(\tilde{u}_i) = F(\tilde{u}_j) \Rightarrow \tilde{u}_i = \tilde{u}_j$$
$$F(\tilde{u}_i) > F(\tilde{u}_j) \Rightarrow \tilde{u}_i > \tilde{u}_j$$

então, F é uma função de ordenamento dos conjuntos nebulosos. Yager(1980) propos várias funções de ordenamento. O primeiro índice de Yager é definido por:

$$F_1(\tilde{u}_i) = \frac{\int_0^1 g(x)\mu_{\tilde{u}_i}(x)dx}{\int_0^1 \mu_{\tilde{u}_i}(x)dx}$$

onde o peso $g(x)$ é uma medida da importância do valor x. Se supomos pesos lineares, isto é, $g(x) = x$ então $F_1(\tilde{u}_i)$ representa o centro da gravidade do conjunto nebuloso $\tilde{u}_i$.

O segundo índice de Yager é sugerido pela teoria da possibilidade:

$$F_2(\tilde{u}_i) = \max_{x \in S_i} \min(x, \mu_{\tilde{u}_i}(x))$$

Neste caso, $F_2(\tilde{u}_i)$ mede a consistência de $\tilde{u}_i$ com $\tilde{x}$ definido por $\mu_{\tilde{x}}(x) = x$.

O terceiro índice de Yager é como segue: se U_i^α é o α-nível de $\tilde{u}_i$ e se $M(U_i^\alpha)$ é o valor médio dos elementos de U_i^α então:

$$F_3(\tilde{u}_i) = \int_0^{\alpha_{\max}} M(U_i^\alpha)d\alpha$$

onde $\alpha_{\max} = hgt(\tilde{u}_i) =$ altura de $\tilde{u}_z$.

Adamo(1980), usa o conceito de α-nível para obter um índice α-preferência que é dado por:

$$F_\alpha(\tilde{u}_i) = \max\{z/\mu_{\tilde{u}_i}(z) \geq \alpha\}$$

para um limite superior dado $\alpha \in I$.

Dubois and Prade(1979) propõem um conjunto de quatro índices capazes de descrever completamente a posição relativa dos números nebulosos $\tilde{u}_i$ e $\tilde{u}_j$.

1. Um grau de possibilidade de dominância PD (de $\tilde{u}_i$ sobre $\tilde{u}_j$)

$$\begin{aligned} PD(\tilde{u}_i) &= \prod_{\tilde{u}_i}([\tilde{u}_j, \infty)) = \text{Poss } (\tilde{u}_i \geq \tilde{u}_j \\ &= \sup_{z_i} \min[\mu_{\tilde{u}_i}(z_i), \sup_{z_j < z_i} \mu_{\tilde{u}_j}(z_j)] \\ &= \sup_{z_i \geq z_j} \min[\mu_{\tilde{u}_i}(z_i), \mu_{\tilde{u}_j}(z_j)] \end{aligned}$$

2. Um grau de possibilidade de dominância estrita PSD

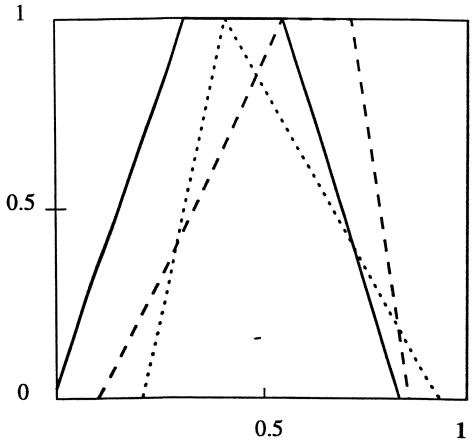
$$\begin{aligned} PSD(\tilde{u}_i) &= \prod_{\tilde{u}_i}((\tilde{u}_j, \infty)) = \text{Poss } (\tilde{u}_i > \tilde{u}_j) \\ &= \sup_{z_i} \inf_{z_j \leq z_i} \min[\mu_{\tilde{u}_i}(z_i), 1 - \mu_{\tilde{u}_j}(z_j)] \end{aligned}$$

3. Um grau de necessidade de dominância ND

$$\begin{aligned} ND(\tilde{u}_i) &= \mathcal{N}_{\tilde{u}_i}([\tilde{u}_j, \infty)) = \text{Nec } (\tilde{u}_i \geq \tilde{u}_j) \\ &= \inf_{z_i} \sup_{z_j \leq z_i} \max[1 - \mu_{\tilde{u}_i}(z_i), \mu_{\tilde{u}_j}(z_j)] \end{aligned}$$

4. Um grau de necessidade de dominância estrita NSD

$$\begin{aligned} NSD(\tilde{u}_i) &= \mathcal{N}_{\tilde{u}_i}(\tilde{u}_j, \infty)) = \text{Nec } (\tilde{u}_i > \tilde{u}_j) \\ &= 1 - \sup_{z_i \leq z_j} \min[\mu_{\tilde{u}_i}(z_i), \mu_{\tilde{u}_j}(z_j)] \\ &= 1 - PD(\tilde{u}_j) \end{aligned}$$



	PD	PSD	ND	NSD
$\tilde{u}_1$	1.00	0.30	0.30	0.00
$\tilde{u}_2$	0.88	0.40	0.50	0.00
$\tilde{u}_3$	1.00	0.60	0.50	0.00

Tabela 3.2 Indices de Dubois e Prade em um exemplo.

$\tilde{u}_1$ ——— $\tilde{u}_2$ ····· $\tilde{u}_3$ - - - -

Figura 15: **Gráfico dos índices de Dubois e Prade em um exemplo.**

Um exemplo é mostrado na tabela 3.2, onde os $\tilde{u}_i$ (Figura 15) estão sobrepostos. A interpretação dos resultados pode ser nenhum conjunto é necessàriamente estritamente dominante pois os suportes se sobrepõem; $\tilde{u}_2$ e $\tilde{u}_3$ são maiores que $\tilde{u}_1$ no lado esquerdo (ND); $\tilde{u}_3$ é maior que $\tilde{u}_1$ e $\tilde{u}_2$ no lado direito (PSD); $\tilde{u}_1$ e $\tilde{u}_3$ podem ser maiores que $\tilde{u}_2$ sobre a direita (PD).

Vamos usar agora, esses índices.

Seja $F : F(I\!R) \rightarrow I\!R$ a função de ordenamento, então:

$$\tilde{a} \leq \tilde{b} \Rightarrow F(\tilde{a}) \leq F(\tilde{b}) \tag{3.17}$$

$$\text{com } F_1(\tilde{a}) = \frac{\int_{\underline{a}}^{\overline{a}} g(x)\mu_{\tilde{a}}(x)dx}{\int_{\underline{a}}^{\overline{a}} \mu_{\tilde{a}}(x)dx} \tag{3.18}$$

$$\text{e } F_3(\tilde{a}) = \int_0^{\alpha_{\max}} M[\underline{a}_\alpha, \overline{a}_\alpha]d\alpha \tag{3.19}$$

onde $\alpha_{\max} = hgt(\overline{a})$. $[\underline{a}_\alpha, \overline{a}_\alpha]$ é o conjunto correspondente ao α-corte de $\tilde{a}$ e $M[\underline{a}_\alpha, \overline{a}_\alpha]$ é o valor médio dos elementos desse α-nível. Considerando os número nebulosos $\tilde{a}, \tilde{b}$ com função de pertinência fp triangulares, de (3.17) temos os seguintes resultados para $\leq_P$.

(i) Com o primeiro índice de Yager e $g(x) = x$:

$$\tilde{a} \leq_P \tilde{b} \Rightarrow \overline{a} + \underline{a} + a \leq \overline{b} + \underline{b} + b \tag{3.20}$$

(ii) Com o terceiro índice de Yager temos:

$$\tilde{a} \leq_P \tilde{b} \Rightarrow \overline{a} + \underline{a} + 2a \leq \overline{b} + \underline{b} + 2b \tag{3.21}$$

(II) Com respeito a segunda aproximação, em Adamo (1980) foi sugerida a k-preferência F_k para classificar número nebuloso:

$$F_k(\tilde{a}) = \max\{x/\mu_{\tilde{a}}(x) \geq k\}, \text{ para um dado nível } k \tag{3.22}$$

De acordo com este método, se o planejador fixa a priori um grau de satisfação $k \in [0, 1]$, então:

$$\tilde{a} \leq_P \tilde{b} \text{ com grau } k \Rightarrow F_k(\tilde{a}) \leq F_k(\tilde{b}) \tag{3.23}$$

Se $\tilde{a}$ e $\tilde{b}$ são números nebulosos triangulares então:

$$ka + (1-k)\overline{a} \leq kb + (1-k)\overline{b} \tag{3.24}$$

Da definição dos índices de Dubois e Prade, dados dois números nebulosos $\tilde{a}$ e $\tilde{b}$ definimos particularmente:

(1) Um grau de possibilidade de dominância de $\tilde{b}$ sobre $\tilde{a}$ como:

$$PD(\tilde{a} \leq \tilde{b}) = \sup_{X_j \leq x_i} \min[\mu_{\tilde{b}}(x_i), \mu_{\tilde{a}}(x_j)]$$

(2) Um grau de necessidade de dominância de $\tilde{b}$ sobre $\tilde{a}$ como:

$$ND(\tilde{a} \leq \tilde{b}) = \inf_{x_i} \sup_{X_j \leq x_i} \max[1 - \mu_{\tilde{b}}(x_i), \mu_{\tilde{a}}(x_j)]$$

Quando $\tilde{a}$ e $\tilde{b}$ são números nebulosos triangulares, os graus acima são:

$$PD(\tilde{a} \leq \tilde{b}) = \begin{cases} 0 & \text{se } \overline{b} \leq \underline{a} \\ \dfrac{\overline{b} - \underline{a}}{\overline{b} - b + a - \underline{a}} & \text{se } a \geq b \text{ e } \overline{b} \geq \underline{a} \\ 1 & \text{se } a \leq b \end{cases}$$

$$ND(\tilde{a} \leq \tilde{b}) = \begin{cases} 0 & \text{se } b \leq \underline{a} \\ \dfrac{b - \underline{a}}{a - \underline{a} + b - \underline{b}} & \text{se } b \geq a \text{ e } a \geq \underline{b} \\ 1 & \text{se } a \leq \underline{b} \end{cases}$$

Se um grau $k \in [0,1]$ é fixado a priori pelo planejador, então

$$\tilde{a} \leq_P \tilde{b} \Leftrightarrow PD(\tilde{a} \leq \tilde{b}) \geq k \tag{3.25}$$

$$\tilde{a} \leq_P \tilde{b} \Leftrightarrow ND(\tilde{a} \leq \tilde{b}) \geq k \tag{3.26}$$

No caso de números nebulosos triangulares, (3.23) e (3.24) implicam:

$$ka + (1-k)\underline{a} \leq kb + (1-k)\overline{b} \tag{3.27}$$

$$ka + (1-k)\underline{a} \leq (1-k)b + k\underline{b} \tag{3.28}$$

Com a informação processada até aqui, vamos gerar modelos auxiliares para resolver o problema de programação nebuloso. Levando em conta que as funções

de pertinência dos números nebulosos foram considerados para ser triangulares, as funções de pertinência dos números nebulosos do lado direito de cada restrição são:

$$[b_i + t_i(1-\alpha), \underline{b}_i + \underline{t}_i(1-\alpha), \overline{b}_i + \overline{t}_i(1-\alpha)], i \in I\!N_m \tag{3.29}$$

onde os limites inferior e superior no k-corte do número nebuloso (3.28) são:

$$k[b_i + t_i(1-\alpha)] + (1-k)[\underline{b}_i + \underline{t}_i(1-\alpha)]$$
$$k[b_i + t_i(1-\alpha)] + (1-k)[\overline{b}_i + \overline{t}_i(1-\alpha)]$$

respectivamente. Para resolver o problema nebuloso faz-se a substituição de $\leq_P$ para cada relação entre números nebulosos determinados acima e sua redução aos números nebulosos:

$\tilde{a}_i$ e $\tilde{b}_i + \tilde{t}_i(1-\alpha)$.

Da aproximação do tipo I, podemos estabelecer o que segue.

A partir de (3.20), podemos definir o modelo auxiliar:

$$\begin{aligned} &\text{Max } Z = c^T x \\ &\text{sa. } (\overline{a}_i + \underline{a}_i + a_i)x \leq \overline{b}_i + \underline{b}_i + b_i + (\overline{t}_i + \underline{t}_i + t_i)\,(1-\alpha) \\ &\quad x \geq 0,\ \alpha \in (0,\,1], i \in I\!N_m \end{aligned} \tag{3.30}$$

onde é mantida a linearidade e a dimensão do problema linear nebuloso. De (3.21) obtemos:

$$\begin{aligned} &\text{Max } Z = c^T x \\ &\text{sa. } (\overline{a}_i + \underline{a}_i + 2a_i)x \leq \overline{b}_i + \underline{b}_i + 2b_i + [\overline{t}_i + \underline{t}_i + 2t_i]\,(1-\alpha) \\ &\quad x \geq 0,\ \alpha \in (0,\,1], i \in I\!N_m \end{aligned} \tag{3.31}$$

Considerando aproximações do tipo que fixa o grau $k \in (0,1]$, obtemos o problema auxiliar correspondente a (3.24):

$$\begin{aligned}
&\text{Max} \quad Z = c^T x \\
&\text{sa.} \quad [(1-k)\overline{a}_i + ka_i]x \leq (1-k)\overline{b}_i + kb_i + [(1-k)\overline{t}_i + kt_i](1-\alpha) \qquad (3.32) \\
&\qquad x \geq 0, \ \alpha \in (0,1], i \in I\!N_m,
\end{aligned}$$

De manera similar, para $k \in (0,1]$ fixado pelo planejador obtemos o problema auxiliar correspondente a (3.25) e (3.27) :

$$\begin{aligned}
&\text{Max} \quad Z = c^T x \\
&\text{sa.} \quad [(1-k)\underline{a}_i + ka_i]x \leq (1-k)\overline{b}_i + kb_i + [(1-k)\overline{t}_i + kt_i](1-\alpha) \qquad (3.33)
\end{aligned}$$

e o problema auxiliar correspondente a (3.26) e (3.28) :

$$\begin{aligned}
&\text{Max} \quad Z = c^T x \\
&\text{sa.} \quad [(1-k)\underline{a}_i + ka_i]x \leq (1-k)b_i + k\underline{b}_i + [(1-k)t_i + k\underline{t}_i](1-\alpha) \qquad (3.34) \\
&\qquad x \geq 0, \ \alpha \in (0,1], i \in I\!N_m
\end{aligned}$$

Assim, o modelo nebuloso para o problema de programação horária proposto pela heurística baseada no ordenamento de números nebulosos pode ser formulado como: Min Z

$$\begin{aligned}
&\text{sa.} \ \ y_{ij} - x_{ij} \leq_p -\tilde{t}_j + t\tilde{t}_j(1-\alpha); \qquad \forall i,j \\
&y_{k,j-1} - y_{ij} \leq_p -\tilde{P}_{i,j-1} + (-t\tilde{P}_{i,j-1})(1-\alpha); \ \forall i,k, \forall j > 1 \\
&x_{i,j-1} - y_{ij} \leq_p -\tilde{P}_{i,j-1} + t\tilde{P}_{i,j-1}(1-\alpha); \ \forall i, \ \forall j > 1 \\
&x_{i-1,j} - y_{ij} \leq_p \ -\tilde{t}_j + t\tilde{t}_j(1-\alpha); \qquad \forall i > 1, \ \forall j \\
&x_{k,j-1} - x_{1,j} \leq_p \ -\tilde{t}_j + t\tilde{t}_j(1-\alpha); \qquad \forall j > 1, \ \forall k \\
&x_{ij} \geq 0, y_{ij} \geq 0, \quad \alpha \in (0,1].
\end{aligned}$$

Aqui, $t\tilde{t}_j$ representa a tolerância nebulosa do número nebuloso $\tilde{t}_j$, $t\tilde{P}_{i,j-1}$ representa a tolerância nebulosa do número nebuloso $\tilde{P}_{i,j-1}$. O símbolo $\leq p$ representa um método de ordenamento dos números nebulosos definidos precedentemente. De

acordo com o método escolhido, pode-se gerar um modelo auxiliar para resolver o nosso problema, por exemplo (3.30), (3.31), (3.32), (3.33) e (3.34).

3.4 Comentários.

Neste capítulo foram apresentadas cinco heurísticas, duas baseadas na abordagem linguística da lógica nebulosa, outras duas baseadas na teoria de programação linear nebulosa e a quinta heurística baseada no conceito de ordenamento dos números nebulosos. A primeira e a terceira heurísticas foram elaboradas com simplicidade suficiente para permitir-nos, de maneira clara e segura, elaboração, aplicação e análise no uso da lógica nebulosa nos problemas que nos propomos, quais sejam, os de programação horária de manufatura de peças em células fleíveis que apresentam incertezas. As duas subsequentes, a segunda e a quarta heurísticas são uma tentativa de melhorar os algoritmos anteriores, procurando aproximar das técnicas exatas de programação. Certamente, a partir deste nosso aprendizado, é possível extender os resultados para outras abordagens e heurísticas. Na abordagem via programação linear fuzzy, foram utilizados procedimentos como de Zimmermann (1978) e Werners (1987). Finalmente, a quinta heurística foi baseada nos conceitos encontrados em Delagado et al (1989).

No capítulo que segue, vamos aplicar os algoritmos propostos aqui em alguns exemplos e comentar os resultados, analisando e comparando.

Capítulo 4
Exemplos de aplicação

Vamos apresentar neste capítulo, diversos exemplos de aplicação dos algoritmos 3.1, 3.2, 3.3, 3.4 e o método de ordenamento de números nebulosos aplicado ao modelo 2 do capítulo 2 (algoritmo 3.5), todos estudados no capítulo anterior.

Consideramos inicialmente, um exemplo com 3 jobs e 5 máquinas para ilustrar as heurísticas. Os dados estão apresentados na Tabela 4.1, onde PT indica o tempo total de processamento das peças menos o tempo de transporte.

4.1 Aplicação do algoritmo 3.1

Passo 1: Dados do problema

Jobs	$t_j = G1$ para todos os jobs					PT
1	G5/M1	G3/M2	G8/M3	G4/M4	G6/M5	G26
2	G4/M1	G3/M2	G7/M3	G5/M4	G1/M5	G20
3	G5/M1	G6/M2	G7/M3	G2/M4	G3/M5	G23

Tabela 4.1 Dados do problema

Passo 2: Construção das matrizes e vetores

$$M = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}$$

$$P = \begin{pmatrix} G5 & G3 & G8 & G4 & G6 \\ G4 & G3 & G7 & G5 & G1 \\ G5 & G6 & G7 & G2 & G3 \end{pmatrix}$$

$$J = \begin{pmatrix} \text{peça } 1 \\ \text{peça } 2 \\ \text{peça } 3 \end{pmatrix}$$

Passo 3: Aplicação da Heurística 3.1

Jobs						PT
1	G8/M3	G6/M5	G5/M1	G4/M4	G3/M2	G26
2	G7/M3	G5/M4	G4/M1	G3/M2	G1/M5	G20
3	G7/M3	G6/M2	G5/M1	G3/M5	G2/M4	G23

Tabela 4.2 Aplicação da regra LPT nas linhas.

Jobs						PT
1	G8/M3	G6/M5	G5/M1	G4/M4	G3/M2	G26
2	G7/M3	G6/M2	G5/M1	G3/M5	G2/M4	G23
3	G7/M3	G5/M4	G4/M1	G3/M2	G1/M5	G20

Tabela 4.3 Aplicação da regra LPT na soma dos tempos de processamento.

$$J = \begin{pmatrix} \text{peça } 1 \\ \text{peça } 3 \\ \text{peça } 2 \end{pmatrix}; M = \begin{pmatrix} 3 & 5 & 1 & 4 & 2 \\ 3 & 2 & 1 & 5 & 4 \\ 3 & 4 & 1 & 2 & 5 \end{pmatrix}$$

$$P = \begin{pmatrix} G8 & G6 & G5 & G4 & G3 \\ G7 & G6 & G5 & G3 & G2 \\ G7 & G5 & G4 & G3 & G1 \end{pmatrix};$$

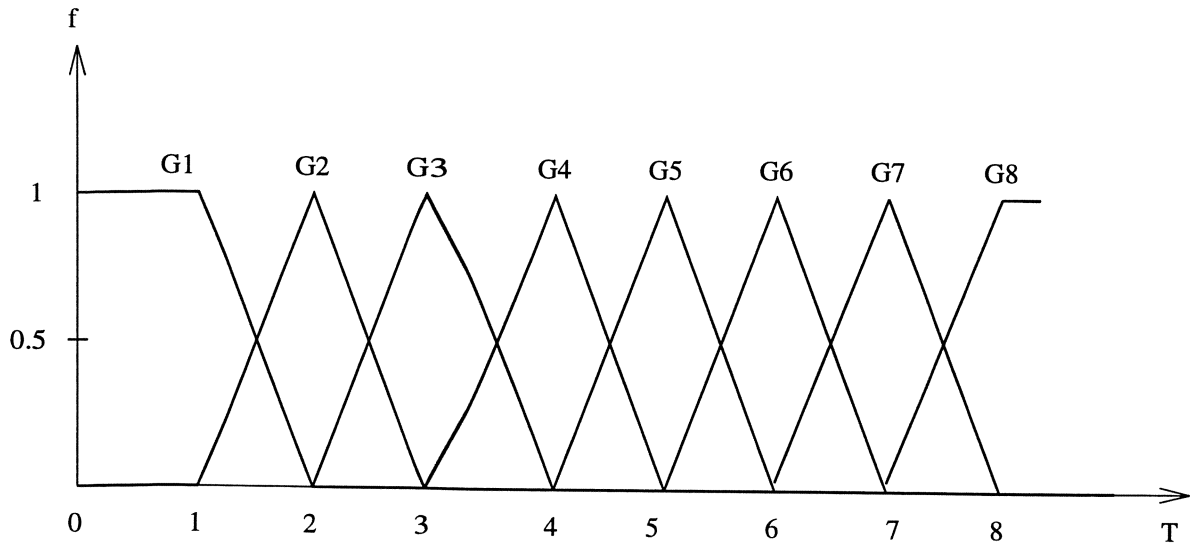


Figura 16: **Funções de pertinência das variáveis linguísticas do problema.**

Passo 4: Usando a base de regras (Tabela 3.1 (a) e Figura 16), obtemos uma programação horária (usando a regra LPT) mas ela têm sobreposições, portanto este schedule do robô é provisório.

Solução obtida:

Jobs						PT	Fluxo de tempo
1	G8/M3	G6/M5	G5/M1	G4/M4	G3/M2	G26	G31
3	G6/M2	G5/M1	G2/M4	G3/M5	G7/M3	G23	G29
2	G5/M4	G7/M3	G3/M2	G4/M1	G1/M5	G20	G30

Tabela 4.4 Aplicação da regra LPT (tempo de transporte incluso)

$$M = \begin{pmatrix} 3 & 5 & 1 & 4 & 2 \\ 2 & 1 & 4 & 5 & 3 \\ 4 & 3 & 2 & 1 & 5 \end{pmatrix} P = \begin{pmatrix} G8 & G6 & G5 & G4 & G3 \\ G6 & G5 & G2 & G3 & G7 \\ G5 & G7 & G3 & G4 & G1 \end{pmatrix};$$

Passos 5, 6 e 7:

Com as informações do passo anterior, determinamos a matriz A com os tempos finais dos processos das peças nas máquinas.

$$A = \begin{pmatrix} G9 & G16 & G22 & G27 & G31 \\ G8 & G14 & G17 & G21 & G29 \\ G8 & G18 & G22 & G28 & G30 \end{pmatrix}$$

Portanto, a seqüência RT a ser seguida pelo robô que minimiza o makespan é:

$$\begin{aligned} RT = & ((1,3),(3,2),(2,4),(1,5),(3,1),(2,3),(1,1),(3,4) \\ & (2,2),(1,4),(3,5),(2,1),(1,2),(3,3),(2,5)] \end{aligned}$$

onde (i,j) representa peça i na máquina j.

Passo 8: Programação horária resultante no gráfico de Gantt (Figura 17)

4.2 Aplicação do algoritmo 3.2

No mesmo problema do item anterior, vamos aplicar o Algoritmo 3.2. A programação horária obtida para este caso, encontra-se tabulada no quadro que segue, onde à direita dos jobs encontra-se a programação das máquinas nos diferentes

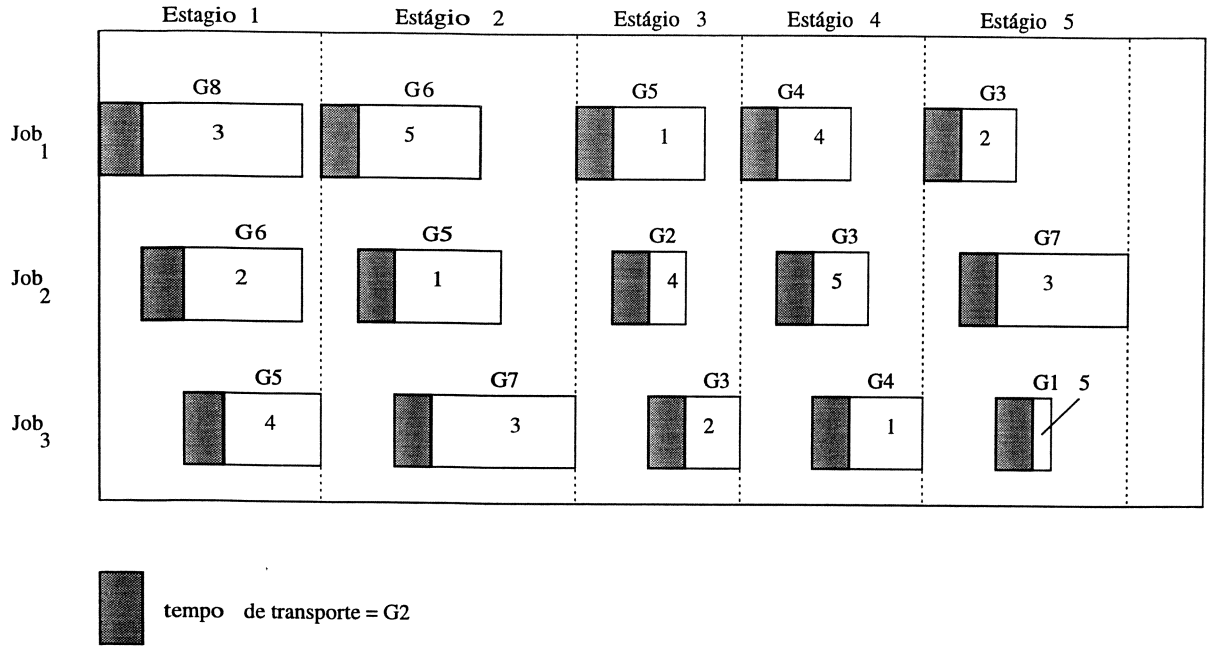


Figura 17: **Gráfico de Gantt da programação horária do exemplo.**

estágios.

Job 1	$G8/M_3$	$G6/M_5$	$G5/M_1$	$G4/M_4$	$G3/M_2$
Job 2	$G5/M_1$ $G2/M_4$	$G7/M_3$	$G6/M_2$	$G3/M_5$	
Job 3	$G3/M_2$ $G1/M_5$	$G5/M_4$	$G7/M_3$	$G4/M_1$	

Os instantes de tempos finais dos jobs em cada estágio estão calculados a seguir, onde as somas dos valores liguísticos foram calculados usando a tabela 3.1 (b).

$$\text{Peça 1} = \begin{cases} G1 + G8 = G9 \\ G1 + G9 + G6 = G16 \\ G1 + G9 + G5 = G22 \\ G1 + G22 + G4 = G27 \\ G1 + G27 + G3 = G31 \end{cases}$$

$$\text{Peça 2} = \begin{cases} G2 + G5 + G1 + G2 = G10 \\ G1 + G10 + G7 = G18 \\ G1 + G18 + G6 = G25 \\ G1 + G25 + G3 = G29 \end{cases}$$

$$\text{Peça 3} = \begin{cases} G3 + G3 + G1 + G1 = G8 \\ G1 + G8 + G5 = G14 \\ G1 + G14 + G7 = G22 \\ G1 + G22 + G4 = G27 \end{cases}$$

Os resultados obtidos estão na figura 18, em comparação com os resultados do algoritmo anterior.

4.3 Aplicação do algoritmo 3.3

A seguir, vamos resolver o nosso exemplo, aplicando o algoritmo 3.3 baseado na programação linear fuzzy, apresentada no capítulo 2 e utilizando o algoritmo 3.3 sugerido.

$$\begin{array}{rcll} y_{ij} - x_{ij} & \lesssim & -t_j & \forall i, j \\ y_{k,j-1} - y_{ij} & \lesssim & -P_{i,j-1} & \forall i, k,\ \forall j > 1 \\ x_{i,j-1} - y_{ij} & \lesssim & -P_{i,j-1} & \forall i, \forall j > 1 \\ x_{i-1,j} - x_{ij} & \lesssim & -t_j & \forall i > 1, \forall j \\ x_{k,j-1} - x_{1,j} & \lesssim & -t_j & \forall j > 1, \forall k \end{array}$$

$u \in I\!R^{33}$ com $u_1 = y_{11}, u_2 = x_{11}, u_3 = y_{12}, u_4 = x_{12}, u_5 = y_{13}, u_6 = x_{13}, u_7 = y_{14}, u_8 = x_{14}, u_9 = y_{15}, u_{10} = x_{15}, u_{11} = y_{16}, u_{12} = y_{21}, u_{13} = x_{21}, u_{14} = y_{22}, u_{15} =$

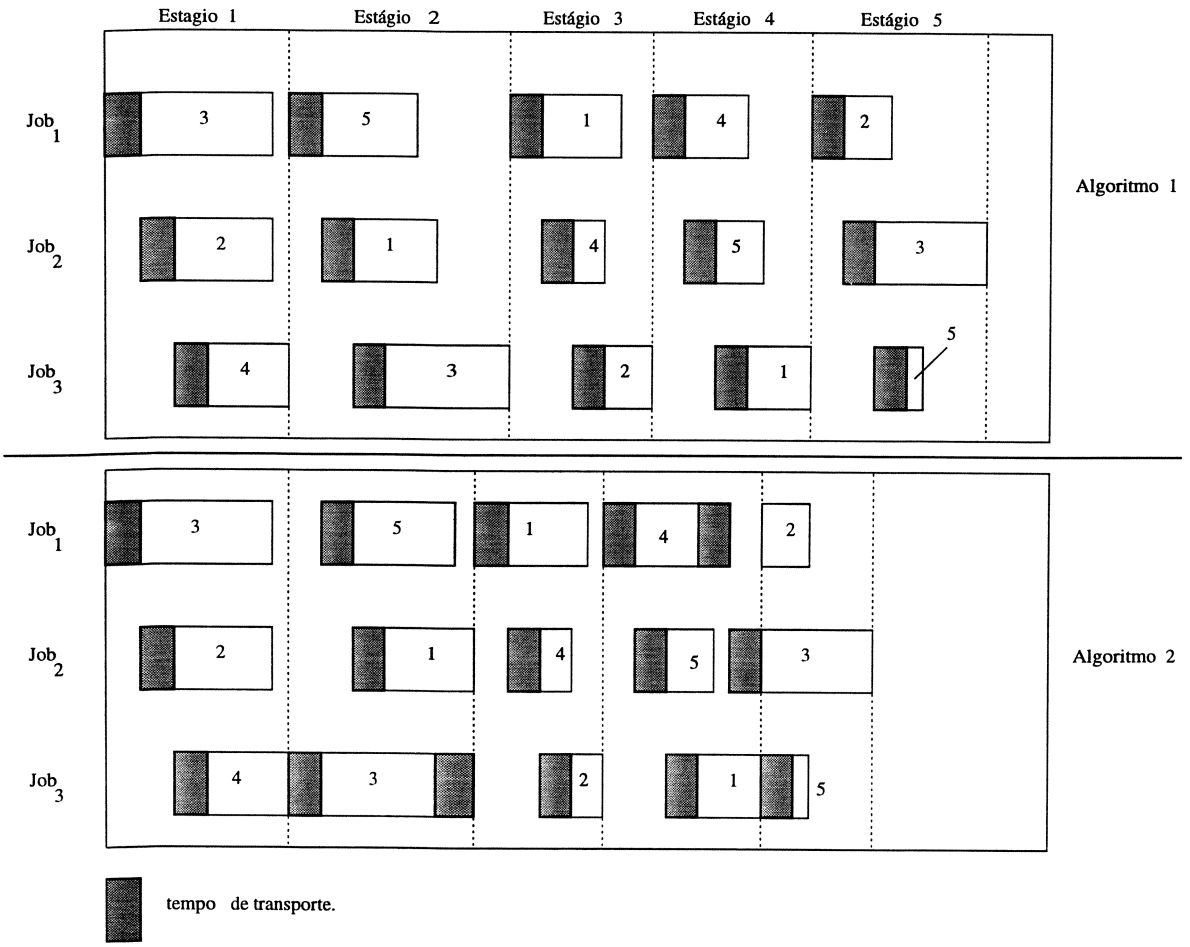


Figura 18: **Gráfico de Gantt da programação horária do exemplo comparando os algoritmos 3.1 e 3.2.**

$x_{22}, u_{16} = y_{23}, u_{17} = x_{23}, u_{18} = y_{24}, u_{19} = x_{24}, u_{20} = y_{25}, u_{21} = x_{25}, u_{22} = y_{31}, u_{23} = x_{32}, u_{24} = y_{32}, u_{25} = x_{32}, u_{26} = y_{33}, u_{27} = x_{33}, u_{28} = y_{34}, u_{29} = x_{34}, u_{30} = y_{35}, u_{31} = x_{35}, u_{32} = y_{26}, u_{33} = y_{36}$

$b \in I\!R^{42}, A \in I\!R^{42\times 33}$

Vamos agora, iniciar o processo indicado pela heurística para obter uma solução satisfatória do problema de scheduling de máquinas e jobs de modo a minimizar o makespan.

Primeiro resolvemos o problema de programação linear paramétrico usando o Simplex obtendo o vetor solução:

$S = (y_{11} = 0, x_{11} = 1.00, y_{12} = 9.00, x_{12} = 10.00, y_{13} = 16.00, x_{13} = 17.00, y_{14} = 22.00, x_{14} = 23.00, y_{15} = 27.00, x_{15} = 28.00, y_{16} = 31.00, y_{21} = 0.00, x_{21} = 2.00, y_{22} = 8.00, x_{22} = 10.00, y_{23} = 15.00, x_{23} = 16.00, y_{24} = 18, x_{24} = 19, y_{25} = 22, x_{25} = 23, y_{26} = 30, y_{31} = 0.00, x_{31} = 3.00, y_{32} = 8.00, x_{32} = 11.00, y_{33} = 18.00, x_{33} = 19.00, y_{34} = 22.00, x_{34} = 23.00, y_{35} = 27.00, x_{35} = 28.00, y_{36} = 29.00)$ com solução ótima $Z^* = 31,00$. Segundo, resolvemos o problema linear nebuloso considerando as restrições fuzzy com suas tolerâncias máximas α_i, β_{ij} para t_i e $P_{ij}, i = 1,2,3; j = 1,2,3,4,5$

Min Z

s.a. (1) $x_{ij} - y_{ij} \geq t_j - \theta\alpha_j;\ i = 1,2,3;\ j = 1,\ldots,5$

(2) $y_{ij} - y_{k,j-1} \geq P_{i,j-1} - \theta(\beta_{i,j-1});\ i,k = 1,2,3;\ j = 2,3,4,5$

(3) $y_{ij} - x_{i,j-1} \geq P_{i,j-1} - \theta\beta_{i,j-1};\ i = 1,2,3;\ j = 2,3,4,5,6$

(4) $x_{ij} - x_{i-1,j} \geq t_j - \theta\alpha_j;\ i = 2,3; j = 1,\ldots,5,$

(5) $x_{1j} - x_{k,j-1} \geq t_j - \theta\alpha_j;\ j = 2,\ldots,5, k = 1,\ldots,3$

$x_{ij} \geq 0,\ \theta \in [0,1]$

θ	y_{11}	x_{11}	y_{12}	x_{12}	y_{13}	x_{13}
0	0	1.0000	9.0000	10.0000	16.0000	17.0000
0.1000	0.0000	0.9500	8.6500	9.6000	15.4000	16.3500
0.2000	-0.0000	0.9000	8.3000	9.2000	14.8000	15.7000
0.3000	-0.0000	0.8500	7.9500	8.8000	14.2000	15.0500
0.4000	-0.0000	0.8000	7.6000	8.4000	13.6000	14.4000
⋮						
0.9000	0.0000	0.5500	5.8500	6.4000	10.6000	11.1500
1.0000	0	0.5000	5.5000	6.0000	10.0000	10.5000

y_{14}	x_{14}	y_{15}	x_{15}	y_{16}	y_{21}	x_{21}
22.0000	23.0000	27.0000	28.0000	31.0000	0	2.0000
21.1500	22.1000	25.9000	26.8500	29.7500	0	1.9200
20.3000	21.2000	24.8000	25.7000	28.5000	0	1.8000
19.4500	20.3000	23.7000	24.5500	27.2500	0	1.700
18.6000	19.4000	22.6000	23.4000	26.0000	0	1.6000
⋮						
14.3500	14.0000	16.0000	16.5000	18.5000	0	1.1000
13.5000	14.0000	16.0000	16.5000	18.5000	0	1.0000

y_{22}	x_{22}	y_{23}	x_{23}	y_{24}	x_{24}	y_{25}
8.0000	11.0000	16.0000	17.0000	19.0000	20.0000	23.0000
7.7000	10.5500	15.3500	16.3000	17.3500	18.3000	20.3500
7.4000	10.1000	14.7000	15.6000	16.7000	17.6000	19.7000
7.1000	9.6500	14.0500	14.9000	16.0500	16.9000	19.0500
6.8000	9.2000	13.4000	14.2000	15.4000	16.2000	18.4000
⋮						
5.3000	6.9500	10.1500	10.7000	12.1500	12.7000	15.1500
5.0000	6.5000	9.5000	10.0000	11.5000	12.0000	14.5000

x_{25}	y_{26}	y_{31}	x_{31}	y_{31}	x_{32}	y_{33}
24.0000	31.0000	0	3.0000	8.0000	9.0000	16.0000
21.3000	28.0000	0	2.8500	7.6600	8.6100	15.3100
20.6000	27.0000	0	2.7000	7.3000	8.2000	14.6000
19.9000	26.0000	0	2.5500	6.9500	7.7000	13.8000
19.2000	25.0000	0	2.4000	6.6000	7.4000	13.2000
⋮						
15.7000	20.0000	0	1.6500	4.8500	5.4000	9.7000
15.0000	19.0000	0	1.5000	4.5000	5.0000	9.0000

x_{33}	y_{34}	x_{34}	y_{35}	x_{36}	y_{36}
17.0000	20.0000	21.0000	25.0000	26.0000	27.0000
16.2600	19.1600	20.1100	23.9100	24.8600	25.8100
15.5000	18.3000	19.2000	22.8000	23.7000	24.6000
14.6500	17.3500	18.2000	21.6000	22.4500	22.3000
14.0000	16.6000	17.4000	20.6000	21.4000	22.2000
⋮					
10.2500	12.3500	12.9000	15.1000	15.6500	15.2000
10.0000	12.0000	12.5000	14.5000	15.0000	14.5000

Tabela 4.5 Solução paramétrica do problema .

	y_{11}	x_{11}	y_{12}	x_{12}	y_{13}	x_{13}
	0.0000	0.8000	7.6000	8.4000	13.6000	14.4000
y_{14}	x_{14}	y_{15}	x_{15}	y_{16}	y_{21}	x_{21}
18.6000	19.4000	22.6000	23.4000	26.0000	0	1.6000
y_{22}	x_{22}	y_{23}	x_{23}	y_{24}	x_{24}	y_{25}
6.8000	9.2000	13.4000	14.2000	15.4000	16.2000	18.4000
x_{25}	y_{26}	y_{31}	x_{31}	y_{31}	x_{32}	y_{33}
19.2000	25.0000	0	2.4000	6.6000	7.4000	13.2000
x_{33}	y_{34}	x_{34}	y_{35}	x_{36}	y_{36}	
14.0000	16.6000	17.4000	20.6000	21.4000	22.2000	

Tabela 4.6: Solução utilizando modelo de Werners.

A seguir a tabela 4.4 mostra a solução ótima para diferentes valores do parâmetro θ e o valor das variáveis.

Terceiro, vamos resolver o problema usando o modelo obtido aplicando a proposta de Werners (1987). Precisamos os valores de W^1, W^0. Estes valores correspondem aos valores máximo e mínimo da solução do problema paramêtrico dada pela tabela 4.5. Então temos: $W^0 = 21.50$, $W^1 = 31.00$. Com estes valores resolvemos o problema:

$$\begin{array}{ll} \text{Min } \theta & \\ \text{s.a.} & (1) \quad Z \leq W^0 + \theta(W^1 - W^0) \\ & (2) \quad \text{restrições } (3.12). \\ & x \geq 0, \theta \in [0,1]. \end{array}$$

onde as tolerâncias máximas dos elementos do vetor do lado direito das restrições vão ser consideradas como sendo as mesmas do problema paramêtrico já resolvido.

A solução ótima fornece os valores, $W^* = 26.60$ em $\theta = 0.4$ com $\mu(W^*) = 0.6$ e o valor das variáveis podem ser vistos na tabela 4.5.

Quarto, vamos resolver o problema usando o modelo obtido pela aplicação do método de Zimmermann(1984). Para isto precisamos conhecer a meta da função objetivo b_0 e sua tolerância máxima γ_0 . Estes valores podem ser escolhidos pelo planejador a partir da tabela 4.5, que fornece valores bastante aproximados da solução. Suponhamos que escolhemos $b_0 = 27.25$ em $\theta = 0.3$ da tabela 4.4. con tolerância máxima $\gamma_0 = 3$. Com estes valores podemos resolver o problema:

$$\begin{array}{ll} \text{Min } \theta & \\ \text{s.a.} & (1) \quad Z \leq b_0 + \theta\gamma_0 \\ & (2) \quad \text{restrições do problema (3.14)} \\ & x \geq 0, \theta \in [0,1]. \end{array}$$

A solução ótima do problema obtida aplicando o método de Zimmermann fornece $Z^* = 28.17$ em $\theta = 0.3$ com $\mu_0(Z^*) = 0.7$; $Z^* = 28.17$ em $\theta = 0.3$ e com

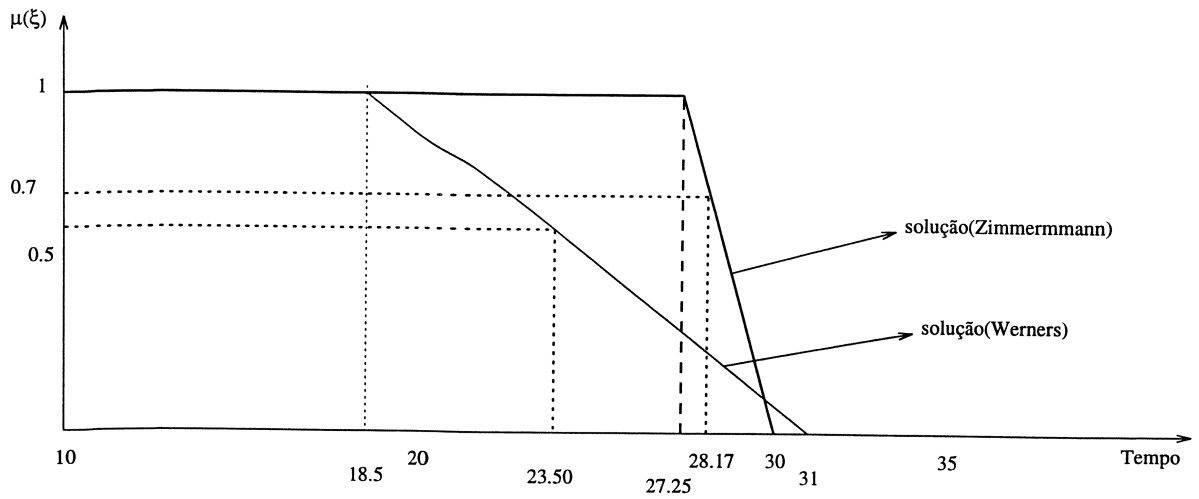


Figura 19: **Comparação das soluções obtidas pela aplicação dos métodos de Werners e Zimmermann.**

p_0=3 e os valores das variáveis estão colocados na tabela 4.6.

	y_{11}	x_{11}	y_{12}	x_{12}	y_{13}	x_{13}
	0.0000	0.9000	8.3000	9.2000	14.8000	15.7000
y_{14}	x_{14}	y_{15}	x_{15}	y_{16}	y_{21}	x_{21}
20.3000	21.2000	24.8000	25.7000	28.0700	0	1.8000
y_{22}	x_{22}	y_{23}	x_{23}	y_{24}	x_{24}	y_{25}
7.4000	10.1000	14.7000	15.6000	16.7000	17.6000	19.7000
x_{25}	y_{26}	y_{31}	x_{31}	y_{31}	x_{32}	y_{33}
20.6000	27.0000	0	2.7000	7.3000	8.2000	14.6000
x_{33}	y_{34}	x_{34}	y_{35}	x_{36}	y_{36}	
15.5000	18.3000	19.2000	22.8000	23.7000	24.6000	

Tabela 4.7: Solução utilizando método de Zimmermann.

4.4 Aplicação do procedimento 3.5

Como foi colocado, os métodos de ordenamento dos números nebulosos aplicados aos problemas de programação linear fuzzy, produzem problemas auxiliares,

cujas soluções são aproximações da solução do problema original. Vamos estabelecer estes problemas auxiliares a partir do modelo 1 definido no capítulo 2 e 3 baseado no problema 4.1 a seguir.

$$
\begin{array}{ll}
\text{Min} & Z \\
\text{sa.} & y_{i,j} - x_{ij} \lesssim -\tilde{t}_j \qquad \forall i, \forall j > 1 \\
& y_{k,j-1} - y_{ij} \lesssim -\tilde{P}_{i,j-1} \qquad \forall i, k, \forall j > 1 \\
& x_{i,j-1} - y_{ij} \lesssim -\tilde{P}_{i,j-1} \qquad \forall i, \forall j > 1 \\
& x_{i-1,j} - x_{ij} \lesssim -\tilde{t}_j \qquad \forall i > 1, \forall j \\
& x_{k,j-1} - x_{1j} \lesssim -\tilde{t}_j \qquad \forall k, \forall j > 1 \\
x \geq 0, i \in I\!N_n \; j \in I\!N_m &
\end{array}
\tag{4.1}
$$

onde os coeficientes nebulosos são definidos por:

$$
\begin{aligned}
\tilde{t}_j &= [t_j, t_1 - \alpha_j, t_j + c_j], j \in I\!N_m. \\
\tilde{P}_{i,j-1} &= [P_{i,j-1}, P_{i,j-1} - \beta_{i,j-1}, P_{i,j-1} + d_{i,j-1}], i \in I\!N_n.
\end{aligned}
\tag{4.2}
$$

e onde as restrições são nebulosas. O planejador permite violações na satisfação das cinco restrições em valores em torno de v_k, $k = 1, 2, 3, 4, 5$.
Vamos supor que $\tilde{v}_k = (v_k, v_k - s_k, v_k + r_k)$

Então (4.1) é equivalente ao problema:

$$
\begin{array}{ll}
\text{Min} & Z \\
\text{sa.} & g_1(x) \leq_P -\tilde{t}_j + \tilde{v}_1(1-\alpha) \\
& \\
& g_2(x) \leq_P -\tilde{P}_{i,j-1} + \tilde{v}_2(1-\alpha) \\
& \\
& g_3(x) \leq_P -\tilde{P}_{i,j-1} + \tilde{v}_3(1-\alpha)
\end{array}
$$

$$g_4(x) \leq_P \quad -\tilde{t}_j + \tilde{v}_4(1-\alpha) \tag{4.3}$$

$$g_5(x) \leq_P \quad -t_j + \tilde{v}_5(1-\alpha)$$
$$x \geq 0, \alpha \in (0,1]$$

Usando (3.23) vemos que os termos do lado direito das restrições são:

$$[-t_j, -t_j - \alpha_j, -t_j + c_j] + [v_k(1-\alpha), (v_k - s_k)(1-\alpha), (v_k + r_k)(1-\alpha)]$$
$$= [-t_j + v_k(1-\alpha), -t_j - \alpha_j + (v_k - s_k)(1-\alpha), -t_j - c_j + (v_k + r_k)(1-\alpha)]$$
$$k = 1, 4, 5$$

$k = 2$

$$-[P_{i,j-1}, P_{i,j-1} - \beta_{i,j-1}, P_{i,j-1} + d_{1,j-1}]$$
$$+[v_2, v_2 - s_2, v_2 + r_2](1 - \alpha) = [-P_{i,j-1} + v_2(1-\alpha)$$
$$+P_{i,j-1} - \beta_{i,j-1} + (v_2 - s_2)(1-\alpha), P_{i,j-1} + d_{1,j-1} + (v_2 + r_2)(1-\alpha)]$$

$k = 3$

$$[P_{i,j-1}, P_{i,j-1} - \beta_{i,j-1}, P_{i,j-1} + d_{i,j_1}] + [v_3, v_3 - s_3, v_3 + r_3](s-1)$$
$$= [P_{i,j-1} + v_3(s-1), P_{i,j-1} - \beta_{i,j-1} + (v_3 - s_3)(s-1), P_{i,j-1}$$
$$+d_{i,j-1} + (v_3 + r_3)(s-1)]$$

Portanto, considerando que os coeficientes da matriz tecnológica são todos iguais a um e denotando os lados esquerdos das restrições por $g_k(x)$, as formulações dos modelos auxiliares para resolver (4.1) são os seguintes:

1. (3.30) $\Rightarrow$

$$\text{Min } Z \tag{4.4}$$
$$\text{sa. } g_k(x) \leq -3tj - \alpha_j + c_j + (3v_k - s_k + r_k)(1-\alpha),\ k = 1, 4, 5,$$

$$g_2(x) \leq 3P_{i,j-1} + \beta_{i,j-1} - d_{1,j-1} + (3v_2 - s_2 + r_2)(1-\alpha)$$
$$g_3(x) \leq -3P_{i,j-1} + \beta_{i,j-1} - d_{i,j-1} + (3v_3 - s_3 + r_3)(1-\alpha) x \geq 0, \alpha \in (0,1]$$

2. (3.31) $\Rightarrow$

$$\text{Min } Z \tag{4.5}$$
$$\text{sa. } g_k(x) \leq -4tj + \alpha_j - c_j + (4v_k - s_k + r_k)(1-\alpha), k = 1,4,5$$
$$g_2(x) \leq -4P_{i,j-1} + \beta_{i,j-1} - d_{1,j-1} + (4v_2 - s_2 + r_2)(1-\alpha)$$
$$g_3(x) \leq -4P_{i,j-1} + \beta_{i,j-1} - d_{i,j-1} + (4v_3 - s_3 + r_3)(1-\alpha)$$
$$x \geq 0, \alpha \in (0,1]$$

3. (3.32) $\Rightarrow$

$$\text{Min } Z \tag{4.6}$$
$$\text{sa. } g_k(x) \leq (1-\omega)(-\overline{t}_j) + \omega(-t_j) + [(1-\omega)\overline{v}_k + \omega v_k](1-\alpha),\ k = 1,4,5$$
$$g_2(x) \leq (1-\omega)(-\overline{P}_{i,j-1}) + (-P_{i,j-1}) + [(1-\omega)\overline{v}_2 + \omega v_2](1-\alpha)$$
$$g_3(x) \leq (1-\omega)(-\overline{P}_{i,j-1}) + \omega(-P_{i,j-1}) + [(1-\omega)\overline{v}_3 + \omega v_3](1-\alpha)$$
$$x \geq 0, \alpha \in (0,1]$$

com ω, o grau de satisfação fixado pelo planejador, $\omega \in (0,1]$

4. (3.33) $\Rightarrow$

$$\text{Min } Z \tag{4.7}$$
$$\text{sa. } g_k(x) \leq (1-\omega)(-\overline{t}_j) + \omega(-t_j) + [(1-\omega)\overline{v}_k + \omega v_k](1-\alpha),\ k = 1,4,5$$
$$g_2(x) \leq (1-\omega)(-\overline{P}_{i,j-1}) + \omega(-P_{i,j-1}) + [(1-\omega)\overline{v}_2 + \omega v_2](1-\alpha)$$
$$g_3(x) \leq (1-\omega)(-\overline{P}_{i,j-1}) + \omega(-P_{i,j-1}) + [(1-\omega)\overline{v}_3 + \omega v_3](1-\alpha)$$
$$x \geq 0, \alpha \in (0,1]$$

com $\omega \in (0,1]$ (o grau de satisfação fixado pelo planejador)

pois a matriz tecnológica neste problema é sem incertezas.

5. (3.34) $\Rightarrow$

$$\begin{aligned}
&\text{Min } Z \\
&\text{sa. } g_k(x) \leq (1-\omega)(-t_j) + \omega(-t_j - \alpha_j) + [(1-\omega)v_k + \omega(v_k - s_3)](1-\alpha) \\
&\qquad k = 1,4,5
\end{aligned}$$

$$\begin{aligned}
&g_2(x) \leq (1-\omega)(-P_{i,j-1}) + \omega(-P_{i,j-1} - \beta_{i,j-1}) \\
&+[(1-\omega)v_2 + \omega(v_k - s_k)](1-\alpha) \\
&g_3(x) \leq (1-\omega)(-P_{i,j-1}) + \omega(-P_{i,j-1} - \beta_{i,j-1}) \\
&+[(1-\omega)v_3 + \omega(v_3 - s_3)](1-\alpha) \qquad (4.8) \\
&x \geq 0, \alpha \in (0,1]
\end{aligned}$$

e $\omega \in (0,1]$ (o grau fixado pelo planejador)

Desta forma, o problema 4.1 pode ser reformulado para:

$$\begin{aligned}
&\text{Min } Z \\
&\text{s.a.} \quad y_{ij} - x_{ij} \leq_p -\tilde{t}_j + t\tilde{t}_j(1-\alpha) && ;\forall i,j \\
&\qquad x_{i,j-1} - y_{ij} \leq_p -\tilde{P}_{i,j-1} + t\tilde{P}_{i,j-1}(1-\alpha) && ;\forall i, \forall j > 1 \\
&\qquad y_{k,j-1} - y_{ij} \leq_p -\tilde{P}_{i,j-1} + \\
&\qquad (-t\tilde{P}_{i,j-1})(1-\alpha) && ;\forall i,k, \forall j > 1 \\
&\qquad x_{i-1,j} - x_{ij} \leq_p -\tilde{t}_j + t\tilde{t}_j(1-\alpha) && ;\forall i > 1, \forall j \\
&\qquad x_{k,j-1} - x_{1j} \leq_p -\tilde{t}_j + t\tilde{t}_j(1-\alpha) && ;\forall k, \forall j > 1 \\
&\qquad x_{ij} \geq 0, y_{ij} \geq 0, \alpha \in (0,1).
\end{aligned}$$

Aqui $t\tilde{t}_j$ e $t\tilde{P}_{ij}$ correspondem às tolerâncias dos números $\tilde{t}_j$ e $\tilde{P}_{ij}$. As tolerâncias foram consideradas como números nebulosos simétricos, isto é: $\tilde{a} = [a, \underline{a} =$

$a - z, \overline{a} = a + d]$ com $d = z = 1$.

A forma explícita dos modelos nos cinco casos são definidos a seguir.

Modelo do Caso 1

Min Z

$$
\begin{aligned}
3y_{ij} - 3x_{ij} &\leq -(t_j + \overline{t}_j + \underline{t}_j) + [tt_j + t\overline{t}_j + t\underline{t}_j](1-\alpha); \forall i, j \\
3x_{i,j-1} - y_{ij} &\leq -(P_{i,j-1} + \overline{P}_{i,j-1} + \underline{P}_{i,j-1}) + \\
&\quad [tP_{i,j-1} + t\overline{P}_{i,j-1} + t\underline{P}_{i,j-1}](1-\alpha); \forall i, \forall j > 1 \\
3y_{k,j-1} - 3y_{ij} &\leq -(P_{i,j-1} + \overline{P}_{i,j-1} + \underline{P}_{i,j-1}) + \\
&\quad [tP_{i,j-1} + t\overline{P}_{i,j-1} + t\underline{P}_{i,j-1}](1-\alpha); \forall i, k, \forall j > 1 \\
3x_{i-1,j} - 3x_{ij} &\leq -(t_j + \overline{t}_j + \underline{t}_j) + [tt_j + t\overline{t}_j + t\underline{t}_j](1-\alpha); \forall i > 1, \forall j \\
3x_{k,j-1} - 3x_{1j} &\leq -(t_j + \overline{t}_j + \underline{t}_j) + [tt_j + t\overline{t}_j + t\underline{t}_j](1-\alpha); \forall k, \forall j > 1
\end{aligned}
$$

$x_{ij} \geq 0,\ y_{ij} \geq 0,\ \alpha \in (0, 1]$.

Aqui o coeficiente 3 das variáveis corresponde à especificação $(a_i + \overline{a}_i + \underline{a}_i)$ do modelo auxiliar do caso 1 pois temos considerado que $\tilde{1} = [1, \underline{1}, \overline{1}] = [1, 1-0.5, 1+0.5]$.

Modelo do Caso 2

Min Z

$$
\begin{aligned}
&4y_{ij} - 4x_{ij} \leq -(2t_j + \overline{t}_j + \underline{t}_j) + [2tt_j + t\overline{t}_j + t\underline{t}_j](1-\alpha); \forall i, j \\
&4x_{i,j-1} - 4y_{ij} \leq (-(2P_{i,j-1} + \overline{P}_{i,j-1} + \underline{P}_{i,j-1}) + \\
&\qquad [2tP_{i,j-1} + t\overline{P}_{i,j-1} + t\underline{P}_{i,j-1}](1-\alpha); \forall i, \forall j > 1 \\
&4y_{k,j-1} - 4y_{ij} \leq -(2P_{i,j-1} + \underline{P}_{i,j-1}) + \\
&\qquad [2tP_{i,j-1} + t\overline{P}_{i,j-1} + t\underline{P}_{i,j-1}](1-\alpha); \forall i, k,\ \forall j > 1 \\
&4x_{i-1,j} - 4x_{ij} \leq -(2t_j + \overline{t}_j + \underline{t}_j) + [2tt_j + t\overline{t}_j + t\underline{t}_j](1-\alpha); \forall i > j, \forall j \\
&4x_{k,j-1} - 4x_{1j} \leq -(2t_j + \overline{t}_j + \underline{t}_j) + [2tt_j + t\overline{t}_j + t\underline{t}_j](1-\alpha); \forall k, \forall j > 1
\end{aligned}
$$

$$x_{ij} \geq 0,\ y_{ij} \geq 0,\ \alpha \in (0,1].$$

Aqui os coeficientes 4 e 2 correspondem à especificação do modelo auxiliar do caso 2.

Modelo do Caso 3

Min Z

$$\begin{aligned}
a(y_{ij} - x_{ij}) &\leq -(kt_j - (1-k)\bar{j}_j) + (ktt_j + (1-k)t\bar{t}_j)(1-\alpha); \forall i,j \\
a(x_{i,j-1} - y_{ij}) &\leq k(-P_{i,j-1}) + (1-k)(-\overline{P}_{i,j-1}) + \\
&\quad [k(-tP_{i,j-1}) + (1-k)(-t\overline{P}_{i,j-1})](1-\alpha); \forall i, \forall j > 1 \\
a(y_{v,j-1} - y_{ij}) &\leq -(kP_{i,j-1} + (1-k)\overline{P}_{i,j-1}) + [ktP_{i,j-1} + (1-k)t\overline{P}_{i,j-1}](1-\alpha); \\
&\quad \forall i, v, \forall j > 1 \\
a(x_{i-1,j} - x_{ij}) &\leq -(kt_j + (1-k)\bar{t}_j) + [ktt_j + (1-k)t\bar{t}_j](1-\alpha); \forall i > 1, \forall j \\
a(x_{v,j-1} - x_{1j}) &\leq -(kt_j + (1-k)\bar{t}_j) + [ktt_j + (1-k)t\bar{t}_j](1-\alpha); \forall v, \forall j > 1
\end{aligned}$$

$$x_{ij} \geq 0,\ y_{ij} \geq 0,\ k,\ \alpha \in (0,1].$$

onde o coeficiente $a = 1.5 - 0.5k$.

Modelo do Caso 4

O modelo do caso 4 é similar ao modelo do caso 3 com $a = 0.5(1+k)$.

Modelo do Caso 5

Min Z

$$a(y_{ij} - x_{ij}) \leq -((1-k)t_j - k\underline{t}_j) + [(1-k)tt_j + kt\underline{t}_j](1-\alpha); \forall i,j$$

$$
\begin{aligned}
a(x_{i,j-1} - y_{ij}) &\leq (1-k)(-P_{i,j-1}) + k(-\underline{P}i, j-1) \\
&\quad [(1-k)(-tP_{i,j-1}) + k(-t\underline{P}_{i,j-1})](1-\alpha); \forall i, \forall j > 1 \\
a(y_{v,j-1} - y_{ij}) &\leq -((1-k)P_{i,j-1} + k\underline{P}_{i,j-1}) + [(1-k)tP_{i,j-1} + kt\underline{P}_{i,j-1}](1-\alpha); \\
&\quad \forall i, v, \forall j > 1 \\
a(x_{i-1,j} - x_{ij}) &\leq -((1-k)t_j + k\underline{t}_j) + [(1-k)tt_j + kt\underline{t}_j](1-\alpha); \forall i > 1, \forall j \\
a(x_{v,j-1} - x_{1j}) &\leq -((1-k)t_j + k\underline{t}_j) + [(1-k)tt_j + kt\underline{t}_j](1-\alpha); \forall v, \forall j > 1
\end{aligned}
$$

$$x_{ij} \geq 0,\ y_{ij} \geq 0, k,\ \ \alpha \in (0,1].$$

onde o coeficiente $a = 1.5 - 0.5k$

Os resultados de aplicação destes modelos auxiliares ao problema de programação horária em células flexíveis de manufatura são dados na Tabela 4.6, onde o grau k de certeza das tolerâncias foi fixado em 0.8 para os modelos indicados como caso 3, caso 4, caso 5, e a pertinência dos resultados é $\mu(x)$=0.8 para todos as soluções de todos os casos.

Na primeira coluna estão as variáveis do problema, onde y_{16}, y_{26}, y_{36}, indicam os tempos finais do processamento das peças 1,3 e 2, respectivamente. Esta ordem é devido o tempo de processamento das peças ser considerado na ordem não crescente, processando primeiro a peça com maior tempo total de processamento. Vamos assumir $\mu(x) = 0.8$ e $k = 0.8$.

Os resultados são relativamente similares salvo os casos 4 e 5 correspondentes ao método de ordenamento de Dubois e Prade onde no caso 4 temos um valor para y_{36}, por exemplo, maior que todos os outros casos e no caso 5 temos um valor para y_{36} menor que todos os outros casos. Isto é devido no caso 4, o método considerar apenas os limites superiores dos números nebulosos e no caso 5, considerar somente os limites inferiores dos números nebulosos.

No gráfico de Gantt da Figura 20, pode-se comparar as soluções correspondentes à programação horária da peça 1 nos seguintes casos: a primeira programação horária corresponde à solução do problema sem incertezas; a segunda programação horária corresponde ao método de Werners para $\theta = 0.5$, isto é ,com

valor de pertinência $\mu = 0.5$; a terceira programação horária corresponde ao método de Zimmermann com $\theta = 0.2$, isto é, com valor de pertinência $\mu = 0.8$; e a quarta programação horária corresponde ao caso 2 do método de ordenamento dos números nebulosos (segundo índice de Yager) com valor de pertinência 0.8 .

Programação horária da peça 1

Variável	Caso 1	Caso 2	Caso 3	Caso 4	Caso 5
y_{11}	0	0	0	0	0
x_{11}	0.9500	0.9000	0.9636	1.1778	0.2889
y_{12}	8.5000	8.3500	8.0182	9.8000	6.6889
x_{12}	9.5523	9.2500	8.9818	10.9778	6.9778
y_{13}	14.8000	14.9000	14.2182	17.3778	12.2667
x_{13}	15.800	15.8000	15.1818	18.5556	12.5556
y_{14}	20.457	20.4500	19.5091	23.8444	16.7333
x_{14}	21.400	21.3500	20.4727	25.0222	17.0222
y_{15}	25.400	25.2000	24.0727	29.4222	20.3111
x_{15}	26.500	26.1000	25.0364	30.6000	20.6000
y_{16}	28.800	28.7000	27.5455	33.6667	22.7778

Programação horária da peça 3

Variável	Caso 1	Caso 2	Caso 3	Caso 4	Caso 5
y_{21}	0	0	0	0	0
x_{21}	1.9000	1.8000	1.9272	2.3556	0.5778
y_{22}	7.1500	7.4500	7.1636	8.7556	5.8667
x_{22}	10.5023	10.1500	9.9454	12.1556	7.2667
y_{23}	15.4023	14.8000	14.2727	17.4445	11.4445
x_{23}	16.3523	15.7000	15.2363	18.6223	11.7334
y_{24}	18.1523	17.5000	17.0181	20.8000	13.0223
x_{24}	19.1023	18.4000	17.9817	21.9778	13.3112
y_{25}	21.8415	21.2000	20.6726	25.2667	15.7112
x_{25}	22.7915	22.1000	21.6362	26.4445	16.0001
y_{26}	28.9815	28.5500	27.7817	33.9556	21.2890

Programação horária da peça 2

Variável	Caso 1	Caso 2	Caso 3	Caso 4	Caso 5
y_{31}	0	0	0	0	0
x_{31}	2.8500	2.7000	2.8908	3.5334	0.8667
y_{32}	7.3500	7.3500	7.2181	8.8223	5.0445
x_{32}	8.0000	8.2500	8.1817	10.0000	5.5778
y_{33}	14.7000	14.7000	14.3272	17.5112	10.8667
x_{33}	15.6000	15.6000	15.2908	18.6889	11.1555
y_{34}	21.6000	21.5500	21.2545	23.7556	18.6445
x_{34}	22.6000	22.4500	22.2181	24.9334	18.9334
y_{35}	26.5000	26.5000	25.9999	29.5556	22.4445
x_{35}	27.5000	27.4000	26.9636	30.7334	22.7334
y_{36}	28.5000	28.3000	27.8363	31.8000	23.4667

Tabela 4.8 Solução do problema (métodos de ordenamento)

O gráfico de Gantt da Figura 21, compara a solução obtida com os dados considerados sem incertezas, com os casos 3 e 4 do método de ordenamento de números nebulosos (métodos de Dubois e Prade). O valor k indica o grau de credibilidade do planejador com respeito às tolerâncias dos coeficientes do problema.

4.5 Alguns resultados de avaliação.

Nos quadros que seguem apresentam alguns resultados da aplicação dos algoritmos 3.1, 3.2, 3.3, 3.4 e 3.5. Este último corresponde ao método de ordenamento de Dubois e Prade para os limites inferiores (a) e limites superiores (b).

Escolhemos 3 exemplos com complexidades distintas: o primeiro com 3 peças e 5 máquinas, o segundo com 5 peças e 5 máquinas e o terceiro com 2 peças e 4 máquinas. Para cada exemplo, criamos 4 cenários com o objetivo de testar os algoritmos em diferentes situações, a saber: tempo de processamento das peças similares entre si, tempo de processamento similares sòmente nas mesmas máquinas

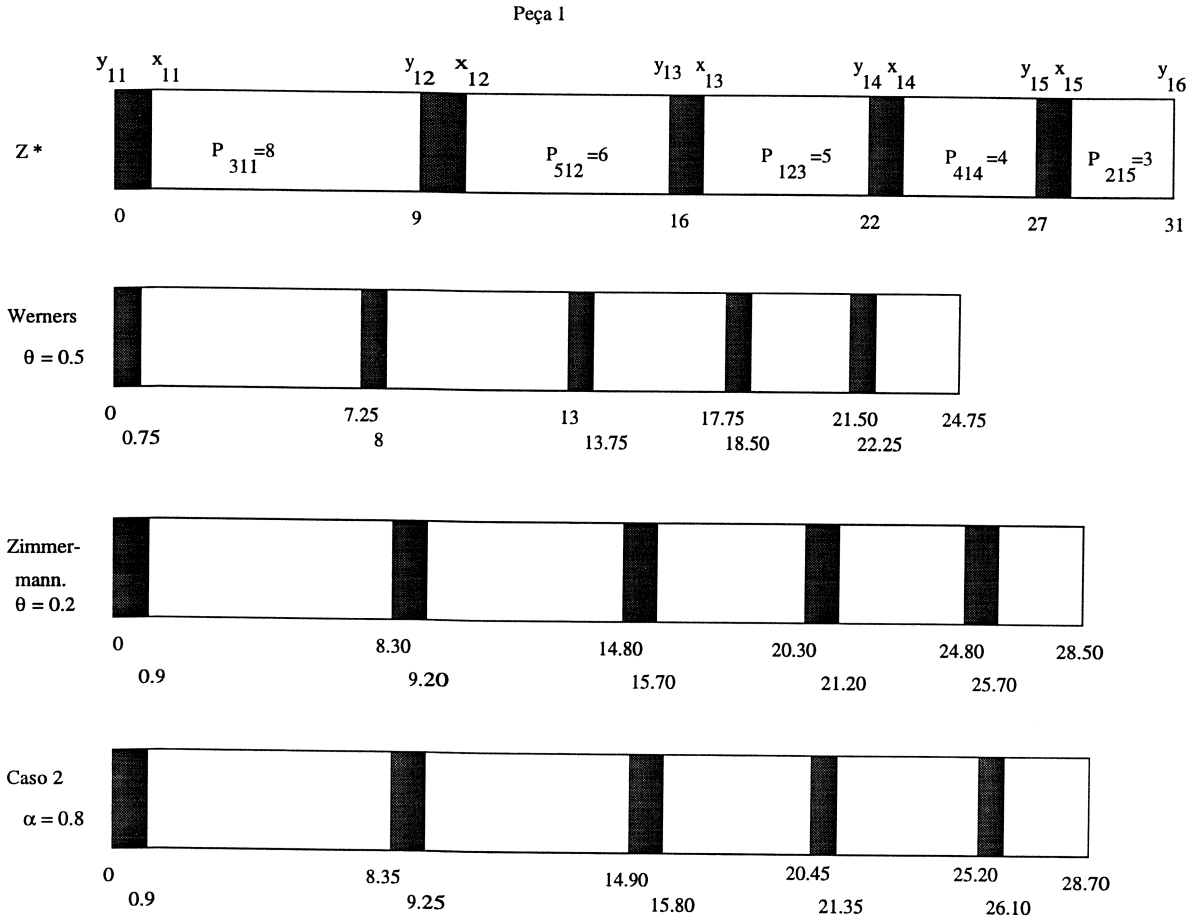


Figura 20: **Gráfico de Gantt da programação horária da peça 1 (caso crisp, Werners, Zimmermann e caso 2).**

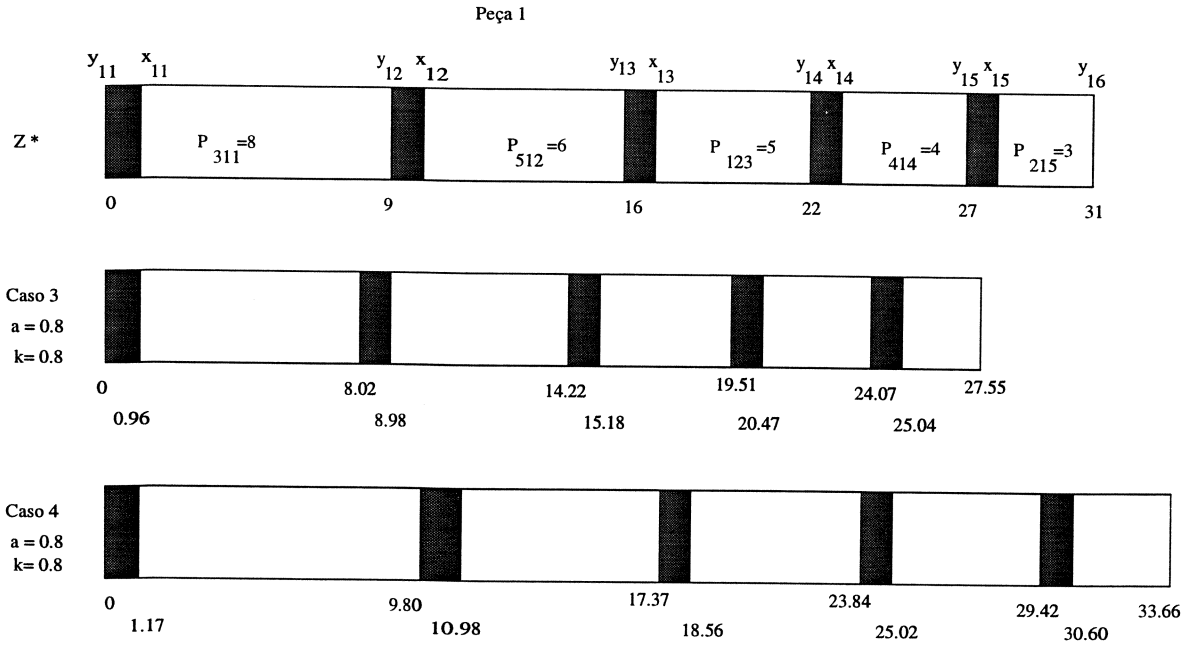


Figura 21: **Gráfico de Gantt da programação horária da peça 1 (casos 3 e 4).**

e distintas entre máquinas, tempos de processamento bastante distintos entre peças na mesma máquina e tempo de processamento de uma das peça bem maiores que das outras. Os tempos de transporte, por simplicidade, foram considerados iguais a um. Os resultados estão expostos nas tabelas 4.7, 4.8 e 4.9, respectivamente para os exemplos 1, 2 e 3.

Os tempos de processamento dos algoritmos para resolver os problemas foram determinados aproximadamente, utilizando uma estação de trabalho Sparc 1+ da SUN Microsystems e rodando os programas em MATLAB.

Os valores linguísticos G41 e G34 no primeiro problema representam a soma dos tempos de transporte e processamento das peças e os tempos ociosos que ocorrem nos diferentes estágios, todos expressos em números nebulosos. O mesmo vale para os outros valores linguísticos que aparecem na tabela. Z^* expressa a solução ótima do problema sem incertezas obtida resolvendo o problema formulado como de

programação inteira mixta.

Problema	Algoritmo	Processamento de aproximado (segundos)	Makespan (minutos)	Observações
	3.1	4	G41	
7/1 6/2 8/3 7/4 8/5	3.2	4	G34	Neste problema os
6/1 6/2 6/3 5/4 5/5	3.3	8	48.78	tempos de processa-
4/1 4/2 7/3 5/4 4/5	3.4	9	41.40	mento das peças
	3.5	12	a) 38.50	são todos similares.
			b) 45.30	$Z^* = 41$
	3.1	4	G29	
2/1 2/2 10/3 5/4 4/5	3.2	5	G28	Aqui os tempos de
1/1 3/2 9/3 3/4 5/5	3.3	7	31.40	processamento das
2/1 1/2 5/3 5/4 4/5	3.4	9	29.35	peças são similares
	3.5	14	a) 27.25	nas mesmas máquinas
			b) 34.50	$Z^* = 29$
	3.1	5	G54	
5/1 18/2 9/3 14/4 2/5	3.2	6	G52	Aqui os tempos de
17/1 1/2 14/3 5/4 8/5	3.3	10	59.70	processamento das
1/1 13/2 4/3 8/4 17/5	3.4	15	54.90	peças são distintos
	3.5	17	a) 50.8	nas mesmas máquinas
			b) 57.30	$Z^* = 54$
	3.1	4	G51	
8/1 10/2 12/3 9/4 7/5	3.2	5	G48	Aqui os tempos de
5/1 1/2 2/3 5/4 2/5	3.3	9	56.30	processamento da
3/1 1/2 2/3 1/4 4/5	3.4	10	51.40	primeira peça
	3.5	13	a) 50.80	são maiores que as
			b) 60.60	outras. $Z^* = 51$

Tabela 4.9: Exemplo com 3 peças e 5 máquinas.

4.6 Comentários.

O algoritmo 3.1 teve um comportamento muito bom quanto à rapidez e facilidade em fornecer resultados. Ele cumpriu perfeitamente nossos objetivos de aplicar a teoria de conjuntos nebulosos a um problema de programação horária de

peças com incertezas em células flexíveis de manufatura. O conceito de estágio aqui utilizado foi fundamental para permitir o nosso propósito e se adequou perfeitamente. O algoritmo 3.1 forneceu um resultado pior se comparado com o algoritmo 3.2 como esperado, pois este segundo algoritmo que foi criado com o objetivo de explorar melhor as alternativas de programação, baseou-se no conceito de janelas de tempo. No entanto, também este algoritmo mostrou-se muito eficiente quanto à rapidez em fornecer resultados.

Problema	Algoritmo	Processamento aproximado (segundos)	Makespan (minutos)	Observações
8/1 9/2 9/3 8/4 7/5	3.1	10	G47	
8/1 7/2 7/3 8/4 8/5	3.2	12	G46	$Z^* = 47$
6/1 7/2 7/3 7/4 8/5	3.3	28	50.80	
6/1 6/2 7/3 5/4 5/5	3.4	30	47.30	
5/1 4/2 5/3 6/4 6/5	3.5	45	a) 40.40	
			b) 52.30	
3/1 15/2 7/3 5/4 1/5	3.1	15	G67	
3/1 13/2 7/3 5/4 1/5	3.2	20	G60	$Z^* = 67$
3/1 13/2 7/3 5/4 1/5	3.3	40	71.40	
3/1 13/2 7/3 4/4 1/5	3.4	30	67.25	
2/1 12/2 7/3 4/4 1/5	3.5	35	a) 62.30	
			b) 74.48	
5/1 15/2 11/3 9/4 3/5	3.1	15	G48	
5/1 9/2 13/3 2/4 3/5	3.2	15	G40	$Z^* = 48$
2/1 5/2 3/3 12/4 7/5	3.3	35	52.95	
11/1 2/2 1/3 8/4 4/5	3.4	35	48.26	
6/1 1/2 4/3 2/4 10/5	3.5	40	a) 42.34	
			b) 52.41	
12/1 15/2 20/3 6/4 8/5	3.1	12	G66	
4/1 7/2 6/3 5/4 3/5	3.2	12	G59	$Z^* = 66$
2/1 5/2 3/3 6/4 6/5	3.3	25	68.15	
5/1 2/2 2/3 5/4 4/5	3.4	30	66.45	
4/1 1/2 4/3 3/4 5/5	3.5	35	a) 63.75	
			b) 69.80	

Tabela 4.10: Exemplo com 5 peças e 5 máquinas.

Problema	Algoritmo	Processamento aproximado (segundos)	Makespan (minutos)	Observações
	3.1	4	G53	
10/1 13/2 8/3 15/4	3.2	4	G48	
12/1 14/2 9/3 10/4	3.3	4	58.62	$Z^* = 53$
	3.4	4	53.40	
	3.5	7	a) 50.60	
			b) 57.50	
	3.1	4	G29	
8/1 15/2 4/3 21/4	3.2	5	G48	
8/1 14/2 3/3 20/4	3.3	5	57.40	$Z^* = 52$
	3.4	6	52.35	
	3.5	8	a) 50.50	
			b) 55.40	
	3.1	4	G50	
5/1 9/2 1/3 22/4	3.2	4	G45	
11/1 20/2 8/3 4/4	3.3	6	53.40	$Z^* = 50$
	3.4	7	50.65	
	3.5	10	a) 47.30	
			b) 52.50	
	3.1	3	G39	
8/1 10/2 5/3 12/4	3.2	3	G34	
1/1 5/2 2/3 4/4	3.3	4	43.56	$Z^* = 39$
	3.4	4	39.50	
	3.5	8	a) 37.50	
			b) 41.45	

Tabela 4.11: Exemplo com 2 peças e 4 máquinas.

O algoritmo 3.3 também, como esperado, forneceu resultados bastante amplos no sentido de permitir a utilização de diferentes metodologias na sua abordagem, sem perder quanto à rapidez e eficiência computacional.

Dentre as soluções obtidas utilizando os métodos de ordenamento de números fuzzy (procedimento 3.5), o planejador pode escolher uma "boa" solução

com um grau de pertinência $\mu(x)$ "adequado" e um grau de credibilidade nas folgas nos recursos "ajustados" à realidade de cada caso. Esta flexibilidade, no entanto, é obtida com um custo alto tanto do ponto de vista de envolvimento do planejador, como de processamento.

Capítulo 5 . Conclusões

O problema de sequenciamento do processamento de peças pelas máquinas, junto com a questão de translado das mesmas em células flexíveis de manufatura (scheduling em FMC), foi o alvo de nossa pesquisa neste trabalho. Abordamos um problema clássico de jobshop, mas introduzindo dois aspectos: primeiro, na formulação do problema que permitiu reduzir substancialmente o número de variáveis inteiras e até mesmo conseguindo a sua eliminação, e, segundo, na introdução de incertezas nos tempos de processamento das peças pelas máquinas, como também nos tempos de transporte das peças pelo robô.
Os problemas de programação horária de peças e máquinas numa FMC geralmente são modelados como de programação linear inteira mixta e sua resolução é abordada por técnicas que não podem levar em conta as imprecisões que existem na prática e os resultados obtidos, muitas vezes carecem de significados mais abrangentes e reais.

Considerando o cenário acima definido, apresentamos cinco metodologias que foram frutos da aplicação da teoria de conjuntos nebulosos (fuzzy set theory):

- Na primeira, a lógica fuzzy foi utilizada através da construção de sentenças baseadas em regras clássicas para resolver conflitos dinâmicos que aparecem nas máquinas e no robô. Para tornar esta aplicação possível, foi definido o conceito de *estágio*, para lidarmos com incertezas nos conflitos nas máquinas e no robô. Com a utilização deste conceito e das variáveis linguísticas e regras de inferência, funções de pertinência foram construídas para as variáveis de decisão. Esta metodologia teve como primeiro objetivo, permitir a aplicação de lógica nebulosa nos problemas em consideração. Apesar deste aspecto, aplicada a problemas clássicos de sequenciamento em FMC, mostrou-se bastante eficiente, tanto do ponto de vista computacional como na qualidade dos resultados obtidos. É importante lembrar que a formulação adotada foi fundamental para o sucesso deste algoritmo, considerando que ficamos sem as variáveis inteiras. Vale ressaltar que é mais eficiente que os métodos clássicos,

uma vez que não resolve problemas de programação inteira mixta e que fornece soluções sub-ótimas.

- Visando aperfeiçoar o algoritmo anterior, procuramos extender o conceito de estágio para *janelas* de tempo, onde mais que um estágio de um mesmo job pode ocorrer. Com esta alteração, formulamos um segundo algoritmo, com qualidade de resultados melhor que o primeiro, sem no entanto, perder quanto à eficiência computacional.

- A idéia do terceiro algoritmo foi essencialmente baseada no primeiro algoritmo, mas utilizando Programação Linear Fuzzy, aplicado ao primeiro modelo do capítulo 2. Para isso, manteve-se o conceito de estágio e a programação horária foi feita de estágio em estágio. Aqui, depois de transferir o problema de programação linear inteiro mixto a um problema de programação linear, usamos as técnica de programação paramétrica, a programação linear fuzzy de Werners e a programação linear fuzzy de Zimmermann para apresentar um conjunto de soluções alternativas ao planejador para a escolha da melhor solução para cada caso. Este conjunto de soluções fornece ao planejador, uma ampla idéia sobre o domínio onde se localiza o ótimo (α Pareto ótimo). Este modelo FLP fornece um novo ponto de vista para tratar as imprecisões apresentadas nos tempos de término das tarefas e nos tempos prontos para o início da próxima tarefa e é um modelo computacionalmente muito eficiente, já que não lida com variáveis inteiras.

- No quarto algoritmo, procuramos aperfeiçoar o terceiro, no sentido de melhorar a qualidade da solução. Utilizamos o conceito do segundo algoritmo (janela de tempo) e aplicamos ao modelo 1 de programação horária definido no capítulo 2. Neste caso, houve a necessidade de trabalharmos com variáveis inteiras que definem a precedência ou não do estágio de um job que demanda o mesmo tipo de máquina em relação aos estágios de outros jobs. No procedimento

de resolução proposto, existe a necessidade de detectar a presença ou não de conflitos entre tarefas dos jobs e foi sugerida uma metodologia para esta questão, baseada no conceito de grau de pertinência do conflito. Este algoritmo também mostrou-se computacionalmente eficiente, uma vez que assumimos uma heurística que vai resolvendo os conflitos de job em job.

- No quinto, adotamos o mesmo modelo que da terceira metodologia quanto à natureza do problema, mas utilizando na resolução, os conceitos de comparação de números fuzzy, através de funções de ordenamento conhecidas, inspirado no modelo geral de Delgado et al. Esta aproximação fornece uma amplitude bem maior de possibilidades na análise do problema, relacionado com todos os coeficientes, tanto da função objetivo como da matriz tecnológica e recursos. No entanto, na obtenção dessa riqueza de resultados, é necessário um maior esforço no seu tratamento e na sua resolução.

O primeiro e o terceiro algoritmos são mais eficientes que o segundo e o quarto do ponto de vista computacional e fornece a flexibilidade para o usuário ou decisor na escolha da heurística mais adequada para cada caso. Por outro lado, o terceiro e o quarto algoritmos fornecem ao decisor, uma visão ampla das influências das variações dos parâmetros do problema, com vantagens do quarto sobre o terceiro quanto à qualidade do resultado. Os exemplos mostraram que todos os quatro enfoques são computacionalmente eficientes e forneceram resultados bastante próximos do ótimo, senão o ótimo e bastante aderentes com a realidade. O quinto modelo teve como objetivo a aplicação dos conceitos de ordenamento de números fuzzy e foi mostrado que fornece uma amplitude bastante grande quanto aos possíveis resultados.

É do nosso conhecimento que os problemas de programação horária são do tipo NP-hard e, assim sendo, na medida que a dimensão em termos de número de máquinas e de jobs aumenta, a complexidade cresce exponencialmente. Com a utilização dos enfoques aqui propostos, é possível resolver problemas de grandes

dimensões utilizando recursos clássicos de computação, ressalvando a quarta e a quinta metodologias, pois nestes casos, temos um complicante de termos que tratar variáveis inteiras e incertezas de muitos coeficientes, respectivamente.

Cabe lembrar que as técnicas aqui propostas são perfeitamente utilizáveis em problemas com estruturas semelhantes e que sejam modeláveis como problemas de programação linear fuzzy, em função da presença de incertezas nos seus parâmetros.

Assim, uma extensão natural das metodologias aqui propostas são para problemas de jobshop com rotas alternativas e/ou com capacidades de estoque limitados, etc. Os problemas de programação inteira ou mixta, também podem sofrer tratamentos semelhantes aos apresentados neste trabalho, com a utilização de funções de pertinência associadas às vaiáveis inteiras incertas, adequadas para cada caso. Uma outra extensão que apresenta boas possibilidades é para problemas não-lineares de programação matemática, notadamente para os convexos. Alguns autores já fizeram incursões nesta direção (Statinikov, (1990); Sobol, (1992); Stauer and Sun, (1995)), obtendo resultados interessantes e promissores na linha denominada PSI - Parameter Space Investigation. Por outro lado, considerando que estamos tratando de problemas de alta complexidade com dimensões que crescem exponencialmente, que estes problemas apresentam estruturas particulares e que a informática coloca à disposição equipamentos cada vez mais eficientes e sofisticados, a utilização de técnicas apropriadas de decomposição associada a técnicas de processamento paralelo apresenta-se como uma alternativa promissora de investigação.

Referências bibliográficas

[1] Adamo, J. M., Fuzzy Decissions Trees, Fuzzy Sets and Systems, vol. 4, (1980), pp. 207-219

[2] Ashour, S., A Branch-and-Bound Algorithm for Flow-Shop Scheduling Problems, AILE Trans. vol. 2, (1970), pp. 172-176.

[3] Baker, K. R., Introduction to Sequencing and Scheduling., John Wiley, and Sons, New York, (1974).

[4] Baldwing, J. F. and N. C. Guild, Comparison of Fuzzy Sets on the Same Decision Space, Fuzzy Sets and Systems, vol. 2, (1979), pp. 213-233.

[5] Baldwing, J. F., Fuzzy Logic and its Applications to Fuzzy Reasoning, In: Gupta, Ragade and Yager, (1979).

[6] Baldwing, J. F. and N. C. Guild, Comments on the Fuzzy Max Operator of Dubois and Prade, Internat. J. Systems Sci. vol. 10, (1979), pp. 1063-1064

[7] Baldwing, J. F. and N. C. F. Guild, Feasible Algorithms for Approximate Reasoning using Fuzzy Logic., Fuzzy Sets and Systems, vol. 3, (1980).

[8] Baldwing, J. F., and B. W. Pilsworth, Axiomatic Approach to Implication for Approximate Reasoning with Fuzzy Logic, Fuzzy Sets and Systems, vol. 3, (1980).

[9] Bazaraa, M. S/ an J. J. Jarvis, Linear Programming and Network Flows, John Wiley, and Sons, (1977).

[10] Bellmann, R. E. and Zadeh, L. A., Decision-making in a Fuzzy Environment, Management Sci. vol. 17, (1970), pp. 141-164.

[11] Bensana, E., G. Bel and D. Dubois, OPAL: A Multi-Knowledge-based System for Industrial Job-Shop Scheduling., Int. J. Production Res., vol. 26, No. 5, (1988), pp. 795-819.

[12] Bestwick, P. F. and N. A. J. Hastings, A New Bound for Machine Scheduling, Op. Res. Quart. vol. 27, (1976), pp. 479-487.

[13] Bilge, U. and G. Ulusoy, A Time Window Approach to Simultaneous Scheduling of Machines and Material Handling System in An FMS, Operations Research, Vol.43, No.6, November-December (1995), pp.1058-1070.

[14] Blackstone JR, J. H., D. T. Phillips and G. L. Hogg, A State-of-the-Art Survey of Dispatching Rules for Manufacturing Job Shop Operation, Int. J. Production Res., vol. 20, No. 1, (1982).

[15] Bortolan, G. and R. A. Degani, A Review of Some Methods for Ranking Fuzzy Sets, Fuzzy Sets and Systems, vol. 16, (1985), pp. 123-138.

[16] Campos, L. and J. L. Verdegay, Linear Programming Problems and Ranking of Fuzzy Numbers, Fuzzy Sets and Systems, 32, (1989), pp. 1-11.

[17] Chanas, S., The Use of Parametric Programming in Fuzzy Linear Programming, Fuzzy Sets and Systems, vol. 11, (1983), pp. 243-251.

[18] Chang, T. C., K. Hasegawa and C. N. Ibbs, The Effects of Numbership Function on Fuzzy Reasoning, Fuzzy Sets and Systems, vol. 44, (1991).

[19] Chryssolouris, G., K. Dicke and M. Lee, An Approach to Real-Time Flexible Scheduling, The International Journal of Flexible Manufacturing Systems, vol. 6, (1994), pp. 235-253.

[20] Cunto, E., Scheduling Boats to Sample Oil Wells in Lake Maracaibo, Operations Research, vol. 26, nº 1, (1973), pp. 183-196.

[21] Daniels, R.L. and J.B. Mazzola, Flow Shop Scheduling With Resource Flexibility, Operations Research, Vol.42, No.3, May-June (1994), pp.504-522.

[22] Delgado, M. Verdegay, J. L., Vila, M. A., A general model for fuzzy linear programming, Fuzzy Sets and Systems, vol. 29, (1989), pp. 21-29.

[23] De Matta, R. and M. Guinard, Dynamic Production Scheduling For A Process Industry, Operations Research, Vol.42, No.3,May-June (1994), pp.492-503.

[24] Dubois, D. and H. Prade, Fuzzy Real Algebra: Some Results, Fuzzy Sets and Systems, vol. 2, (1979), pp. 327-348.

[25] Dubois, D. and H. Prade, Fuzzy Sets in Approximate Reasoning, Part 1: Interface with Possibility Distributions, Fuzzy Sets and Systems, vol. 40, (1991).

[26] Dubois, D., J. Lang and H. Prade, Fuzzy Sets in Approximate Reasoning, Part 2: Logical Approaches, Fuzzy Sets and Systems, vol. 40, (1991).

[27] Gaines, B. R., Foundations of Fuzzy Reasoning, Fuzzy Automata and Decision Processes., Edited by M. M. Gupta G. N. Saridis, B. R. Gaines, (1977).

[28] Garey, M.R., Graham, R.L., Johnson, D.S., Performance Guarantees for Scheduling Algorithms, Operational Research, v. 26, n.1, (1978), p.3-21.

[29] Garfinkel, R. S. and G. L. Nemhauser, Integer Programming, John Wiley, and Sons, (1972).

[30] Giles, R., Logic and Fuzzy Set Theory, Intern. J. of Man-Machine Studies, vol. 8, (1977).

[31] Gonzalez, T. and S. Sahni, Flowshop and Jobshop Schedules: Complexity and Approximations, Operations Research, vol. 26, nº 1, (1978), pp. 36-52.

[32] Graham, R.L., Bounds on Multiprocessing Timing Anomalies, SIAM J. Appl. Math., vol.17, No2, March(1980), pp.416-429.

[33] Gupta, J. N. D., Optimal Schedules For Special Structure Flowshops, Nav. Res. Log. Quart., vol. 22, (1975), pp. 255-269.

[34] Hamacher, H., Laberling, H., and Zimmermann, H. -J., Sensitivity Analisis in Linear Programming., Fuzzy Sets and Systems, 1, (1978), pp. 269-281.

[35] Hirota, K. and W. Pedrycz, Analysis and Synthesis of Fuzzy Systems by the use of fuzzy sets., Fuzzy Sets and Systems, vol. 10, (1983).

[36] Horowitz, E., and S. Sartoy, Fundamentals of Computer Algorithms, Computer Science Press, (1978).

[37] Inman, R. R. and P. C. Jones, Decomposition for scheduling manufacturing systems, Operations Research, vol. 41, (1993).

[38] Johnson, S. M., Optimal Two and three-stage Production Schedules with Setup Times Inclued, Nav. Res. Log. Quart., vol. 1, (1954), pp. 61-68.

[39] Kandel, A., Fuzzy Mathematical Techniques with Applications (Addison-Wesley, Reading, MA, (1986).

[40] Kaufmann, A., Progress in Modeling of Human Reasoning of Fuzzy Logic, Fuzzy Automata and Decision Processes, Edited by M. M. Gupta, G. N. Saridis, B. R. Gaines, (1977).

[41] Kickert, W. J. M. and E. H. Mamdani, Towards an Analysis of Linguistic Modeling, Fuzzy Sets and Systems, vol. 2, (1979).

[42] Kizka, J. B., M. E. Kochanska and D. S. Sliwinska, The Influence of some Fuzzy Implications Operators on the Accuracy of a Fuzzy Model, Part I, Part II, Fuzzy Sets and Systems, vol. 15, (1985).

[43] Klir, G. J. and Folger, T. A., Fuzzy Sets, Uncertainty and Information, Prentice Hall, (1988).

[44] Koulamas, Ch., The Total Tardiness Problem: Review and Extensions, Operations Research, Vol.42, No.6, November-December, (1994), pp.1025-1041.

[45] Lai, Young-J and Ching-L. Hwang, Interactive Fuzzy Linear Programming, Fuzzy Sets and Systems, vol. 45, (1992), pp. 169-183.

[46] Lageweg, B. J., Lenstra, J. K., and Rinnooy Kan, H. G., Job-Shop Scheduling by Implicit Enumeration, Management Sci., vol. 24, No. 4, (1976), pp. 441-449.

[47] Lageweg, B. J., J. R. Lenstra, and A. G. H. Rinnooy Kan, A General Bounding Scheme for the Permutation Flow-Shop, Operations Research, vol. 26, nº 1, (1978), pp. 53-67.

[48] Lee, C. C., Fuzzy Logic in Control Systems: Fuzzy Logic Controler-Part I e Part II, IEEE Trans. on Syst. Man Cybern., vol. 20, nº 2, (1990), pp. 404-435.

[49] Lenstra, J. K. and A. H. G. Rinnooy Kan, Complexity of Scheduling under Precedence Constraints, Operations Research, vol. 26, nº 1, (1978), pp. 22-35.

[50] Mansfield, E., The Diffusion of FMS in Japan, Europe and United States, Management Science, vol. 39, (1993), pp. 149-159.

[51] Merabet, A. A., Dynamic Job Shop Scheduling: An Operating System Based Design, Studies in Management Science and Systems, vol. 12, (1986), pp. 257-270.

[52] Misumoto, M., and H. J. Zimmermann, Comparison of Fuzzy Reasoning Methods, Fuzzy Sets and Systems, vol. 8, (1982).

[53] Morton, T. E., and T. L. Smunt, A Planning an Scheduling System for Flexible Manufacturing, Studies in Management Science and Systems, vol. 12, (1986).

[54] Negoita, C. V. and Ralescu, D. A., Applications of Fuzzy Sets to Systems Analysis (John Wiley, and Sons 1975).

[55] Negoita, C. V. and Salaria, M., On Fuzzy Programming and Tolerances in Planning., Econom. Comp. Econom. Cybernet. Stud. Res., 1, (1976), pp. 3-15.

[56] PanWalker. S. S. and W. Iskander, A Survey of Scheduling Rules, Operations Research, vol. 25, nº 1, (1977).

[57] Prade, H., Using Fuzzy Set Theory in a Scheduling problem: A Case Study, Fuzzy Sets and System, vol. 2, (1979).

[58] Ragade, R. K., and Gupta, M. M., Fuzzy Set Theory: Introduction, Fuzzy Automata and Decision Process, Edited by M. M. Gupta, G. N. Saridis, B. R. Gainess, (1977).

[59] Ramik, J. and Rimanek, Inequality Relaton Between Fuzzy Numbers and its use in Fuzzy Optimization, Fuzzy Sets and Systems, vol. 23, (1985), pp. 123-138.

[60] Rao, M. B., and A. Rashed, Some Comments on Fuzzy Variables, Fuzzy Sets and Systems, vol. 6, (1981).

[61] Reumay, R. P. and Yamakami, A., An Approach to Fuzzy Scheduling Problems in a Flexible Manufacturing Cells, ELECTRO 95: Xl Congreso Chileno de Ingenieria Eléctrica, Punta Arenas, Chile, 13-17 Noviembre, (1995).

[62] Reumay, R. P. and Yamakami, A. Mathematical Programming Applied to Worpiece Scheduling Problem in a Flexible Manufacturing Cells with Uncertaintes, IFIP WG-76, Noisy-le-Grand, France, 28-30 May, (1996).

[63] Reumay, R. P. and Yamakami, A., An Efficient Fuzzy-Logic Based Heuristic Applied to a Flexible Manufacturing Cell, ITHUR'S - International Conference on Intelligent Technology and Human-Related Science, León, Spain 05-07/July, (1996).

[64] Reumay, R. P. and Yamakami, A., Programação Linear Fuzzy Aplicada ao Problema de Sequenciamento de Peças em Células Flexiveis de Manufatura com Incertezas, aceito no XI Congresso Brasileiro de Automática/I SAA, São Paulo, Brasil, 2-6 setembro, (1996).

[65] Rinnooy Kan, A. H. G., Machine Scheduling Problems:Classification, Complexity and Computations, The Hague, Netherlands, (1976).

[66]Rudin, W., Principles of Mathemathical Análisis, McGraw-Hill, New York, (1964).

[67] Shaw, M. J. P., and A. B. Whinston, Application of Artificial Intelligence to Planning and Scheduling in Flexible manufacturing, Studies in Managemet Science and Systems, vol. 12, (1986).

[68] Skala, H. J., On Many-valued Logics, Fuzzy Sets, Fuzzy Logics and their Applications, Fuzzy Sets and Systems, vol. 1, (1978).

[69] Sobol, I. M., A Global Search for Multicriterial Problems, Proceedings of the Nineth International Conference on Multiple Criteria Decision Making, Springer-Verlag, (1992), pp. 401-412.

[70] Statinikov, R. B., Multicriterial Design of Machines, IXth International Conference on Multiple Criteria Decision Making, Virginia, (1990).

[71]Stauer, R. E. and Sun, M., The Parameter Investigation Method of Multiple Objective Nonlinear Programming: A Computational Investigation, Operations Research, Vol. 43, nº4, (1995), pp. 641-648.

[72] Stecke, K. E., Formulation and Solution of Nonlinear Integer Production Planning Problems for Flexible Manufacturing Systems., Management Science, 29(3), (1983), pp. 273-288.

[73] Taha, H. A., Integer Programming Theory, Applications, and Computations, Academic Press, INC., (1975).

[74] Stecke, K. E. and J. J. Solberg, Loading and Control Policies for a Flexible Manufacturing System, International Journal of Production Research, vol. 19, No. 5, (1981), pp. 481-490.

[75] Tanaka, H., and K. Asai, Fuzzy Solution in Fuzzy Linear Programming Problems, IEEE Trans. Systems M. Cybernetics, 14, nº vol. 2, (1984-a).

[76] Tanaka, H., and K. Asai, Fuzzy Linear Programming Problems whith Fuzzy Numbers, Fuzzy Sets and Systems, vol. 13, (1984-b), pp. 1-10.

[77] Teixeira, E. M. A. and Yamakami, A., Workpieaces Scheduling in a Flexible Manufacturing Cell, Management Methods, North-Holland, (1990), pp. 217-224.

[78] Teixeira, E. M. A., Programação Horária de Peças em Células Flexíveis de Manufatura, Tese de Doutorado, FEE-Unicamp, Brasil, (1993).

[79] Venkata Subba Reddy P. and M. Syam Basu, Some Methods of Reasoning for Fuzzy Conditional Propositions, Fuzzy Sets and Systems, vol. 62, (1992).

[80] Wagner, W., A Fuzzy Model of Concept Representations in Memory, Fuzzy Sets and systems, vol. 6, (1981).

[81] Wang, L. X. and J. M. Mendel, Generating Fuzzy Rules from Numerical Data, with Applications, IEEE Trans. on Systems, Man, and Cybern., 22, nº, vol. 6, (1992), pp. 1414-1427.

[82] Wang L. C. and C. Liu, Intelligent Scheduling of FMSs with Inductive Learning Capability Using Neural Networks, International Journal of Flexible Manufacturing Systems, vol. 7, No. 2, (1995), pp. 147-175.

[83] Werners, B., An Interactive Fuzzy Programming System, Fuzzy Sets and Systems, vol. 23, n°1, (1987), pp. 131-147.

[84] Wiedely, G., and H. J. Zimmermann, Media Selection and Fuzzy Linear Programming, J. Oper. Res. Soc., vol. 2, (1987), pp. 1071-1084.

[85] Won, Y. J., and H. S. Cho, A Fuzzy Rule-Based Method for Seeking Stable Arc Condition under Short Circuiting Mode of GMA Welding Process, Part 1:, Journal of Systems and Control Engineering, (1992), pp. 117-125.

[86] Yager, R. R., On Choosing Between Fuzzy Subsets, Kybernetes, vol. 9, (1980), pp. 151-154.

[87] Yager, R. R., A Procedure for Ordering Fuzzy Subsets of the Unit Interval, Inform. Sci. 24 (1981), pp. 143-161.

[88] Yager, R. R., S. Ovchinnikov, R. M. Tong, H. T. Nguyen, Fuzzy Sets and Applications: Selected papers by L. A. Zadeh, Wiley, New York, (1987).

[89] Yager, R. R., Connectives and Quantifiers in Fuzzy Sets, Fuzzy Sets and Systems, vol. 40, (1991).

[90] Yamakami, A. and Reumay, P., Fuzzy Linear Programming Applied to Job-Shop Scheduling Problem, accepted to ISFL'97-Second ICCS Symposium on Fuzzy Logic and Applications, Zurich, Switzerland, 12-14 February, (1997).

[91] Zhao, R. and R. Govind, Defuzzification of Fuzzy Intervals, Fuzzy Sets and Systems, vol. 43, (1991).

[92] Zadeh, L. A., Fuzzy Sets, Information and Control, vol. 8, (1965), pp. 338-353.

[93] Zimmermann, H. J., Description and Optimization of Fuzzy System, Int. J. General Systems, vol. 2, (1976), pp. 209-215.

[94] Zimmermann, H. J., Fuzzy Programming and Linear Programming with Several Objetive Functions, Studies in the Management Sciences, vol. 20, (1984), pp. 109-121.

[95] Zimmermann, H. J., M. A, Pollastechek, Fuzzy 0-1 Linear Programs, Studies in the Management Sciences 20, (1984), pp. 133-145.