UNIVERSIDADE FEDERAL DE PELOTAS
Centro de Desenvolvimento Tecnológico
Programa de Pós-Graduação em Computação

Dissertação

Escalonamento de Tarefas BoT em Grades Computacionais considerando o Regime de Incerteza

Bruno Moura Paz de Moura

Pelotas, 2017

Bruno Moura Paz de Moura

Escalonamento de Tarefas BoT em Grades Computacionais considerando o Regime de Incerteza

Dissertação apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal de Pelotas, como requisito parcial à obtenção do título de Mestre em Ciência da Computação

Orientadora: Prof. Dr. Renata Hax Sander Reiser
Coorientador: Prof. Dr. Adenauer Corrêa Yamin

Pelotas, 2017

Universidade Federal de Pelotas / Sistema de Bibliotecas
Catalogação na Publicação

M929e Moura, Bruno Moura Paz de

Escalonamento de tarefas BoT em grades computacionais considerando o regime de incerteza / Bruno Moura Paz de Moura ; Renata Hax Sander Reiser, orientadora ; Adenauer Corrêa Yamin, coorientador. — Pelotas, 2017.

113 f.

Dissertação (Mestrado) — Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, 2017.

1. Lógica fuzzy. 2. Computação em grade. 3. Escalonamento de tarefas. 4. Bag-of-tasks. I. Reiser, Renata Hax Sander, orient. II. Yamin, Adenauer Corrêa, coorient. III. Título.

CDD : 005

Elaborada por Maria Inez Figueiredo Figas Machado CRB: 10/1612

Dedico este trabalho aos meus pais Doli Oliveira de Moura, Marlene Moura Paz de Moura minha irmã Débora Moura Paz de Moura, ao meu filho Miguel de Almeida Moura e a minha esposa e amiga, Daniela Martins de Almeida, pelo apoio incondicional e compreensão, pelas horas que dediquei à elaboração desta dissertação, aos meus amigos e colegas de trabalho que, de alguma forma, contribuíram direta ou indiretamente.

AGRADECIMENTOS

À minha família, e especialmente aos meus pais, irmã, esposa e filho que me incentivaram nos momentos mais difíceis e sempre me motivaram e ajudaram de todas as formas em minha educação e formação.

À minha esposa Daniela, pela compreensão, apoio, confiança e inspiração, que foram fatores fundamentais para a conclusão desta etapa.

Aos professores orientadores, pela confiança, atenção e orientação dedicada, pelo conhecimento transmitido e por todas as reuniões, revisões e demais colaborações essenciais ao desenvolvimento desse trabalho.

Aos membros da banca, pelo interesse e pelas contribuições com correções e sugestões.

Aos amigos que me apoiaram em todas as horas.

Aos colegas e professores da Universidade Federal de Pelotas – UFPel, Universidade Federal do Pampa – UNIPAMPA e ao Instituto Federal de Educação Ciência e Tecnologia Sul-Rio-grandense – IFSul.

A todas as pessoas que, direta ou indiretamente, me ajudaram a percorrer este caminho e a construir todo o conhecimento necessário para a realização deste trabalho, seja por meios de ensinamentos ou de inspiração.

A menos que modifiquemos
a nossa maneira de pensar,
não seremos capazes de
resolver os problemas causados
pela forma como nos
acostumamos a ver o mundo.
— ALBERT EINSTEIN

RESUMO

MOURA, Bruno Moura Paz de. **Escalonamento de Tarefas BoT em Grades Computacionais considerando o Regime de Incerteza**. 2017. 113 f. Dissertação (Mestrado em Ciência da Computação) – Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2017.

Dentre os desafios de pesquisa, na Computação em Grade temos a necessidade de tornar o sistema robusto tanto quanto às incertezas das diferentes medidas extraídas da infraestrutura computacional, como quanto à imprecisão das computações referente à tomada de decisões. Particularmente, a apropriação de informações relevantes ao escalonamento de tarefas a partir de um conjunto de fontes heterogêneas e de comportamento dinâmico tem sido foco de um significativo número de estudos. Neste cenário, esta dissertação tem como objetivo central a consolidação do módulo fuzzy *fGrid* para o escalonamento de tarefas na Computação em Grade, visando com isso sua extensão, considerando o tratamento das informações relacionadas ao Poder Computacional e ao Custo de Comunicação dos recursos computacionais envolvidos. Para tal, esta abordagem emprega lógica fuzzy do tipo 2 na concepção para a extensão denominada de *Int-fGrid*, contribuindo, desta forma, em duas relevantes frentes, tanto com o regime de incerteza destas informações, quanto com a imprecisão das computações envolvidas. Neste sentido, o trabalho se embasa em uma abordagem multi-valorada; em particular, integrando a lógica fuzzy e a matemática intervalar. No que diz respeito à avaliação dos módulos, é empregado um *framework* de simulação para aplicações distribuídas com intuito de facilitar a reprodutibilidade dos diferentes algoritmos em distintos cenários para infraestrutura de grade. Os resultados obtidos através das execuções de simulações com os escalonamentos gerados com auxílio dos módulos *fGrid* e *Int-fGrid* se mostram promissores e apontam para continuidade da pesquisa. Estas execuções levaram em consideração diferentes configurações de cenários para as infraestruturas de grades, ao mesmo tempo que, diversificados perfis de aplicações constituídas por tarefas Bag-of-Tasks, mostraram que as abordagens com uso da lógica fuzzy do tipo 1 e tipo 2 obtiveram ganhos de até 96,37% no *makespan*, comparado ao método Randon de seleção de recursos, 36,9 vezes confrontado ao Round-Robin, e 18,5 vezes em relação ao XSufferage.

Palavras-chave: Lógica Fuzzy, Computação em Grade, Escalonamento de Tarefas, Bag-of-Tasks.

ABSTRACT

MOURA, Bruno Moura Paz de. **Employing Fuzzy Logic for the Scheduling of Tasks in Computational Grids Considering the Uncertainty Regime**. 2017. 113 f. Dissertação (Mestrado em Ciência da Computação) – Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2017.

Among the research challenges in Grid Computing we need to make the system robust both to the uncertainties of the different measures extracted from the computational infrastructure, and to the imprecision of the computations related to decision making. In particular, the appropriation of information relevant to the scheduling of tasks from a set of heterogeneous sources and dynamic behavior has been the focus of a significant number of studies. In this scenario, this masters dissertation aims to consolidate the fuzzy module *fGrid* for the scheduling of tasks in Grid Computing, aiming at its extension considering the information processing related to Computational Power and the Communication Cost of resources involved. For this, this approach employs type 2 fuzzy logic in the design for the so-called *Int-fGrid* extension, thus contributing to two relevant fronts, both with the uncertainty regime of this information and with the imprecision of the computations involved. In this sense, the work is based on a multi-valued approach, in particular integrating fuzzy logic and interval mathematics. As regards the evaluation of the modules, a simulation framework is used for distributed applications in order to facilitate the reproducibility of the different algorithms in different scenarios for grid infrastructure. The results obtained through the simulations with the schedules generated with the help of the *fGrid* and *Int-fGrid* modules show promise for the continuity of the study. These executions took into account different configurations of scenarios for grid infrastructures, while different application profiles made up of Bag-of-Tasks tasks showed that the approaches using fuzzy logic type 1 and type 2 obtained gains of up to 96.37% in makespan, compared to the Randon method of resource selection, 36.9 times compared to Round-Robin, and 18.5 times in relation to XSufferage.

Keywords: Fuzzy Logic, Grid Computing, Scheduling Tasks, Bag-of-Tasks.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BR	Base de Regras
BoT	Bag-of-Tasks
CC	Custo de Comunicação
CFs	Conjuntos Fuzzy
DAG	Directed Acyclic Graph
ETGC	Escalonamento de Tarefas em Grades Computacionais
FP	Função de Pertinência
GC	Grade Computacional
IT2FLT	Interval Type-2 Fuzzy Logic System Toolbox
LFI	Lógica Fuzzy Intervalar
LF	Lógica Fuzzy
P2P	Peer-to-Peer
PC	Poder Computacional
P	Prioridade
SBRF1	Sistemas Baseados em Regras Fuzzy 1
SBRF2	Sistemas Baseados em Regras Fuzzy 2
TLs	Termos Linguísticos
VLs	Variáveis Linguísticas

SUMÁRIO

1 INTRODUÇÃO

Em consequência ao advento das tecnologias de comunicação dos últimos tempos, traduzidas principalmente no sucesso da Internet, motivou usuários, desenvolvedores e administradores de *clusters* e de supercomputadores a construírem aplicações que utilizassem recursos em escala mundial. Aplicações que antes utilizavam uma quantidade limitada de recursos de uma única organização passaram a ter à disposição centenas de milhares de computadores, surgindo assim as organizações virtuais. O poder de processamento desses novos computadores virtuais é resultante principalmente do aproveitamento dos ciclos não utilizados de computadores pessoais (DANIEL MACEDO BATISTA, 2010; SOOD; KOUR; KUMAR, 2016; GULLAPALLI, 2016).

Para o escalonamento otimizado de tarefas em Grades Computacionais, frequentemente, dentre os fatores que implicam em incerteza, tem-se o Poder Computacional (PC) e o Custo de Comunicação (CC). Por exemplo, o CC entre equipamentos que compartilham a Internet como meio de interoperabilidade é dependente do uso médio da infraestrutura de comunicação no momento que o escalonamento ocorre entre as máquinas que compõem a Grade Computacional (GC). Consequentemente, surge a necessidade da implementação de sistemas robustos à incerteza de medidas de variáveis extraídas do contexto da Computação em Grade (SOMASUNDARAM et al., 2014).

Na concepção da abordagem da proposta, é utilizado o formalismo e fundamentação com base na Lógica Fuzzy (LF). A LF, lógica nebulosa, ou ainda, lógica difusa, é uma lógica multivalorada capaz de absorver informações vagas, normalmente descritas em uma linguagem natural e convertê-las para um formato numérico de fácil manipulação computacional. Esta facilidade mostra-se oportuna para o uso na concepção de heurísticas de escalonamento de custo computacional reduzido, procurando com isso modelar os modos imprecisos do raciocínio na tomada de decisões a partir de informações obtidas através da infraestrutura dos ambientes de GC.

As primeiras noções da LF foram apontadas por um lógico polonês Jan Lukasiewicz (FONT; HÁJEK, 2002) em 1920, introduzindo a lógica dos conceitos vagos a partir dos conjuntos com três graus de pertinência (0, 0.5, 1).

Segundo (VON ALTROCK, 1996), a primeira publicação sobre LF foi em 1965, quando recebeu esta denominação. Seu autor foi Lotfi Asker Zadeh (ZADEH, 1965), professor na Universidade da Califórnia, Berkeley, USA, desenvolvendo os fundamentos da LF ao combinar os conceitos da lógica clássica e os conjuntos de Lukasiewicz, definindo as funções de pertinência como extensão das funções características (ZADEH, 1965, 1975, 1994).

A principal diferença entre a proposição definida pelos conjuntos clássicos e a proposição definida sobre conjuntos fuzzy introduzida por Zadeh, está na valoração do grau de pertinência, cujos valores são números reais entre 0 e 1. Na abordagem clássica, um elemento pertence ou não a um determinado conjunto, ou é verdadeiro, ou é falso, ou ainda, pode ser 0 ou 1. Na teoria dos conjuntos fuzzy, o elemento pode pertencer, não pertencer ou pertencer parcialmente a um determinado conjunto. Assim, a cada elemento de um conjunto fuzzy é atribuído um grau de pertinência, esse valor pode ser: pertinência total, recebendo o valor 1, não pertinência, mantendo como valor 0, e pertinência parcial, tendo como valoração um número real maior que 0 e menor que 1.

1.1 Objetivos

O desenvolvimento desta dissertação tem como objetivo geral contribuir com um módulo para o Escalonamento de Tarefas em Grades Computacionais (ETGC) empregando a Lógica Fuzzy na modelagem das variáveis geradoras de incerteza PC e CC.

O esforço de estudo e pesquisa contemplou duas frentes:

- (i) a consolidação do *fGrid*, um módulo para o ETGC. Uma versão inicial deste trabalho esta reportada em (SAMPAIO et al., 2015) que emprega a LF tipo 1 na modelagem das variáveis causadoras de incerteza PC e CC.

- (ii) a concepção do módulo *Int-fGrid* gerando assim uma versão intervalar do *fGrid* com a utilização da Lógica Fuzzy Intervalar (LF tipo 2), visando com isto a integração ao tratamento da incerteza também uma abordagem para o tratamento da imprecisão, e aplicando um ambiente de grade virtual para a obtenção dos resultados.

Considerando este objetivo geral, os objetivos específicos elencados são:

- Caracterizar os principais conceitos sobre LF e LFI utilizados para a concepção dos módulos para ETGC;

- Organizar os principais conceitos sobre Grades Computacionais;

- Desenvolver sobre o *framework* SimGrid o modelo computacional das diferentes arquiteturas de grade computacional a serem empregadas na avaliação da pesquisa;
- Sistematizar trabalhos relacionados aplicados ao escalonamento de tarefas usando LF, construindo o estado da arte na área;
- Divulgar ante a comunidade científica os resultados atingidos pela pesquisa através de publicações em eventos e/ou jornais especializados da área.

O escalonamento de tarefas é considerado um problema NP-Completo (RONG; ZHIGANG, 2005; PINEDO, 2012; BAJAJ; DOGRA; SINGH, 2015); no entanto, em ambientes de computação distribuída como as grades computacionais o problema de escalonamento torna-se ainda mais desafiador, devido à variação do estado da grade no decorrer do tempo, dinamicidade, heterogeneidade e recursos fisicamente distantes uns dos outros, são fatores que promovem incerteza, podendo ainda esta ser caracterizada por outros aspectos como a flutuação da largura de banda disponível nos canais de comunicação no momento da realização do escalonamento que, ao mesmo tempo, apresentam-se como desafios a serem tratados na literatura em busca de soluções.

Para que a computação em grade possa atingir um bom desempenho, é preciso fazer um escalonamento adequado, dependendo do tipo de aplicação e tarefas que serão escalonadas. Assim, se faz necessária uma análise tanto dos escalonadores, quanto dos algoritmos de escalonamento, considerando ainda dinamicidade destes ambientes distribuídos de grades computacionais.

Seres humanos são capazes de lidar com processos complexos baseados em conhecimentos incertos ou imprecisos. Os sistemas fuzzy permitem modelar este conhecimento em termos linguísticos, considerando a fundamentação teórica dos conjuntos fuzzy.

Esta estruturação lógico-formal dos sistemas fuzzy fornece a base para geração de técnicas na proposta de soluções para processos incertos. As primeiras aplicações bem sucedidas foram na área de controle e automação.

Atualmente, tem-se uma vasta aplicabilidade na área das engenharias, de tomada de decisões, mineração de dados, planejamento e otimização, como pode ser visto em (ATANASSOV et al., 2003; HERRERA; MARTINEZ; SÁNCHEZ, 2005; OJHA; ABRAHAM; SNášEL, 2016; AMADOR-ANGULO; CASTILLO; CASTRO, 2016).

Encontram-se também aplicações envolvendo sistemas que fazem uso da LF integrada às ferramentas computacionais, por exemplo, às redes neurais e a programação evolutiva (DENGFENG; CHUNTIAN, 2002; YE, 2011; MITCHELL, 2005; WANG; XIN, 2005; LEITE et al., 2016). Tais sistemas são caracterizados como sistemas híbridos,

cuja capacidade de aprendizado amplia ainda mais as áreas de aplicação de sistemas fuzzy e motivam o estudo de novas extensões da LF.

Considerando que experimentações em ambientes reais de aplicações de alto desempenho são muitas vezes limitadas a determinados cenários e, quando uma solução para computação de alto desempenho é proposta, faz-se necessário realizar a análise em diversos contextos. Contudo, nem sempre é possível fazer a validação utilizando múltiplos parâmetros e cenários reais diferentes, por questões logísticas e/ou financeiras, dentre outras. Como alternativa, surge o emprego de simuladores dos ambientes reais, como *framework* SimGrid, permitindo, por meio da simulação, a validação de pesquisas em computação de alto desempenho em diversos cenários, configurações e parâmetros. Com uso de simuladores é possível modelar aplicações e arquiteturas de computação de alto desempenho e ainda estudar os diversos desafios na área.

A literatura mostra que existem diversos projetos de simuladores para a utilização em pesquisas na área da computação distribuída e de alto desempenho, dentre eles: OptorSim (BELL et al., 2003), GridSim (BUYYA; MURSHED, 2002), GroudSim (OSTERMANN; PRODAN; FAHRINGER, 2010), SimGrid (CASANOVA et al., 2014) destacam-se por serem utilizados na especificidade de grades computacionais. Neste contexto, a escalabilidade que o SimGrid apresenta motiva a sua escolha para utilização neste trabalho. Em relação às outras ferramentas, o SimGrid destaca-se através da escalabilidade, na possibilidade de controlar as configurações da simulação e no modelo dos recursos simulados.

Na computação distribuída e paralela, um problema recorrente é o de escalonamento. Segundo (CASANOVA; MARCHAL, 2002), o objetivo do escalonamento é encontrar a melhor solução que atribua um conjunto de tarefas a determinado conjunto de recursos disponíveis, de forma que o tempo de execução total da aplicação seja menor.

1.2 Organização do texto

Este trabalho está estruturado da seguinte forma: no Capítulo 1, está a introdução, os objetivos e a motivação. No Capítulo 2, tem-se a fundamentação conceitual referente ao tema abordado. No Capítulo 3, são tratados os assuntos sobre os trabalhos relacionados considerados para o desenvolvimento desta dissertação. No Capítulo 4, são apresentados os aspectos para a concepção, modelagem e prototipação dos sistemas fuzzy propostos para auxiliar na tomada de decisão no contexto do ETGC, e análise dos trabalhos relacionados. Logo no Capítulo 5, são apresentados os resultados obtidos e, por fim, no Capítulo 6, costam as considerações finais, principais conclusões, publicações realizadas e as pretensões de trabalhos futuros.

2 FUNDAMENTOS CONCEITUAIS

Este capítulo é dedicado a registrar os principais aspectos conceituais que se fizeram necessários para o desenvolvimento desta dissertação.

2.1 Grades Computacionais: Conceitos, Escalonamento e Simulação

As Grades Computacionais são constituídas através de infraestrutura composta por hardware e software que permite o compartilhamento de recursos em larga escala (REIS, 2005; FOSTER et al., 2008), tais como: dados, capacidade de processamento e armazenamento, sendo que esses recursos podem estar espalhados geograficamente.

Além disso, as grades podem ser compostas por uma grande variedade de recursos, como: estações de trabalho, supercomputadores, *clusters*, instrumentos científicos, dentre outros.

O objetivo principal das grades é possibilitar a alocação de uma enorme quantidade de recursos para a execução de aplicações paralelas a um custo menor do que se fossem utilizados supercomputadores (DA SILVA; CIRNE; BRASILEIRO, 2003).

As grades computacionais podem ser classificadas em função de sua funcionalidade (KRAUTER; BUYYA; MAHESWARAN, 2002) em:

- **Grades de processamento:** são utilizadas para executar aplicações que precisam de uma grande capacidade computacional, voltadas a resolver problemas que não poderiam ser resolvidos por um único recurso;
- **Grades de dados:** são utilizadas para gerenciar o armazenamento e acesso a grandes quantidades de dados;
- **Grades de serviços:** provê uma grande quantidade de serviços, tais como: softwares e recursos computacionais.

Construir uma grade que possua essas três funcionalidades é muito difícil e custoso, na maioria das vezes opta-se por construir uma grade computacional com a

funcionalidade mais apropriada para a necessidade apresentada.

A seguir, são introduzidas as características de uma grade computacional segundo (DA SILVA; CIRNE; BRASILEIRO, 2003):

- **Heterogeneidade dos recursos:** essa característica dificulta o escalonamento de tarefas em grades, pois os recursos podem ter velocidade de processadores diferente, interconexão diferente, velocidade de memória, velocidade e tamanho de disco diferentes, dentre outros. Diferenças que podem fazer com que algumas aplicações não sejam apropriadas para determinados recursos;
- **Compartilhamento de recursos:** como em uma grade podem ocorrer variações na carga e na disponibilidade das máquinas, pode ocorrer sobrecarga em alguns recursos, prejudicando o desempenho de algumas aplicações;
- **Movimentação de dados:** pelo fato das aplicações que executam em grades terem uma grande quantidade de dados as quais precisam ser movidos de um lugar para o outro, é preciso considerar o tempo gasto com a transferência desses dados.

Em função das características mencionadas anteriormente, o escalonamento eficiente de tarefas em um ambiente de grade é um problema complexo. Para obter um bom desempenho na execução das tarefas, o algoritmo utilizado para distribuí-las necessitará escolher os recursos mais apropriados.

Outro fator a ser considerado quando se pretende trabalhar com grades computacionais, é o tipo de aplicação mais apropriado para executar nesse ambiente. Devido ao fato de que, normalmente, os recursos de uma grade são interligados por grandes redes de computadores, que possuem uma latência alta de comunicação, as aplicações submetidas à grade, são frequentemente do tipo Bag-of-Tasks (BoT), mas não necessariamente.

Esse tipo de aplicação possui tarefas independentes, que não necessitam trocar informações entre si, ou seja, não há comunicação ou dependências entre tarefas. Aplicações BoT incluem buscas maciças tais como quebra de chave, aplicações de manipulação de imagem e algoritmos de mineração de dados (SENGER et al., 2009).

2.1.1 Escalonamento de Tarefas

Escalonamento de tarefas em grades computacionais é o processo de tomar decisões de escalonamento envolvendo recursos sobre múltiplos domínios administrativos. O escalonamento é realizado através da ativação de um conjunto de regras que indicam como e quando determinadas informações do sistema devem ser obtidas, de que maneira essas informações influenciem na distribuição de tarefas e quais serão os recursos utilizados para a execução das aplicações.

Assim, os algoritmos de escalonamento são utilizados para implementar as regras de uma política de escalonamento. Enquanto as políticas de escalonamento ditam as regras gerais de como lidar com processos e administrar recursos do sistema, os algoritmos de escalonamento estão preocupados com a implementação dessas regras que podem ser feitas de diversas formas (SOUZA, 2000; SCHOPF, 2002; REIS, 2005; CIRNE et al., 2007; SUN et al., 2010).

O escalonamento de tarefas em grades computacionais é uma área de pesquisa pelo desafio que a própria natureza desse sistema representa (CIRNE, 2002; JIANG; NI, 2009; GHANEM et al., 2010). As características das grades que implicam em desafios no momento da atribuição das tarefas são:

- **Grande quantidade de recursos:** a grande quantidade de recursos torna-se um problema para o escalonador, que pode se tornar um gargalo do sistema, pois ele deve escolher de forma apropriada, qual recurso irá executar cada tarefa.
- **Grande heterogeneidade de recursos:** máquinas pertencentes à grade podem apresentar configurações heterogêneas. Entre as configurações, as principais são: poder de processamento, interconexões e sistemas operacionais.
- **Alto compartilhamento de recursos:** a variação de carga nas máquinas causada pela submissão de novos processos ao sistema é proporcional ao número de usuários da grade, isto é, quanto mais usuários, maior será a variação de carga do sistema. Isso pode fazer com que políticas de escalonamento que não presumem tal fato atinjam um resultado negativo.
- **Movimentação de dados:** em grade computacional deve-se evitar a submissão de aplicações que realizem muita comunicação, pois a alta latência da rede de interconexão dos recursos pode causar prejuízos ao escalonamento.

A taxonomia para os algoritmos de escalonamento utilizada na literatura, que visa padronizar e facilitar o entendimento dos variados cenários onde podem ser executados foi definida em (CASAVANT; KUHL, 1988), quando foi produzida uma classificação hierárquica para algoritmos de escalonamento em sistemas computacionais distribuídos, apresentada na Figura 1. Para os ambientes de grades computacionais, normalmente o tipo de escalonamento necessário situa-se no ramo Global. Esta dissertação enquadra-se nesta taxonomia, dentro da classificação Global, Estática, Sub-ótimo, e mais especificamente Heurístico (vide destaque na Figura 1).

Na sequência é apresentado o detalhamento desta taxonomia.

- **Local & Global:** o escalonamento local determina como os processos residentes em um único processador são alocados e executados. Já o escalonamento

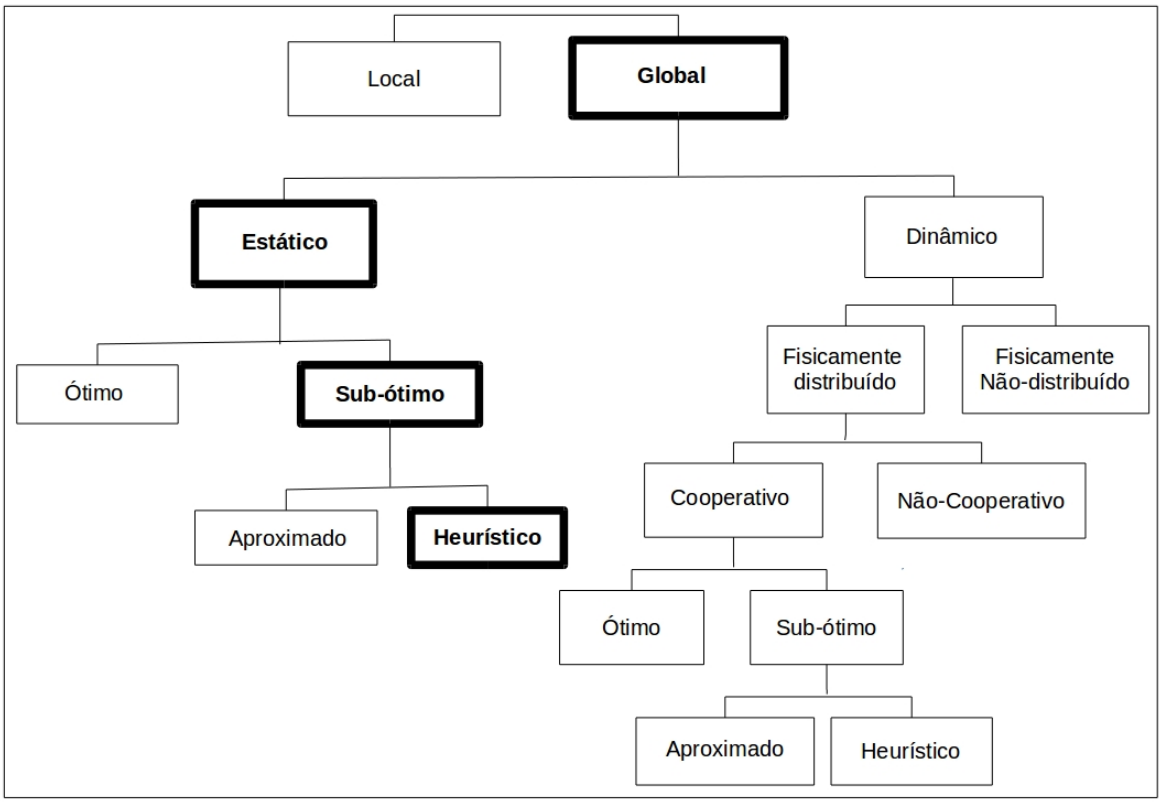


Figura 1: Taxonomia de Casavant.

global utiliza informações sobre o sistema para alocar processos para múltiplos processadores.

- **Estático & Dinâmico:** no escalonamento estático, as informações são obtidas antes do início do escalonamento da aplicação, sem possuir informações sobre as mudanças dinâmicas de estado do sistema no decorrer do processo. Ou seja, no escalonamento estático a atribuição de tarefas às máquinas é feita antes do início da execução. Já no escalonamento dinâmico, é realizada a alocação de tarefas durante a execução da aplicação.

- **Ótimo & Sub-ótimo:** se toda a informação do estado dos recursos e da aplicação é conhecida, o escalonamento poderá ser ótimo. Nos casos em que é computacionalmente inviável obter tais informações, o escalonamento alcançado é considerado sub-ótimo.

- **Aproximação & Heurística:** o algoritmo de aproximação procura implementar uma solução que seja suficientemente boa em relação à que foi definida como ótima. Já o algoritmo de heurística, leva em consideração alguns parâmetros que afetam o sistema de uma maneira indireta como, por exemplo, a comunicação entre processos.

- **Distribuído & Não distribuído:** nos escalonadores dinâmicos, as decisões de escalonamento global podem ser de um escalonador centralizado ou pode ser compartilhada por múltiplos escalonadores distribuídos. Já a estratégia centralizada pode ser mais simples em termos de implementação, se comparada à distribuída. Entretanto, poderá ser um gargalo em termos de desempenho, pois muitas aplicações podem ser enviadas para serem escalonadas simultaneamente.

- **Cooperativo & Não Cooperativo:** no modo cooperativo, cada escalonador responsabiliza-se por carregar sua própria porção do escalonamento de tarefas, mas todos os escalonadores trabalham em conjunto para o sistema global. Esta situação não ocorre no modo não cooperativo, onde os escalonadores individuais agem de forma independente, não se preocupando em melhorar o desempenho do resto do sistema.

  Complementarmente os algoritmos de escalonamento também podem ser classificados em (CASAVANT; KUHL, 1988; NEMETZ, 2011):

- **Preemptivos & Não Preemptivos:** escalonadores preemptivos permitem que uma tarefa em execução seja interrompida temporariamente e posteriormente retomada. Esse fato pode ocorrer se uma tarefa com maior prioridade chegar para ser executada. Já nos escalonadores não preemptivos, uma vez que uma tarefa for iniciada ela será executada até sua conclusão.

- **Homogêneos & Heterogêneos:** escalonadores que operam em sistemas homogêneos são aqueles nos quais as tarefas são distribuídas pelas unidades de processamento que possuem a mesma capacidade, memória e disponibilidade de recursos. Já os escalonadores que operam em sistemas heterogêneos, são capazes de lidar com sistemas nos quais as unidades têm diferentes capacidades computacionais.

2.1.2 SimGrid: Simulação Versátil de Sistemas Distribuídos

Uma maneira de fazer pesquisa em Computação Científica é através da modelagem e simulação de experimentos em computadores (QUINSON; BOBELIN; SUTER, 2010).

Antes da era da computação, os cientistas e engenheiros acreditavam que efetuar centenas de cálculos aritméticos em um segundo estava próximo do imaginável. De qualquer modo, assim que isso foi possível, eles já desejavam milhões, bilhões, trilhões de operações por segundo, ou seja, mais poder computacional (PACHECO, 1996). Quatrilhões de operações por segundo (petaflops), hoje, são uma realidade graças ao constante desenvolvimento das arquiteturas e tecnologias computacionais.

Através dos supercomputadores, grades computacionais e *clusters*, a computação científica tornou-se uma ferramenta essencial para o desenvolvimento da ciência moderna. Atualmente, grandes desafios como viagens espaciais, dinâmica química e molecular, dinâmica de fluidos, exploração petrolífera, por exemplo, podem ser estudados mais detalhadamente.

Aliado ao desenvolvimento das máquinas, há o das técnicas e seu aprimoramento, softwares e ferramentas de Processamento de Alto Desempenho (PAD) que precisam tirar o melhor proveito da capacidade computacional disponível no hardware. Nesse contexto, faz-se necessário também o desenvolvimento de pesquisas nesse campo (CASANOVA et al., 2014).

A experimentação em PAD pode ocorrer através de três abordagens: (i) real, (ii) simulado ou (iii) emulado (CLAUSS et al., 2011). A emulação consiste em executar uma aplicação em um ambiente específico e interceptar as chamadas do sistema fazendo a mediação, sem que seja necessário alterar a aplicação (CASANOVA et al., 2014).

Experimento real é aquele que se implementa uma solução para um determinado objeto de estudo em um ambiente real. O ponto positivo da experimentação real é que demonstra a aplicabilidade da estratégia proposta (CLAUSS et al., 2011). Para esse tipo de experimentação, a comunidade científica disponibiliza diversos projetos que permitem o desenvolvimento de aplicações e estudos em PAD onde podem ser executados em um ambiente real, tais como o Grid'5000 (BOLZE et al., 2006) e PlanetLab (CHUN et al., 2003).

No entanto, alguns pontos negativos devem ser destacados na abordagem real:

- (i) investimento de muito tempo e trabalho;
- (ii) dificuldade em escolher os cenários para validar o trabalho científico;
- (iii) limitação nos resultados aos cenários onde o experimento foi executado;
- (iv) impossibilidade de controlar o ambiente;
- (v) impossibilidade de reprodução por outros pesquisadores, que é essencial na metodologia científica dos experimentos.

A simulação resolve alguns dos problemas encontrados na experimentação real. Na simulação:

- (i) não é necessário desenvolver uma aplicação real;
- (ii) existe a possibilidade de controlar o ambiente e de repetir os experimentos;
- (iii) os cenários para testes não são limitados; e

- (iv) há possibilidade de o experimento ser repetido por outros pesquisadores (CLAUSS et al., 2011).

A tradução livre de simulação é a tentativa de predizer o comportamento de algum sistema através da criação de um modelo matemático aproximado (HOWE et al., 1993).

Algumas questões técnicas que surgem para a solução de problema em grade computacional são:

- (i) como será a distribuição dos dados entre os processadores?
- (ii) como será a comunicação entre os hosts?
- (iii) como será feito o mapeamento dos dados?
- (iv) se um host falhar, como será a recuperação dos dados?

Essas são apenas algumas questões que podem surgir com um problema dessa natureza. As soluções técnicas para essas questões existem, mas qual é a mais eficiente? Às vezes, o problema possui uma determinada particularidade que uma técnica seja menos indicada que outra. No entanto, esse comportamento somente é observado quando a aplicação já foi toda implementada.

Com a simulação, essas questões podem facilmente serem avaliadas com um esforço bem menor e com menor custo. As simulações permitem que diversas configurações sejam testadas de modo a permitir que ajustes nas estratégias sejam efetuados com o objetivo de garantir a eficiência do método aplicado. É mais fácil, rápido e eficiente, pensar em uma estratégia e fazer a simulação antes de implementá-la. Deste modo, é possível implementar a estratégia, testar, debugar para, depois, finalmente implementá-la.

Fazer experimentações por simulação de PAD é uma opção bastante interessante visto que as possibilidades de testes podem ser extrapoladas com a utilização de diversos parâmetros e cenários. No SimGrid, ela pode ser desenvolvida apenas com o propósito de estudo de uma solução ou problema. Mas a simulação pode ser convertida em uma aplicação real. Na continuidade, é apresentada a arquitetura deste *framework*, com o intuito de analisar e entender a base de seu funcionamento.

O SimGrid é um *framework* de simulação para aplicações distribuídas (CASANOVA et al., 2014). Ele pode ser usado para pesquisas em *Clusters*, Grades Computacionais, Algoritmos P2P, Computação Voluntária e Estratégias e Heurísticas para Algoritmos Paralelos.

Conforme (CASANOVA et al., 2014), as principais características do SimGrid são: (i) o kernel da simulação é escalável, extensível e implementa alguns modelos simulados, o que permite a simulação de inúmeras topologias de rede, computação

dinâmica, recursos de rede disponíveis, bem como suas falhas; (ii) API que possibilita, a pesquisadores em computação distribuída, o desenvolvimento rápido de simulações; (iii) APIs para o desenvolvimento de aplicações distribuídas que podem ser executadas em ambiente simulado ou real.

A literatura aborda sobre diversos ambientes para o desenvolvimento de pesquisas e aplicações de computação distribuída, como pode ser observado na Tabela 1.

Tabela 1: Comparação de ferramentas existentes para experimentação de larga escala em pesquisa de computação distribuída.

Simuladores	CPU	Disco	Rede	Aplicação	Requisito	Configurações	Escalabilidade
Grid'5000	direto	direto	direto	direto	acesso	fixado	$<$ 5000
PlanetLab	virtualizado	virtualizado	virtualizado	virtualizado	nenhum	não controlado	$<$ 850
ModelNet	-	-	emulação	emulação	muito material	controlado	100 nós / host real
MicroGrid	emulação	-	△ fino	emulação	nenhum	controlado	Poucos 100
ns2	-	-	△ fino	grosseira △	C++ e Tcl	controlado	$<$ 1000
SSFNet	-	-	△ fino	grosseira △	Java	controlado	$<$ 100 000
GTNetS	-	-	△ fino	grosseira △	C++	controlado	$<$ 177 000
ChicSim	grosseira △	-	△ fino	grosseira △	C	controlado	milhares
OptorSim	grosseira △	quantidade	△ matemática	grosseira △	Java	controlado	Poucos 100
GridSim	grosseira △	△ fino	△ fino	grosseira △	Java	controlado	Poucos 100
PlanetSim	-	-	tempo constante	grosseira △	Java	controlado	100 000
PeerSim	-	-	-	máquina de estado	Java	controlado	1 000 000
SimGrid	grosseira △	-	△ matemática	△ emulação	C or Java	controlado	Poucos 10 000

△ = simulação de eventos discretos

Os autores do SimGrid são Henri Casanova, da Universidade do Havaí; Arnaud Legrand, do Laboratório LIG; e Martin Quinson, da Universidade de Nancy; além de possuir diversos colaboradores. A primeira versão do SimGrid surgiu em 1999 quando Casanova tornou-se membro do grupo de pesquisa AppleS do Departamento de Ciência da Computação e Engenharia da Universidade da Califórnia.

A principal linha de interesse desse grupo eram algoritmos de escalonamento para aplicações científicas em ambientes distribuídos e heterogêneos. No desenvolvimento dos projetos de pesquisa, Casanova sentiu a necessidade de executar seus experimentos através de simulação ao invés de um ambiente real.

Nesse momento, Legrand e Casanova desenvolveram um simulador ad-hoc, observando que os pesquisadores do AppleS eventualmente precisavam executar simulações, implicando dessa forma que parte de seus códigos desenvolvidos para ambiente real fossem reescritos. Casanova, então, fez algumas alterações no simulador desenvolvido em conjunto com Legrand:

- (i) tornou o simulador mais genérico; e
- (ii) disponibilizou uma API simples. Essa versão foi chamada de SimGrid v1.0 (SG) descrita em (CASANOVA, 2001).

Em 2001, Legrand inicia a sua tese de doutorado. Seu trabalho tinha como objeto de investigação as heurísticas de escalonamento descentralizado. A versão 1.0 do SimGrid dava esse suporte; no entanto, usar essa versão seria muito complicado além de possuir um escopo muito restrito. Então, Legrand adicionou no topo do SG uma

camada denominada de MSG (Meta-SimGrid) que é apresentada em (LEROUGE; LEGRAND, 2002). Foi incluído nessa camada o suporte a threads, permitindo a execução de processos de computação e comunicação de forma independente, viabilizando dessa forma, o modo assíncrono. Na Figura 2, são apresentados os componentes do SimGrid, em seguida, um detalhamento de sua arquitetura é exposto.

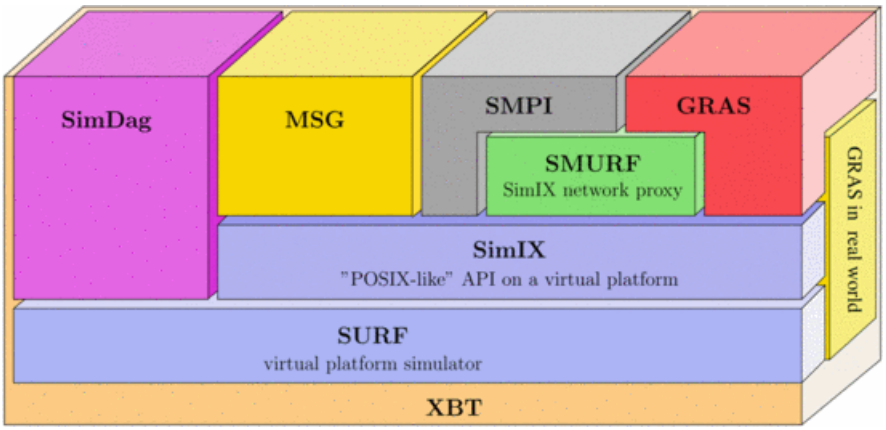


Figura 2: Visão Arquitetural do SimGrid.

O SimGrid possui quatro APIs (CASANOVA et al., 2014):

- SimDAG é herança do SimGrid v1.0, é projetada para trabalhar com heurísticas de escalonamento para aplicações do tipo *task graphs*;
- MSG permite o estudo de aplicações do tipo *Concurrent Sequential Processes* (CSP);
- GRAS permite desenvolver aplicações distribuídas reais;
- SMPI permite simular código MPI diretamente; Os módulos mais internos são (CASANOVA et al., 2014):
- XBT é um módulo *toolbox* utilizado em todo o software. Ele disponibiliza funções que implementam recursos como log, tratamento de erro, suporte a configuração, portabilidade e estruturas de dados;
- SURF é o módulo onde está implementado o kernel da simulação;
- SIMiX é um módulo ponto de extensão do SimGrid, pois provê uma interface POSIX-like que facilita o desenvolvimento de APIs para simulação, como por exemplo, openMP;
- SMURF permite a distribuição dos processos da simulação por um cluster, pois, antes desse módulo, as simulações ficavam limitadas a memória local.

Os principais conceitos do SimGrid são (QUINSON, 2006), (LEGRAND; MARCHAL; CASANOVA, 2003):

- *Agents:* são processos individuais que fazem parte da aplicação distribuída;
- *Sockets/Channel:* é o canal de comunicação entre os Agents;
- *Messages:* representa as informações que são trocadas entre os *Agents*;
- *Callbacks:* são funções definidas por usuário que são executas automaticamente de acordo com o tipo de mensagem recebida;
- *Link:* provê a abstração para o roteamento entre hosts.

Segundo (CASANOVA et al., 2014), categorizam as API de alto nível SimDAG, MSG, GRAS e SMPI em duas: para pesquisadores e para desenvolvedores. A razão para isso é que o SimDAG e MSG estão relacionadas com os estudos de estratégias e heurísticas para problemas de escalonamento. O GRAS e o SMPI permitem que aplicações reais sejam executadas no modo de simulação e no ambiente real sem alteração do código fonte.

A MSG foi desenvolvida para modelar processos sequenciais concorrentes, modelar problemas teóricos e comparar heurísticas. Inicialmente essa API foi desenvolvida para o estudo de escalonamento; no entanto, tornou-se aplicável em outros contextos, como, por exemplo, para computação voluntária. Na versão 3.3 foi incluída a versão jMSG, que permite o uso da API em Java (CASANOVA et al., 2014).

Para o desenvolvimento de aplicações de Grade e P2P, o SimGrid disponibiliza o *Grid Reality And Simulation* (GRAS). A simulação desenvolvida nesse módulo pode facilmente ser disponibilizada no ambiente real. Basta o desenvolvedor fazer a compilação com a uma das bibliotecas implementadas simulada ou real (CASANOVA et al., 2014). O GRAS foi projetado para construir infraestrutura distribuída, através de serviços específicos, para aplicações distribuídas e *middlewares* (QUINSON, 2006). O GRAS baseia-se no paradigma *active message* e as aplicações oriundas dele seguem o modelo *event-driven*.

O módulo SMPI permite que qualquer código MPI seja executado em modo de simulação sem nenhuma alteração. O SimGrid consegue fazer isso interceptando a primitivas MPI.

O kernel do SimGrid está implementado no SURF. Nesse módulo está presente o código responsável pela simulação e foi desenvolvido com dois objetivos principais: ser completamente modular para permitir o desenvolvimento de outros paradigmas de recursos, facilitando, dessa forma, a extensão do SimGrid; e ser o mais otimizado possível para que o desempenho da simulação não seja prejudicado (CASANOVA et al., 2014).

A simulação possui dois elementos principais:

- (i) um conjunto de recursos link, CPU, por exemplo; e
- (ii) uma aplicação que faz uso dos recursos, seja para executar determinadas ações, transferir arquivo ou ainda executar um cálculo, são alguns exemplos.

O papel do SURF é calcular o tempo gasto para executar as ações. Para isso, o kernel solicita a cada modelo que controla os recursos associados, quando a ação será finalizada; calculando, assim, o tempo mínimo de término daquelas ações.

O SURF atualiza com esse valor o tempo da simulação e o estado das ações e recursos, além de informar ao usuário as ações já finalizadas. Nesse momento, o código de simulação escrito pelo usuário tem a oportunidade de iniciar a execução de nova ação, comunicação, ou computação, fazendo com que o SURF inicie outro ciclo de simulação (CASANOVA et al., 2014). Esse *loop* é executado até que todas as ações tenham sido finalizadas.

A precisão da simulação é abordada em (CASANOVA; LEGRAND; QUINSON, 2008), onde foram apresentados percentuais de erros e identificados em quais modelos e situações ocorreram. Já os cálculos efetuados no SURF estão descritos nos trabalhos em (CASANOVA; MARCHAL, 2002) e (QUINSON; BOBELIN; SUTER, 2010).

2.2 LÓGICA FUZZY

Este capítulo aborda os fundamentos básicos relativos à LF, definindo e diferenciando os conceitos de função característica e Função de Pertinência (FP). Discorre sobre os conceitos dos conjuntos fuzzy tipo 1 e tipo 2, aborda sobre os números fuzzy, conectivos fuzzy, como negações, função N-dual e implicações.

Na teoria clássica dos conjuntos, um elemento pertence ou não pertence a um determinado conjunto. A pertinência ou não pertinência do elemento, pode ser interpretada como uma função característica dada pela definição:

Definição 1 ***Função Característica:*** *Seja* U *um conjunto universo não vazio* $(U \neq \emptyset)$*,* A *um subconjunto de* U *e* x *um elemento de* U*. Defini-se função característica* $\chi_A(x) : U \to \{0, 1\}$

$$\chi_A(x) = \begin{cases} 1, & \textbf{\textit{se}}\ x \in A, \\ 0, & \textbf{\textit{se}}\ x \notin A. \end{cases}$$

Dessa forma, χ_A é uma função cujo domínio é U e a imagem está contida no conjunto $\{0, 1\}$, com $\chi_A = 1$ denotando que o elemento x está em A, e $\chi_A = 0$ denotando que x não é elemento de A. Logo, a função característica descreve completamente o

conjunto A, definindo quais elementos do conjunto universo U são também elementos de A.

No entanto, existem casos em que a pertinência entre elementos pertence ou não a um conjunto. Sistemas que modelam incertezas, por exemplo, nem sempre possuem fronteiras de pertinência bem definidas (BARROS L. C.; BASSANEZI, 2010; ROSS, 2004; SILER; BUCKLEY, 2005; CARLSSON, 2002).

2.2.1 Conjuntos Fuzzy Tipo 1

Na teoria dos conjuntos fuzzy, a função de inclusão de um elemento a um determinado conjunto flexibilizado, um elemento pode pertencer parcialmente a um determinado conjunto, ao invés de simplesmente pertencer ou não pertencer. Uma função define o grau de pertinência de um determinado elemento em um conjunto fuzzy, considerando um universo de discurso. Formalmente:

Definição 2 ***Função de Pertinência:*** *(ZADEH, 1965) Seja U um conjunto universo não vazio $(U \neq \emptyset)$. Um conjunto fuzzy A em U é caracterizado pela função de pertinência $\mu_A(x) : U \rightarrow [0, 1]$ sendo $\mu_A(x)$ o grau de pertinência do elemento x no conjunto fuzzy A para cada $z \in U$.*

O valor $\mu_A(x) \in [0, 1]$ denota o grau com que o elemento x de U está contido no conjunto fuzzy A; $\mu_A = 0$ e $\mu_A(x) = 1$ denotam, respectivamente, a não pertinência e a pertinência completa de x ao conjunto fuzzy A.

A definição de conjunto fuzzy foi obtida através da extensão da função característica $\chi_A : U \rightarrow \{0, 1\}$, cujo contradomínio $0, 1 \subseteq [0, 1]$. Tendo isso em vista, pode-se dizer que um conjunto clássico é um caso particular de um dado conjunto fuzzy, cuja função de pertinência μ_A é sua função característica χ_A.

A partir da Definição 2, pode-se descrever um conjunto fuzzy A em um conjunto universo U como um conjunto de pares ordenados, onde cada elemento genérico x está associado a seu respectivo grau de pertinência $\mu_A(x)$. Esse conjunto de pares ordenados pode ser visto como um conjunto de n-tuplas

$$A = \{(x, \mu_A(x))|x \in U\}.$$

2.2.2 Conjuntos Fuzzy Tipo 2

A teoria dos conjuntos fuzzy tipo 2 foi introduzida por Lotfi Zadeh em (ZADEH, 1975) como uma extensão do conjunto fuzzy tradicional. Seu surgimento está relacionado com a insuficiência da teoria de conjunto fuzzy tradicional em modelar as incertezas inerentes à definição das funções de pertinência dos antecedentes e consequentes em um sistema de inferência fuzzy.

Conjuntos fuzzy tipo 2 são conjuntos fuzzy, cujos graus de pertinência são conjuntos fuzzy tipo 1, e não um único valor pontual.

A teoria de conjuntos fuzzy tipo 2, modela a incerteza oriunda do significado das palavras. Embora a função de pertinência tipo 2 também seja totalmente precisa, esta é composta por uma ”mancha” de incerteza que permite que ela seja trabalhada pelo Sistema Baseado em Regras Fuzzy Tipo 2.

Um conjunto fuzzy tipo 2 $\tilde{A}$ sobre X, é caracterizado por um função de pertinência do tipo 2, $\mu_{\tilde{A}}(x,u)$, onde seu grau de pertinência passa a ser um conjunto de pares ordenados constituídos por conjuntos fuzzy do tipo 1, onde tem-se que o grau de pertinência e os conjuntos fuzzy tipo 2 são valorados intervalarmente conforme Definição 3.

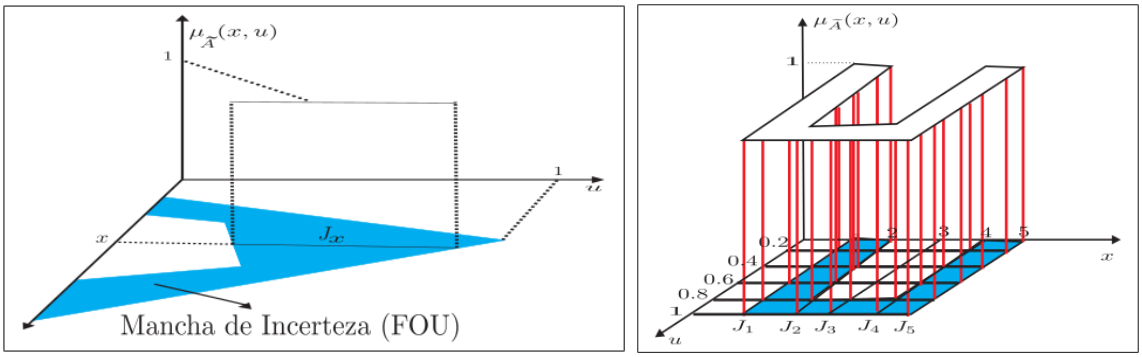
Definição 3 *Um conjunto fuzzy $\tilde{A}$ do tipo 2, é caracterizado por uma função de pertinência $\mu_{\tilde{A}}(x,u)$ do tipo 2, onde $x \in X$ e $u \in J_x \subseteq [0,1]$, ou seja:*

$$\tilde{A} = \{((x,u), \mu_{\tilde{A}}(x,u)) : \forall x \in X, \forall u \in J_x \subseteq [0,1]\}, \tag{1}$$

onde, $0 \leq \mu_{\tilde{A}}(x,u) \leq 1$.

Definição 4 *Um conjunto fuzzy do tipo 2 é intervalar, quando* $\mu_{\tilde{A}}(x,u) = 1$.

Um meio de representar conjuntos fuzzy do tipo 2 é através da forma geométrica de sua função de pertinência. As Figuras 3(a), e 3(b) mostram conjuntos fuzzy do tipo 2 intervalar e suas respectivas áreas coloridas em azul representam, a “Foot print of Uncertainty” (FOU).



(a) (b)

Figura 3: Conjuntos Fuzzy Tipo 2 Intervalar.

Devido ao fato de que o grau secundário dos conjuntos fuzzy do tipo 2 intervalares ser sempre igual a 1, a terceira dimensão acaba não mostrando nenhuma informação adicional; desta forma, o conjunto fuzzy do tipo 2 pode ser representado apenas por sua FOU. As Figuras 4(a), 4(b), 4(c) e 4(d), apresentam conjuntos fuzzy do tipo 2 intervalar, triangular, trapezoidal, gaussiano e “singlenton”, respectivamente.

Definição 5 *O corte vertical de $\mu_{\tilde{A}}(x,u)$ é definido como sendo o plano bidimensional em um dado $x = x'$, cujos eixos são u e $\mu_{\tilde{A}}(x',u)$.*

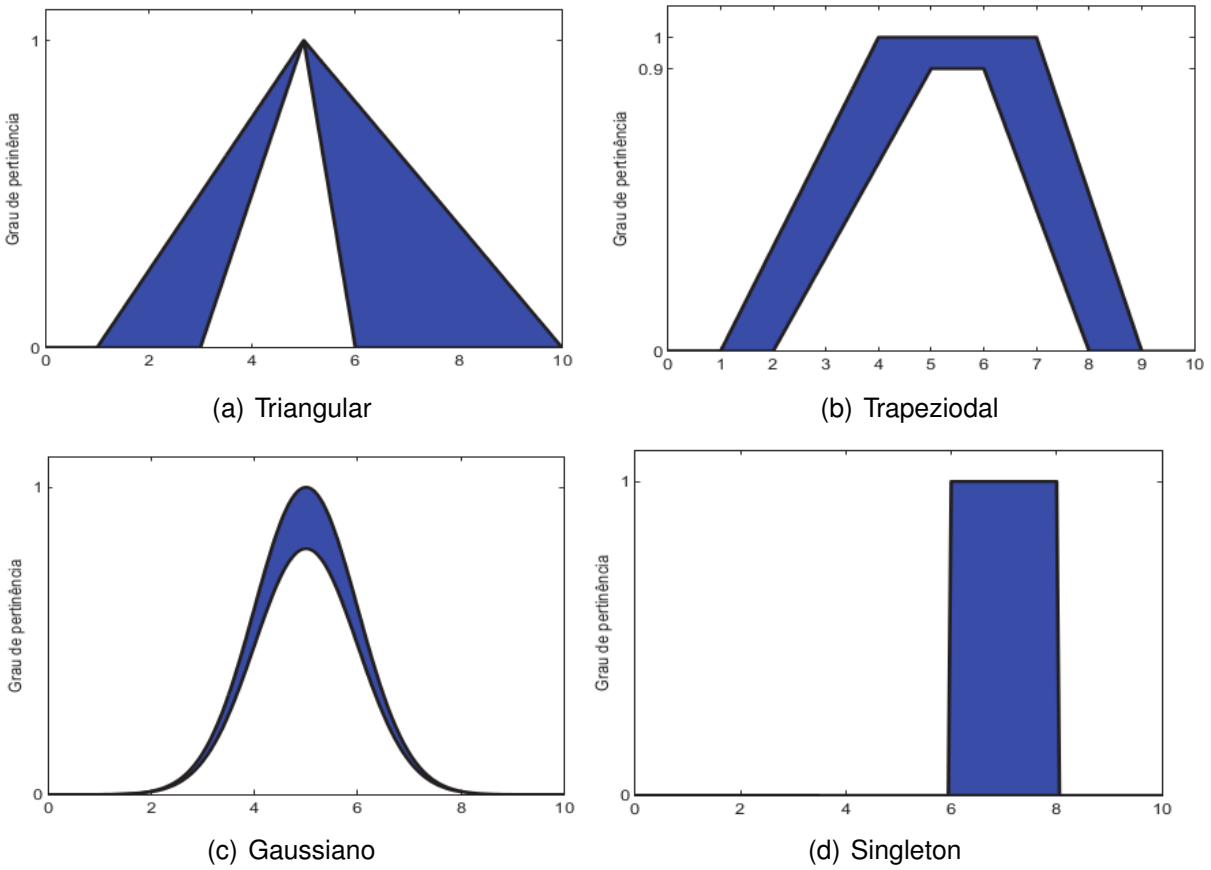


Figura 4: Exemplos de conjuntos fuzzy do tipo 2 intervalares.

Definição 6 *A função de pertinência secundária é o corte vertical de* $\mu_{\tilde{A}}(x,u)$ *em determinado valor de* $x = x^{'}$*. Como mostra, a Figura 5.*

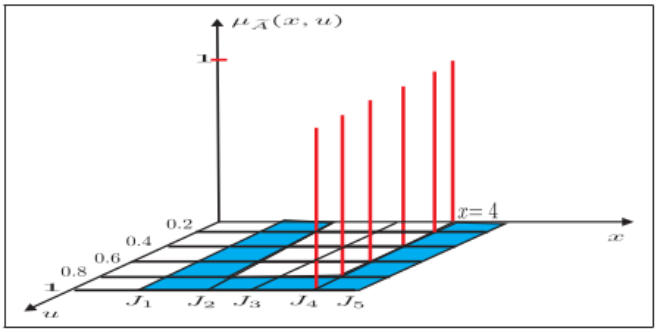


Figura 5: Função de pertinência secundária intervalar em x=4.

Definição 7 *A pertinência primária* J_x *de* x*, é definida como o domínio da função de pertinência secundária para o valor de* x*, com* $J_x = [\underline{J}_x, \overline{J}_x] \subseteq [0,1]$*,* $\forall x \in X$*.*

Definição 8 *A "mancha" de incerteza (FOU) é definida como a união de todas as pertinências primárias, isto é,*

$$FOU(\tilde{A}) = \bigcup_{x \in X} (x, j_x) \tag{2}$$

a FOU é um conjunto fuzzy do tipo 2 delimitada por uma função de pertinência do tipo 1 superior e inferior (RIZOL; MESQUITA; SAOTOME, 2011).

Definição 9 *A função de pertinência superior é representada na forma* $\overline{\mu}_{\tilde{A}}(x), \forall(x) \in X$, *dados por:*

$$\overline{\mu}_{\tilde{A}}(x) = \bigcup_{x \in X} (x, \overline{J}_x), \tag{3}$$

logo, a função de pertinência inferior é representada na forma $\underline{\mu}_{\tilde{A}}(x), \forall(x) \in X$, *dados por:*

$$\underline{\mu}_{\tilde{A}}(x) = \bigcup_{x \in X} (x, \underline{J}_x) \tag{4}$$

onde $\bigcup$ *representa o operador de união.*

A Figura 6 exemplifica uma FOU com suas funções de pertinência superior e inferior.

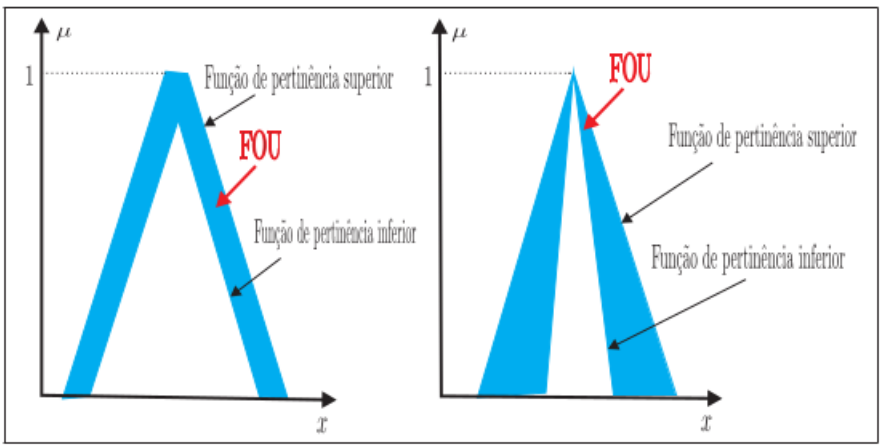


Figura 6: Conjunto fuzzy do tipo 2.

2.2.3 Diferença entre Conjuntos Fuzzy Tipo 1 e Tipo 2 Intervalar

Um exemplo de um conjunto fuzzy do tipo 1, apresentado na Figura 7. Quando apenas os números inteiros são considerados no domínio x, o conjunto fuzzy do tipo 1 pode ser representado como $\{0/2,\ 0.5/3,\ 1/4,\ 1/5,\ 0.67/6,\ 0.33/7,\ 0/8\}$, em que $0/2$ significa que o número 2 possui grau de pertinência 0 no conjunto fuzzy do tipo 1.

Um exemplo das pertinências primárias de um conjunto fuzzy do tipo 2 intervalar discreto é exposto na Figura 8.

Pode-se observar que, ao contrário de um conjunto fuzzy do tipo 1, cujas pertinências para cada x é um número, as pertinências de um conjunto fuzzy do tipo 2 intervalar é um intervalo. Por exemplo, as pertinências primárias dos números 2, 3, 4,

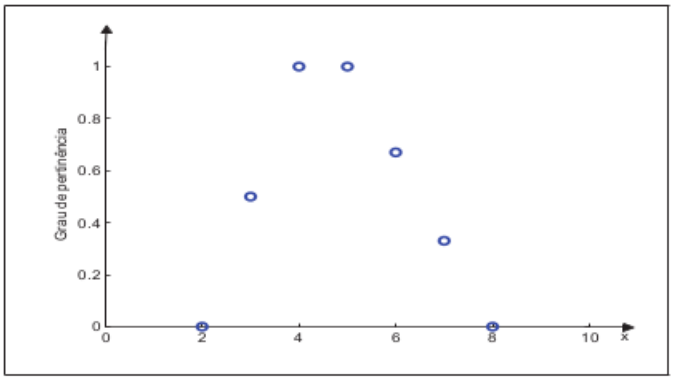


Figura 7: Conjunto fuzzy do tipo 1

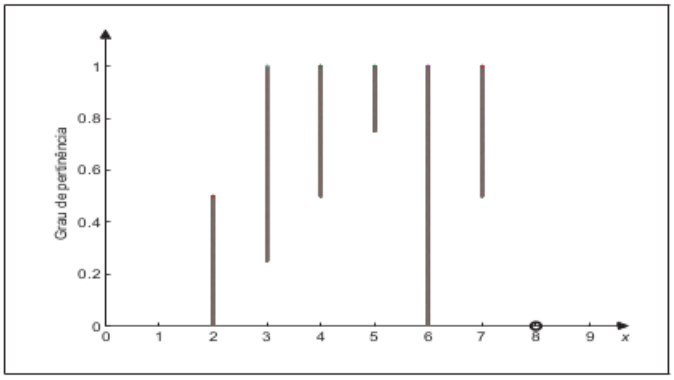


Figura 8: Pertinência primária de um conjunto fuzzy do tipo 2 intervalar discreto.

5, 6 e 7 são intervalos $[0, 0.5]$, $[0.25, 1]$, $[0.5, 1]$, $[0.75, 1]$, $[0, 1]$ e $[0.5, 1]$, respectivamente, e o grau de pertinência primário do número 8 é 0. Neste contexto, um conjunto fuzzy tipo 2 estende um conjunto fuzzy tipo 1 quando consideram-se úteis valores degenerados para os graus de pertinência da função primária $J_x(\overline{J}_x = \underline{J}_x)$.

2.2.4 Números Fuzzy

Assim como no caso clássico, também temos o objetivo de fazer 'contas'. A diferença é que se pretende calcular quantidades imprecisas. Por exemplo, é opinião unânime dizer que o dobro de uma quantidade 'em torno de 15' resulta em outra 'em torno de 30'. Para isto, construíram-se objetos matemáticos que generalizam os números reais. Tais objetos serão chamados de números fuzzy (GEORGE J; BO, 2008).

Definição 10 *Um conjunto fuzzy F é chamado número fuzzy quando o conjunto universo, onde F está definido, é o conjunto dos números reais* $\mathbb{R}$ *e a função de pertinência* $\mu_F : \mathbb{R} \longrightarrow [0, 1]$*, é tal que:*

1. $\mu_F(x)$ *atinge o 1, isto é,* $\sup_{x \in \mathbb{R}} \mu_F(x) = 1.$

2. *$[F]^{\alpha}$ é um intervalo fechado, $\forall \alpha \in (0,1]$.*

3. *O suporte de F é limitado.*

Os números fuzzy mais comuns são triangulares, trapezoidais e senoidais.

Definição 11 *Um número fuzzy F é dito triangular se sua função de pertinência é da forma*

$$\mu_F(x) = \begin{cases} 0, & \text{se } x \leq a, \\ \frac{x-a}{m-a}, & \text{se } a < x \leq m, \\ \frac{x-b}{m-b}, & \text{se } m < x \leq b, \\ 0, & \text{caso contrário.} \end{cases}$$

O gráfico da função de pertinência de um número fuzzy triangular tem a forma de um triângulo, tendo como base o intervalo $[a,b]$ e, como único vértice fora desta base, o ponto $(m,1)$. Deste modo, os números reais $a,\ m\ e\ b$ definem o número fuzzy triangular F, como mostra a Figura 9.

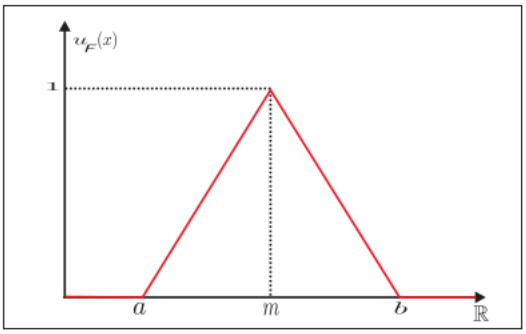


Figura 9: Número fuzzy triangular

Definição 12 *Um número fuzzy F é dito trapezoidal se sua função de pertinência tem a forma de um trapézio e é dado por, como mostra a Figura 10.*

$$\mu_F(x) = \begin{cases} \frac{x-a}{b-a}, & \text{se } a \leq x < b, \\ 1, & \text{se } b \leq x \leq c, \\ \frac{d-x}{d-c}, & \text{se } c < x \leq d, \\ 0, & \text{caso contrário.} \end{cases}$$

Definição 13 *Um número fuzzy tem forma senoidal se a função de pertinência for suave e simétrica em relação ao número real.*

A seguinte função de pertinência verifica as propriedades da Definição 13 para $n,\ a$ e $\delta \in \mathbb{R}$ dados, veja a correspondente representação na Figura 11.

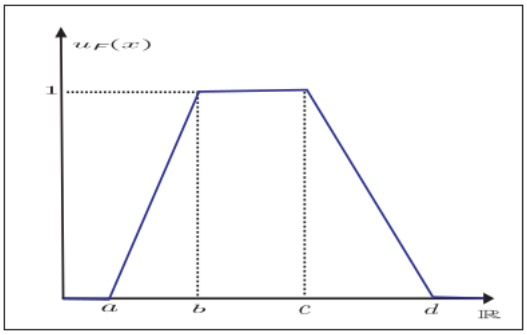


Figura 10: Número fuzzy trapezoidal

$$\mu_F(x) = \begin{cases} \exp(-(\frac{x-n}{a})^2), & \textit{se } n-\delta \leq x < n+\delta, \\ 0, & \textit{caso contrário.} \end{cases}$$

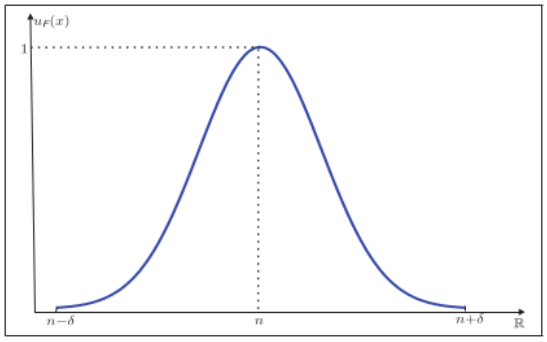


Figura 11: Número fuzzy senoidal

Definição 14 *Seja F um conjunto fuzzy, o Kernel de F é o conjunto de todos os elementos* $x \in U$ *cujo grau de pertinência é 1, isto é,* $Ker(F) = \{x \in U : \mu_F(x) = 1\}$.

Definição 15 *Um elemento* $x \in U$*, no qual* $\mu_F(x) = 0.5$ *é denominado, ponto crossover.*

Definição 16 *Um conjunto F cujo suporte é um único ponto em* U*, com* $\mu_F(x) = 1$*, é denominado, singleton fuzzy.*

Definição 17 *A altura de um conjunto fuzzy F é o valor máximo da pertinência de* x *em* U*, isto é,* $hgt(F) = \sup_{x \in U} \mu_F(x)$.

A Figura 12 apresenta a identificação da altura, kernel e suporte de um conjunto fuzzy.

2.2.5 Operações Aritméticas com Números Fuzzy

As operações aritméticas envolvendo números fuzzy estão estreitamente ligadas às operações aritméticas intervalares. Serão listadas algumas destas operações para intervalos fechados da reta real $\mathbb{R}$.

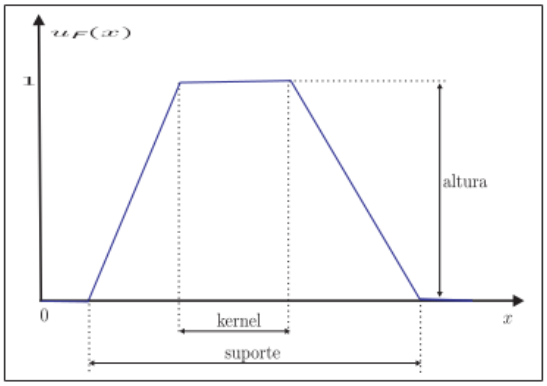


Figura 12: Altura, Kernel e suporte de um conjunto fuzzy

Definição 18 *Sejam λ um número real $A = [a_1, a_2]$, e $B = [b_1, b_2]$ dois intervalos fechados da reta. As operações aritméticas entre intervalos podem ser definidas como (BARROS, 2010):*

(a) $A + B = [a_1 + b_1, a_2 + b_2]$.

(b) $A - B = [a_1 - b_2, a_2 - b_1]$.

(c) $\lambda A = [\lambda a_1, \lambda a_2]$ *se* $\lambda \leq 0$ *e* $\lambda A = [\lambda a_2, \lambda a_1]$, *se* $\lambda < 0$.

(d) $A \cdot B = [minP, maxP]$, *onde* $P = \{a_1 b_1, a_1 b_2, a_2, b_1, a_2, b_2\}$.

(e) $A/B = [a_1, b_2].[\frac{1}{b_2}, \frac{1}{b_1}]$, *se* $0 \notin B$.

2.2.6 Negação Fuzzy

Nesta seção, são tratadas as propriedades pertinentes e necessárias para uma função ser uma negação fuzzy. São apresentados, também, exemplos de negação fuzzy com seus respectivos pontos de equilíbrio. Na última subseção, está a definição de função N-dual.

Uma função $N : N \to U$ é uma negação fuzzy se satisfaz às seguintes propriedades:

N1: $N(0) = 1$ e $N(1) = 0$;

N2: Se $x \leq y$ então $N(x) \geq N(y), \forall x, y \in U$.

Se N também verifica a propriedade involutiva, esta é chamada de negação forte (BUSTINCE; BURILLO; SORIA, 2003; KLEMENT E. MESIAR R., 2003):

N3: $N(N(x)) = x, \forall x \in U$.

Uma negação fuzzy é chamada estrita quando satisfaz as seguintes propriedades:

N4: N é contínua;

N5: Se $x > y$ então $N(x) > N(y), \forall x, y \in U$.

Negações fortes também são negações estritas, mas o contrário não é verdadeiro (BUSTINCE; BURILLO; SORIA, 2003). Por exemplo, a negação fuzzy $N(x) = 1 - x^2$ é estrita, mas não forte.

Observa-se que, se uma negação N é forte, então $N = N^{-1}$.

Observação 1 *Seja N uma negação fuzzy. Se e é um ponto de equilíbrio em N então, pela antitonicidade de N, para cada $x \in U$, se $x \leq e$ então $e \leq N(x)$ e se $e \leq x$ então $N(x) \leq e$ (BEDREGAL, 2010).*

Observação 2 *Seja N uma negação fuzzy. Se e é um ponto de equilíbrio em N e se $x \leq N(x)$ então $x \leq e$, e se $N(x) \leq x$ então $e \leq x$ (BEDREGAL, 2010).*

Todas as negações fuzzy têm no máximo um ponto de equilíbrio, ou seja, se existir esse ponto em uma negação fuzzy N então ele será o único (KLIR, 1993).

A seguir, são descritas algumas negações fuzzy e seu respectivo ponto de equilíbrio.

Exemplificação 1 *Negações correlacionadas com seus respectivos Pontos de equilíbrio.*

1. *Negação padrão forte: $N_S : [0,1] \to [0,1]$ dada por: $N_S(x) = 1 - x$. Ponto de equilíbrio: $x = 0,5$;*
2. *Negação forte: $N : [0,1] \to [0,1]$ dada por: $N(x) = \sqrt{1 - x^2}$. Ponto de equilíbrio: $x = \frac{\sqrt{2}}{2} \approx 0,7$;*
3. *Negação: $N_K : [0,1] \to [0,1]$ dada por: $N_K(x) = 1 - x^2$. Ponto de equilíbrio: $x \approx 0,6$.*

Entretanto, nem todas as negações fuzzy possuem um ponto de equilíbrio. Um exemplo é a negação $N_\perp$ definida a seguir:

$$N_\perp(x) = \begin{cases} 0, & \textit{se } x > 0, \\ 1, & \textit{se } x = 0. \end{cases}$$

2.2.6.1 Função N-Dual

Definição 19 *(DE BAETS, 1997) Seja N uma negação fuzzy forte em $[0,1]$ e $f : [0,1]^n \to [0,1]$ uma função real. A função N-dual de f é a função que, $\forall (x_1, x_2, \ldots, x_n) \in U^n$, está definida pela expressão:*

$$f_N(x_1, x_2, \ldots x_n) = N^{-1}(f(N(x_1), N(x_2), \ldots, N(x_n))). \quad (5)$$

Neste trabalho, tal como considerado em (KLEMENT E. MESIAR R., 2003), quando $N = N_S$ *a Equação (5) é dada por:*

$$f_{N_S}(x_1, x_2, \ldots, x_n) = 1 - f(x_1, x_2, \ldots, x_n) = f(1 - x_1, 1 - x_2, \ldots, 1 - x_n)). \quad (6)$$

Assim, f *e* f_N *são chamadas* ***funções mutuamente duais.***

2.2.7 Relações Fuzzy

O conceito de relação em matemática é formalizado a partir da teoria de conjuntos. Intuitivamente, pode-se dizer que a relação será fuzzy quando opta-se pela teoria dos conjuntos para conceitualizar a relação em estudo (BARROS; BASSANEZI, 2006).

A adoção do tipo de relação depende muito do fenômeno estudado. Porém, a opção pela teoria dos conjuntos fuzzy tem sempre maior robustez no sentido que esta inclui a teoria clássica de conjuntos.

Uma relação clássica indica se há ou não alguma associação entre dois objetos, enquanto que a relação fuzzy além de indicar se há ou não tal associação, indica também o grau desta relação (BARROS; BASSANEZI, 2006).

Em (BENTKOWSKA; KRÓL, 2015) demonstra-se o contexto das relações fuzzy, no que tange a preservação de suas propriedades no processo de agregação, outros trabalhos complementam este estudo formal das propriedades algébricas análogas às relações fuzzy, incluindo classes e aplicações.

Definição 20 *Uma relação clássica* R *é qualquer subconjunto clássico do produto cartesiano* $U_1 \times U_2 \times \cdots \times U_n$*. No produto cartesiano de* $U_1 \times U_2$*, a* R *é denominada de relação binária sobre* $U_1 \times U_2$*. Se* $U_1 = U_2 = \cdots = U_n = U$*, diz-se que* R *é uma relação n-ária sobre* U^n *(BARROS; BASSANEZI, 2006).*

Como a relação R *é um subconjunto do produto cartesiano, então ela pode ser representada por sua função característica.*

$$\chi_R : U_1 \times U_2 \times \cdots \times U_n \to \{0, 1\}, \quad (7)$$

$$\chi_R(x_1, x_2, \ldots, x_n) = \begin{cases} 1, & \textit{se } (x_1, x_2, \ldots x_n) \in R, \\ 0, & \textit{se } (x_1, x_2, \ldots x_n) \notin R. \end{cases}$$

O conceito matemático de relação fuzzy é formalizado a partir do produto cartesiano usual entre conjuntos, estendendo a função característica de uma relação clássica para uma função de pertinência.

Definição 21 *Uma relação fuzzy é qualquer subconjunto fuzzy de* $U_1 \times U_2 \times \cdots \times U_n$*. Assim uma relação fuzzy* R *é definida por uma função de pertinência* $\varphi_R : U_1 \times U_2 \times \cdots \times U_n \to [0, 1]$*.*

Se a função de pertinência da relação fuzzy R for indicada por φ_R, então o número:

$$\varphi_R\{(x_1, x_2, \ldots, x_n) \in [0,1]\}$$

indica o grau com que os elementos x_i, que compõem a n-upla $(x_1, x_2, \ldots, x_n)$ estão relacionados segundo a relação R.

Do pondo de vista de inferência, com o objetivo de tomar alguma decisão, uma relação fuzzy tem grande importância, principalmente na teoria dos controladores fuzzy, é o produto cartesiano (BARROS; BASSANEZI, 2006). Tecnicamente, na teoria dos conjuntos fuzzy, tal operação é similar à intersecção. A grande diferença está nos conjuntos universos envolvidos: enquanto na intersecção os subconjuntos fuzzy são de mesmo universo; no produto cartesiano, eles podem ser diferentes, veja definição a seguir.

Definição 22 *O **Produto Cartesiano** fuzzy dos subconjuntos fuzzy $A_1, A_2, \ldots, A_n$ de $U_1, U_2, \ldots, U_n$, respectivamente, é a relação fuzzy $A_1 \times A_2 \times, \ldots, \times A_n$, cuja função de pertinência é dada por:*

$$\varphi_{A_1 \times A_2 \times \ldots A_n}(x_1, x_2, \ldots, x_n) = \varphi_{A_1}(x_1) \wedge \varphi_{A_2}(x_2) \wedge \cdots \wedge \varphi_{A_n}(x_n),$$

onde $\wedge$ representa o operador de mínimo em [0,1].

Observa-se que se $A_1, A_2, \ldots, A_n$ forem conjuntos clássicos, então o produto cartesiano clássico $A_1 \times A_2 \times \cdots \times A_n$ pode ser obtido pela Definição 22, substituindo as funções de pertinência pelas respectivas funções características dos conjuntos $A_1, A_2, \ldots, A_n$.

As formas mais comuns de se representar uma relação fuzzy binária em $X \times Y$, quando X e Y são finitos, são a tabular e a matricial.

Sejam $X = \{x_1, x_2, \ldots, x_m\}$, $y_1, y_2, \ldots, y_n$ e a relação fuzzy R sobre $X \times Y$, com função de pertinência dada por $\varphi_R(x_i, y_j) = r_{ij}$, para $1 \leq i \leq n$ e $1 \leq j \leq m$.

As representações de R podem ser na forma de tabela ou matriz conforme segue:

$$\begin{array}{c|cccc} \mathcal{R} & y_1 & y_2 & \cdots & y_n \\ \hline x_1 & r_{11} & r_{12} & \cdots & r_{1n} \\ x_2 & r_{21} & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_m & r_{m1} & r_{m2} & \cdots & r_{mn} \end{array} \quad \text{ou} \quad \mathcal{R} = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ r_{21} & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ r_{m1} & r_{m2} & \cdots & r_{mn} \end{bmatrix}.$$

2.2.7.1 Funções de Agregação Fuzzy

Em (BUSTINCE; BARRENECHEA; MOHEDANO, 2004, Definição.2), uma função de *agregação* $M : U^2 \rightarrow U$ satisfaz as seguintes propriedades:

$A1$**:** $M(0,0) = 0$ e $M(1,1) = 1$;

$A2$**:** Se $x \leq z$ então $M(x,y) \leq M(z,y)$, $\forall x,y,z \in U$;

$A3$**:** $M(x,y) = M(y,x)$, $\forall x,y \in U$;

Funções de agregação satisfazendo a propriedade de idempotência

$A4$**:** $M(x,x) = x$, $\forall x \in U$,

são chamadas de *funções de agregação idempotentes*.

Sejam as funções de agregação $\wedge, \vee : U^2 \to U$, respectivamente definidas em (DESCHRIJVER; KERRE, 2005, Definição 4.1) pelas expressões:

$$\wedge(x,y) = \min(x,y); \tag{8}$$

$$\vee(x,y) = \max(x,y), \forall x,y \in U. \tag{9}$$

Se M é uma função de agregação forte ou idempotente, então temos:

$$\wedge(x,y) \leq M(x,y) \leq \vee(x,y), \forall x,y \in U.$$

2.2.7.2 T-normas e T-conormas

As funções de agregação que qualificam as intersecções fuzzy e uniões fuzzy, são geralmente referidas na literatura como t-normas e t-conormas, respectivamente. No contexto da inclusão $\{0,1\} \subseteq [0,1]$, as normas e conormas triangulares são extensões das funções que representam a conjunção e disjunção na lógica clássica, respectivamente.

Definição 23 *(KLEMENT; MESIAR; PAP, 2000) Uma* ***norma triangular*** *(t-norma) é uma função $T : U^2 \to U$, satisfazendo as seguintes propriedades, para todo $u,v,x,y,z \in U$:*

$T1$**:** $T(x,y) = T(y,x)$ *(Comutatividade);*

$T2$**:** $T(x,T(y,z)) = T(T(x,y),z)$ *(Associatividade);*

$T3$**:** $T(x,y) \leq T(u,v)$*, se* $x \leq u$ *e* $y \leq v$ *(Monotonicidade);*

$T4$**:** $T(x,1) = x$ *(Elemento neutro).*

Exemplificação 2 *De acordo com (DUBOIS; PRADE, 2000) a Tabela 2 apresenta os exemplos mais referenciados e utilizados de t-normas.*

Definição 24 *(KLEMENT; MESIAR; PAP, 2000) Uma* ***t-conorma triangular*** *(s-norma) é uma função $S : U^2 \to U$, satisfazendo as seguintes propriedades, para todo $u,v,x,y,z \in U$:*

Nome	Expressão Algébrica de t-normas
Intersecção-Padrão:	$T_M(x,y) = min\ \{x,y\}$
Produto Algébrico:	$T_P(x,y) = x.y$
Intersecção Drástica:	$T_D(x,y) = \begin{cases} 0, & \text{se } x < 1 \text{ e } y < 1 \\ \min\{x,y\}, & \text{caso contrário;} \end{cases}$
Łukasiewicz:	$T_L(x,y) = \max\{x+y-1, 0\}$
Nilpotente Mínimo:	$T_m(x,y) = \begin{cases} 0, & \text{se } x+y \leq 1 \\ \min\{x,y\}, & \text{caso contrário.} \end{cases}$

Tabela 2: Exemplificação de normas fuzzy triangulares

$S1$**:** $S(x,y) = S(y,x)$ ***(Comutatividade);***

$S2$**:** $S(x,S(y,z)) = S(S(x,y),z)$ ***(Associatividade);***

$S3$**:** $S(x,y) \leq S(u,v)$ ***se*** $x \leq u$ ***e*** $y \leq v$ ***(Monotonicidade);***

$S4$**:** $S(x,0) = x$ ***(Elemento neutro).***

Exemplificação 3 *Analogamente, de acordo com (DUBOIS; PRADE, 2000) a Tabela 3 apresenta os principais exemplos de t-conormas.*

Nome	Expressão Algébrica de t-conormas
União Padrão:	$S_M(x,y) = max\ \{x,y\}$
Soma Probabilística:	$S_P(x,y) = x + y - xy$
União Drástica:	$S_D(x,y) = S_D(x,y) = \begin{cases} 1, & \text{se } 0 < x \text{ e } 0 < y \\ \max\{x,y\}, & \text{caso contrário;} \end{cases}\}$
Łukasiewicz:	$S_L(x,y) = \min\{x+y, 1\}$
Nilpotente Máximo:	$S_m(x,y) = \begin{cases} 1, & \text{se } x+y \geq 1 \\ \max\{x,y\}, & \text{caso contrário.} \end{cases}\}$

Tabela 3: Exemplificação de conormas fuzzy triangulares

Logo a seguir, discute-se o conceito de t-norma e t-conorma duais.

2.2.7.3 T-norma e T-conorma Duais

Esta seção apresenta as definições que estabelecem a Dualidade entre T-norma e T-conorma.

Proposição 1 *A função* $T(S) : U^2 \to U$ *é uma t-norma (t-conorma) se, e somente se, existe uma t-conorma* T_N *(t-norma* S_N*) tal que para todo* $(x,y) \in U^2$*, uma das seguintes equivalências são satisfeitas (ZANOTELLI et al., 2015):*

$$T_N(x,y) = N(T(N(x), N(y))), \tag{10}$$

$$S_N(x,y) = N(S(N(x), N(y))). \tag{11}$$

A t-conorma T_N dada pela Equação (10) é denominada de t-conorma derivada de T pela relação de dualidade e, analogamente a t-norma S_N dada pela Equação (11), é chamada de t-norma derivada de S pela relação de dualidade, ambas definidas com respeito à negação fuzzy N.

Proposição 2 *(FODOR; ROUBENS, 1994a; KLEMENT; MESIAR; PAP, 2000) Sejam as t-normas e t-conormas apresentadas na Exemplificação 2 e Exemplificação 3, respectivamente. Então, tem-se que* (T_M, S_M)*,* (T_P, S_P) *,* (T_D, S_D)*,* (T_L, S_L) *e* (T_nM, S_nM) *são pares de t-normas e t-conormas mutuamente duais em relação a* N_S *(negação padrão).*

2.2.8 Implicações Fuzzy

A literatura apresenta uma diversidade de definições e aplicações no assunto de implicações fuzzy e suas duais (BACZYŃSKI, 2004; BACZYŃSKI; JAYARAM, 2008; BALASUBRAMANIAM, 2007; BUSTINCE; BURILLO; SORIA, 2003; FODOR, 1991; FODOR; ROUBENS, 1994b; FODOR, 1995; HORČÍK; NAVARA, 2002; ŁESKI, 2003; RUAN; KERRE, 1993; YAGER, 1983, 1999, 2004). Sendo a implicação fuzzy uma generalização da abordagem clássica, o único consenso em suas definições é que a implicação fuzzy deve ter o mesmo comportamento da implicação clássica quando os valores de entrada da implicação forem uma combinação dos extremos do intervalo $[0,1]$. As implicações estão contidas no conjunto das subimplicações fuzzy, as quais são operadores que não verificam totalmente o comportamento da implicação clássica.

Um operador binário $I : U^2 \to U$ é uma **implicação fuzzy** se I satisfaz as condições de contorno, (**I2** e **I3**) :

I0: $I(1,1) = I(0,1) = I(0,0) = 1$.
I1: $I(1,0) = 0$.
I2: Se $x \leq z$ então $I(x,y) \geq I(z,y)$ (antitonicidade no primeiro argumento);
I3: Se $y \leq z$ então $I(x,y) \leq I(x,z)$ (isotonicidade no segundo argumento);

2.2.9 Sistema Baseado em Regras Fuzzy Tipo 1

Nesta seção, serão abordadas as principais características dos Sistemas Baseados em Regras Fuzzy Tipo 1 (SBRF1), as quais reafirmam que estes modelos garantem o suporte para a construção de um sistema de auxílio ao escalonador.

Como o incremento na complexidade de um sistema, a habilidade de fazer declarações precisas e significativas sobre o seu comportamento diminui, até alcançar um limite além do qual precisão e relevância tornam-se características mutuamente exclusivas.

Conforme Zadeh (ZADEH, 1973), a transcrição acima é definida como o Princípio da Incompatibilidade, e mostra a relevância em utilizar a Lógica Fuzzy para auxiliar na resolução de problemas que tradicionalmente são difíceis de resolver. A ideia básica de um sistema fuzzy considera as funções que mapeiam um valor escalar em um número limitado entre 0 e 1, indicando o grau de pertinência desse valor ao conjunto.

Um sistema fuzzy pode estimar funções de entrada e saída, por meio do uso de técnicas heurísticas, onde um especialista humano, entrevistado para ajudar a formular o conjunto de regras fuzzy, pode articular associações de entrada/saída linguísticas. Assim, sistemas fuzzy podem produzir estimativas de um sistema complexo a partir de variáveis linguísticas familiares da linguagem natural, mas fundamentais em modelos matemáticos. Nesse escopo, a metodologia fuzzy é um método de estimação de entrada e saída que considera modelos matemáticos (SHAW, 1999).

Um Sistema de Inferência considera os seguintes blocos principais:

- **Base de regras (BR):** contendo as regras/proposições fuzzy onde as variáveis antecedentes/consequentes são Variáveis Linguísticas (VLs) e os possíveis valores de uma VL são representados por CFs;
- **Base de dados:** definindo as funções de pertinência dos CFs nas regras fuzzy;
- **Unidade de decisão lógica:** realizando operações de inferência para obter, a partir da avaliação dos níveis de compatibilidade das entradas, com as condições impostas pela BR, uma ação a ser realizada pelo sistema;
- **Interface de fuzzificação:** utilizando as funções de pertinência pré-estabelecidas para mapear cada variável de entrada do sistema em graus de pertinência de cada conjunto fuzzy que representa a variável em questão;
- **Interface de defuzzificação:** transformando os resultados fuzzy da inferência em valores de saída calculados com base na inferência obtida no módulo da Unidade de Decisão Lógica, com as funções de pertinência das VLs da parte consequente das regras para obter uma saída não fuzzy. Nesta etapa, as regiões resultantes são convertidas em valores de saída do sistema.

2.2.9.1 Arquitetura de um Sistema Fuzzy

Esta seção descreve, brevemente, as etapas de estruturação de um sistema baseado na lógica fuzzy. A Figura 13 apresenta um esquema gráfico de uma sistema baseado em regras fuzzy.

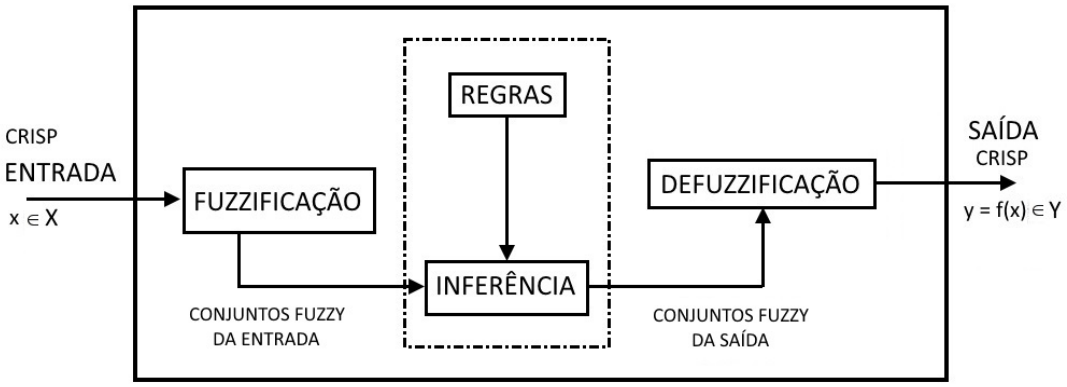


Figura 13: Visão Arquitetural de um Sistema Fuzzy.

2.2.9.2 Fuzzificação

Fuzzificação é um mapeamento de subconjuntos de números reais (em geral discretizado) para o domínio fuzzy. Fuzzificação também indica que há atribuição de valores linguísticos, descrições vagas ou qualitativas, definidas por funções de pertinência às variáveis de entrada.

A fuzzificação também pode indicar uma espécie de *pré-processamento* de categorias ou classes dos sinais de entrada, reduzindo o número de valores a serem processados. Uma menor quantidade de valores processados significa menos complexidade das computações.

As funções de pertinência também podem ser descritas por tabulado de valores numéricos, e um procedimento de consulta a tabelas pode acelerar a etapa de fuzzificação.

Resumindo, têm-se as seguintes subetapas:

- a entrada é um valor numérico;
- a saída está dentro do intervalo real [0,1].

Os dois itens anteriores são dependentes do grau de pertinência, que está relacionado com a definição dos CFs. Para cada valor de entrada é aplicada uma função de pertinência, a qual retornará o grau de pertinência da proposição.

2.2.9.3 Regras e Inferência

As regras podem ser fornecidas por especialistas, em forma de sentenças linguísticas, e se constituem em um aspecto fundamental no desempenho de um sistema de inferência fuzzy.

No caso de controlador fuzzy, o bom desempenho está condicionado às regras que descrevem a estratégia de controle de forma consistente. Extrair regras de especialistas na forma de sentenças condicionais é uma tarefa difícil, por mais conhecedores que eles sejam do problema em questão.

Alternativamente ao uso de especialistas para a definição da base de regras, existem métodos de extração de regras de dados numéricos úteis em problemas de classificação e previsão de séries temporais (KLIR, 2005).

No estágio de inferência, ocorrem as operações com CFs propriamente ditas: combinação dos antecedentes das regras, implicação e *modus ponens* generalizado. Os CFs de entrada, relativos aos antecedentes das regras, e o de saída, referente ao consequente, podem ser definidos previamente ou, alternativamente, gerados automaticamente a partir dos dados.

Nesta etapa, tem-se a aplicação dos operadores fuzzy sendo que a entrada consta de dois ou mais valores, resultantes da fuzzificação. No caso da aplicação do operador de implicação: há uma remodelação nos dados de entrada pela aplicação de uma implicação fuzzy.

Na sequência, tem-se agregação dos resultados das inferências, onde são justapostas todas as saídas fuzzy em um único CF.

Para a elaboração dessas regras é importante ter em mente alguns conceitos importantes. São eles:

- Variáveis Linguísticas (VLs): elas são o centro da técnica de modelagem de sistemas fuzzy. Com elas é possível nomear os conjuntos, e ainda qualificá-los utilizando os qualificadores (muito, regular, pouco). Dessa forma, a modelagem do sistema se torna mais próxima do mundo real.
- Conexões lógicas: do tipo E/OU, para criar a relação entre as variáveis.
- Implicações do tipo: Se $((x$ é $A_1)$ e $(y$ é $B_1))$ então $(z$ é $C_1)$.

Os principais sistemas de inferência fuzzy estão baseados em dois operadores. Para tal, sejam A e B CFs: Existem dois tipos básicos de implicações fuzzy (SHAW, 1999):

1. *modus ponens* (modo afirmativo), frequentemente em controladores fuzzy e sistemas especialistas.[1]

$$
\begin{array}{lll}
 & \text{Premissa 1:} & X = A \\
modus\ ponens: & \text{Premissa 2:} & se\ X = A\ então\ Y = B \\
 & \text{Consequência:} & Y = B
\end{array}
$$

onde $A \subset X\ e\ B \subset Y$.

[1] Sistemas especialistas são sistemas que utilizam uma sequência de regras dependentes para fazer algum diagnóstico. Por exemplo: um sistema de veículo que ao receber a entrada “Motor não liga”, dispara uma sequência de regras ligadas a esta entrada, que verificam as possíveis causas, até chegar a uma conclusão “Verifique a bateria”.

2. *modus tollens:* (modo negativo), operam com base em premissas ou condições, as quais geram uma determinada consequência. Tal tipo é frequentemente utilizado somente em sistemas especialistas.

$$
\begin{array}{lll}
 & \text{Premissa 1:} & Y = não - B \\
modus\ tollens: & \text{Premissa 2:} & se\ X = A\ então\ Y = B \\
 & \text{Consequência:} & X = não - A
\end{array}
$$

O mapa de regras fuzzy relaciona as entradas fuzzy entre si, gerando as saídas fuzzy correspondentes, formando assim a base de conhecimento do sistema. As entradas do mapa são preenchidas durante a identificação do sistema fuzzy quando um operador humano auxilia na identificação da operação e controle do processo.

Na Inferência Fuzzy, também se indicam como as regras são agregadas e combinadas, provendo a construção da região resultante da aplicação das regras.

Na agregação, ou seja, na composição dos vários CFs de entrada em uma regra, as *t*-normas (T_M e T_P) são mais comuns, enquanto que na combinação, ou composição das saídas fuzzy de cada regra, a *t*-conormas S_m e S_P têm sido as mais praticadas. Assim, definem-se as estruturas *max-min* ou *max-produto* para controladores fuzzy. Produto (P) e min (M) são ambos operadores de intersecção fuzzy, implicando em conectivos *AND*.

2.2.9.4 Defuzzificador

No defuzzificador, o valor da VL de saída inferida pelas regras fuzzy será traduzido num valor discreto, ou seja, geometricamente as regiões resultantes do processo de inferência são convertidas em valores precisos para a variável de saída do sistema. O objetivo é obter um único valor numérico discreto (Crisp) que melhor represente os valores fuzzy inferidos da VL de saída.

A defuzzificação é uma transformação inversa que traduz a saída do domínio fuzzy para o domínio discreto. Para selecionar o método apropriado de defuzzificação, pode-se utilizar um enfoque baseado no centroide ou nos valores máximos que ocorrem da função de pertinência resultante.

Existem diversas técnicas de defuzzificação, as principais delas são: centro da área, centro do máximo e média do máximo (SHAW, 1999), resumidas na sequência desta seção.

2.2.9.5 Centro da Área (CoA - Center of Area)

O método centro da área, também conhecido como centro de gravidade, calcula o centroide da área composta que representa o termo de saída fuzzy (μ_{OUT}), esse termo é composto pela união de todas as contribuições de regras. O centroide é um ponto que divide a área de μ_{OUT} em duas partes iguais.

O cálculo do CoA se dá da seguinte forma:

$$u = \frac{\sum_{i=1}^{N} u_i \mu_{OUT}(u_i)}{\sum_{i=1}^{N} \mu_{OUT}(u_i)} \tag{12}$$

onde $\mu_{OUT}(u_i)$ é a área de uma função de pertinência modificada pelo resultado da inferência fuzzy, e u_i é a posição do centroide da função de pertinência individual. Tal equação calcula o centroide composto, para o qual contribuem as funções de pertinência indicadas.

2.2.9.6 Centro do Máximo (CoM - Center of Maximum)

Neste método, os picos das funções de pertinência representados no universo de discurso da variável de saída são usados, enquanto ignoram-se as áreas das funções de pertinência, as contribuições múltiplas de regras são consideradas por esse método.

Os valores não nulos do vetor de possibilidades de saída são posicionados nos correspondentes picos. Assumindo que tais valores representam pesos, o valor de saída defuzzificado (discretizado) é determinado pelo ponto de apoio onde os pesos ficam equilibrados. Assim, as áreas das funções de pertinência não desempenham papel relevante, apenas os máximos são utilizados. A saída discreta é calculada como uma média ponderada dos máximos, cujos pesos são os resultados da inferência. O cálculo do valor defuzzificado é realizado por meio da seguinte equação:

$$u = \frac{\sum_{i=1}^{N} u_i \sum_{k=1}^{n} \mu_{O,k}(u_i)}{\sum_{i=1}^{N} \sum_{k=1}^{n} \mu_{O,k}(u_i)} \tag{13}$$

onde $\mu_{O,k}(u_i)$ indicam os pontos em que ocorrem os máximos (alturas) das funções de pertinência de saída.

Essa abordagem representa um melhor compromisso entre possíveis saídas com multiplicidade de disparo de CFs, ou seja, se três regras forem acionadas, duas impondo uma saída e uma impondo outra saída, por exemplo.

2.2.9.7 Média do Máximo (MoM - Mean of Maximum)

O método Média do Máximo é utilizado em casos onde a função de pertinência tenha mais de um máximo, pois a abordagem CoM não funcionaria bem, devido à necessidade de escolher qual máximo utilizar. Por isso, o uso do método MoM, para tomar-se a média de todos os máximos:

$$u = \sum_{m=1}^{M} \frac{u_m}{M} \tag{14}$$

onde u_m é o m-ésimo elemento no universo do discurso, pressupõe que a função $\mu_{OUT}(u_i)$ tenha um máximo e M é o número total desses elementos. Esse método também é conhecido como solução mais plausível, pelo fato de desconsiderar o formato das funções de pertinência de saída.

2.2.9.8 Controlador Fuzzy

Nesta seção, será abordado o conceito de Controladores Fuzzy e suas principais subdivisões e características.

Controladores fuzzy possuem como principal característica representar um sistema fuzzy por intermédio de regras linguísticas ou equações relacionais, que conectam as entradas imprecisas com as respectivas ações a serem tomadas, com o objetivo de obter a solução mais próxima da ótima para o sistema de acordo com as entradas fornecidas (SHAW, 1999).

Os sistemas de regras fuzzy são úteis para a tomada de decisão e desenvolvimento de controladores. Os valores discretos (não-fuzzy) das variáveis de entrada geralmente podem ser provenientes de sensores das grandezas físicas ou de dispositivos de entrada computadorizados. Um fator de escala pode ser usado para converter os valores reais de entrada para outros que sejam cobertos pelos universos de discurso e pré-definidos para cada variável de entrada. Assim, na fuzzificação, as funções de pertinência contidas na base de conhecimento convertem os sinais de entrada em um intervalo $[0, 1]$ que pode estar associado a rótulos linguísticos.

Na defuzzificação, obtém-se um único valor discreto, utilizável por exemplo, numa ação de controle concreto no mundo real, a partir de valores fuzzy de saída obtidos. Este único valor discreto representa um compromisso entre os diferentes valores fuzzy contidos na saída do controlador.

Esta função é necessária apenas quando a saída do sistema (controlador) tiver que ser interpretada como uma ação de controle discreta, como por exemplo, configurar um seletor numa determinada posição ou mover um motor para uma posição angular pré-definida.

Existem sistemas que não exigem defuzzificação porque a saída fuzzy é interpretada de modo qualitativo. Por exemplo, na fabricação de produtos, a saída fuzzy é comparada com aqueles que correspondem a certos aspectos qualitativos gerados por especialistas que avaliam a qualidade do produto e assim uma saída linguística seria razoável.

2.2.9.9 Controlador Fuzzy Baseado em Regras

Um controlador fuzzy baseado em regras é representado por regras do tipo:

$$\text{SE } pressão = alta \text{ ENTÃO } válvula = abrir\ bastante,$$

conforme citadas na Seção 2.2.9.3, e todas são atividades que ocorrem em paralelo. Assim, um controlador fuzzy baseado em regras “raciocina” com inferência sobre as regras (SHAW, 1999). Quando uma entrada é fornecida, um controlador fuzzy dispara cada regra em paralelo para inferir um resultado ou saída.

Essa operação paralela garante aos controladores fuzzy sua alta velocidade de processamento (por exemplo, num controlador fuzzy industrial típico, de 3 entradas e 1 saída, com 80 regras, o tempo de ciclo para varrer a estrutura de regras pode levar menos de um milissegundo). Sistemas fuzzy raciocinam com conjuntos linguísticos em vez de proposições lógicas bivalentes, conforme visto na forma geral de uma regra fuzzy na Seção 2.2.9.3.

2.2.9.10 Vantagens

Controladores fuzzy baseados em regras têm um grande número de vantagens práticas, que os tornam mais populares entre os softwares de sistemas de desenvolvimento de controladores fuzzy.

- Regras de controle fuzzy são de fácil compreensão pelo pessoal de manutenção, na medida em que são baseadas no senso comum, e o efeito ou resultado de cada regra pode ser facilmente interpretado;
- Todas as funções de controle associadas com uma regra podem ser testadas individualmente. Isso aumenta a facilidade de manutenção, porque a simplicidade das regras permite o uso de pessoal menos treinado;
- Facilidade para controlar sistemas complexos utilizando a combinação de expressões simples da LF;
- Processamento paralelo muito rápido, pois o controlador fuzzy completa a tarefa de processamento sem envolver muitos cálculos, aumentando bastante a velocidade de execução;
- Confiáveis e robustos, apresentando resistência a perturbações externas e desgaste ao envelhecimento de componentes internos.

2.2.9.11 Controlador Fuzzy Paramétrico

Outra estrutura de controle fuzzy é a forma paramétrica das regras fuzzy, que apresenta como ideia principal, utilizar uma mistura entre a abordagem de regras e equações lineares para determinar a saída do controlador.

Utilizando dados de entrada e saída como exemplos, é feita uma estimação de seus coeficientes lineares que, por meio de uma regressão linear, tais coeficientes são encontrados e então são formadas equações de saída para o controlador.

O desenvolvimento de um controlador por meio de equações lineares elimina a necessidade de um operador com experiência, pois se baseia em medições cujos resultados são “aprendidos” pelo sistema. Isto evita problemas associados com a formulação das regras de controle fuzzy baseadas em entrevistas com especialistas humanos.

Os coeficientes das equações lineares são determinados com base em dados de exemplos, por meio de análises de regressão linear e procedimentos estáticos, que posteriormente são ajustados por simulações.

Na fase de estimação, que é posterior a fase de “treinamento”, o sistema estima, reconhece e classifica dados desconhecidos ou incompletos, inferindo soluções e capturando relações entre os dados.

O controlador fuzzy paramétrico é composto por proposições condicionais cujos antecedentes são VLs e cujos consequentes são funções, abordando os problemas com uma combinação entre a descrição baseada em regras e aproximações lineares locais.

Os sistemas fuzzy e as etapas de seu desenvolvimento foram descritas, mostrando os ganhos de se usar CFs na modelagem da imprecisão, incerteza e de definições qualitativas, contornando as dificuldades que surgem no caso da teoria clássica da probabilidade quando da tentativa de definir o estado deste sistema.

2.2.10 Sistema Baseado em Regras Fuzzy Tipo 2

O Sistema Baseado em Regras Fuzzy Tipo 2 (SBRF2) é utilizado em aplicações onde existe incerteza na determinação exata do grau de pertinência e em aplicações onde não existe alta confiança no modelo (RIZOL; MESQUITA; SAOTOME, 2011). O diagrama de blocos do SBRF2 é composto por cinco componentes: (i) fuzzificador, (ii) inferência, (iii) base de regras, (iv) redutor de tipo e (v) defuzzificador. Este sistema é composto por, no mínimo, um conjunto fuzzy do tipo 2 presente em um dos antecedentes ou consequentes que compõem uma das regras que formam o sistema. A descrição de cada bloco do SBRF2 é apresentada na Figura 14.

1 **Fuzzificador**: o bloco fuzzificador transforma o vetor de entrada $x^{'} = (x_1^{'}, x_2^{'}, \ldots, x_I^{'}), i = 1, 2, \ldots, I$, do SBRF2 em conjuntos fuzzy do tipo 2;

2 **Base de Regras**: a base de regras do SBRF2 permanece a mesma forma do tipo 1. A diferença entre SBRF1 e SBRF2 está na natureza das funções de pertinência;

Regra 1: Se $((x_1$ é $\tilde{X}_1^1)$ e $(x_2$ é $\tilde{X}_2^1))$ então y é Y^1;

Regra 2: Se $((x_1$ é $\tilde{X}_1^2)$ e $(x_2$ é $\tilde{X}_2^2))$ então y é Y^2;

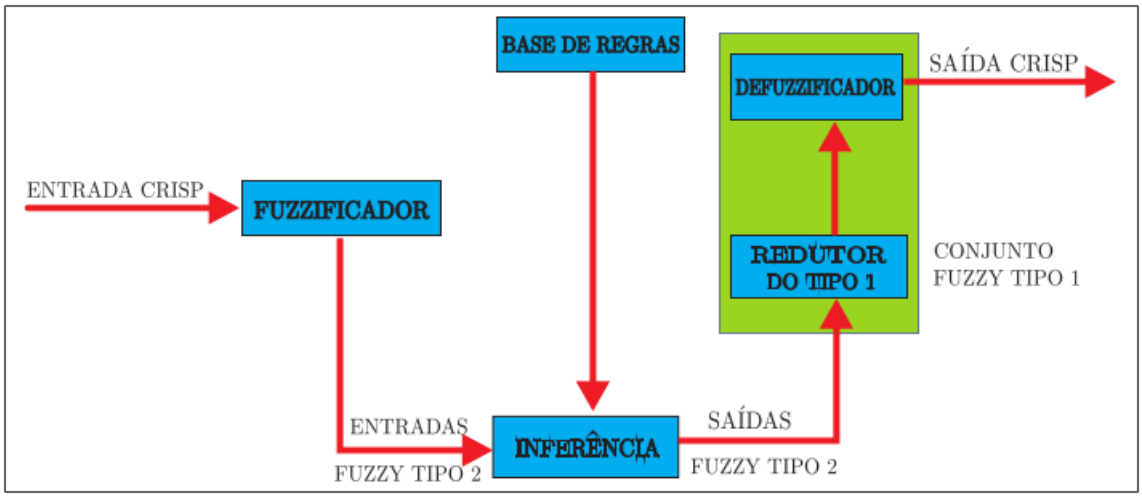


Figura 14: Sistema Baseado em Regras Fuzzy 2.

3 **Inferência:** o bloco de inferência realiza o cálculo do SBRF2 com base nas regras fuzzy;

4 **Redutor do Tipo 1:** o bloco redutor do tipo 1 tem função de transformar um conjunto fuzzy do tipo 2 em conjunto fuzzy do tipo 1, ou seja, procura o melhor conjunto fuzzy do tipo 1 que representa o conjunto fuzzy do tipo 2 e que deve satisfazer a seguinte premissa: quando toda a incerteza desaparecer, o resultado do SBRF2 é reduzido em um SBRF1 (MENDEL, 2001);

5 **Defuzzificador**: a saída defuzzificada do SBRF2 é dada pela média dos pontos limites y_L e y_R, ou seja;

$$y(x) = \frac{y_L + y_R}{2}, \forall x \in \chi, \tag{15}$$

Os valores y_L e y_R, podem ser calculados utilizando o método iterativo de Karnik e Mendel (algoritmo KM) (RIZOL; MESQUITA; SAOTOME, 2011). Da mesma forma, o passo de defuzzificação pode ainda ser obtido através da utilização de um método convencional, tal como o centroide para obter o valor final da inferência.

Este capítulo apresentou os assuntos pertinentes a LF, dando ênfase aos tópicos sobre conjuntos fuzzy tipo 1 e tipo 2, números fuzzy, operações aritméticas com números fuzzy, negação fuzzy, relações fuzzy, funções de agregação, função N-Dual e implicações fuzzy. Apresentou também a arquitetura dos sistemas baseados em LF do tipo 1 e tipo 2.

3 TRABALHOS RELACIONADOS

Este capítulo apresenta os trabalhos que abordam o tema escalonamento de tarefas utilizando a LF necessários para a compreensão e elaboração desta dissertação.

Resume alguns escalonadores de tarefas previamente publicados, e expõem propostas existentes na literatura para lidar com as incertezas presentes no escalonamento de tarefas em grades computacionais ilustrando a necessidade de haver um tratamento preventivo das incertezas.

3.1 Escalonamento Estático de Tarefas Paralelas em Grades

Na pesquisa de (VAHDAT-NEJAD; MONSEFI, 2008) intitulada de Escalonamento Estático de Tarefas Paralelas em Grades Computacionais, foi descrito o projeto de um escalonador fuzzy para tarefas independentes em ambientes distribuídos de grades computacionais. Para obter a escalabilidade, é empregada uma abordagem distribuída, é considerado nível de escalonamento global, e atribuindo cada tarefa a um cluster. Para isto, são considerados os seguintes itens: (i) as necessidades computacionais das tarefas, (ii) número de recursos computacionais para execução das tarefas, e (iii) a quantificação de comunicação das tarefas. A fim de resolver o problema de escalonamento, é aplicado o princípio fuzzy para modelar as fontes que causam incerteza nos estados globais da grade.

O escalonamento ocorre depois que o usuário envia uma tarefa para um dos *clusters*, o processo do escalonamento global é iniciado pelo escalonador do *cluster*. O objetivo é encontrar um *cluster* adequado para atribuição da tarefa. O escalonamento global consiste em dois estágios: na primeira etapa, todos os *clusters* são considerados igualmente, e a prioridade é atribuída a cada um deles. Em seguida, o *cluster* com a prioridade mais alta é selecionado como candidato para a atribuição da tarefa.

Quando o *cluster* selecionado para atribuir a tarefa for remoto, então é realizado um processo de permissão, visando verificar a disponibilidade do *cluster* para o recebimento das tarefas para processamento, uma vez que, uma ou mais tarefas podem terem sido escalonadas para esse *cluster* por seu escalonador local ou outros esca-

lonadores durante o intervalo entre a escolha desse *cluster* para a submissão e envio da tarefa. Caso o retorno da permissão para o envio das tarefas não seja “Ok”, então o escalonador local repetirá o processo de escalonamento global.

Para a caracterização do estado do *cluster*, são considerados dois parâmetros: (i) número disponível de CPUs; e (ii) a carga de rede do *cluster*. O que se entende por carga de rede do *cluster* é o nível de comunicação presente no tráfego da rede *Local Area Network* (LAN) de interconexão do *cluster*, que é representado, nesse caso, pela relação entre zero e um.

Quando uma aplicação chega para ser alocada aos recursos, o escalonador é acionado para atribuí-la a um *cluster*. Para isso, são considerados todos os *clusters* e atribuído três parâmetros: (i) W_1 denominado peso do *cluster* para o número de máquinas, determinando o grau correspondente entre o número de CPUs de baixa carga disponível no *cluster*, e o número de tarefas da aplicação, (ii) W_2 chamado de peso do *cluster* para carga da rede, considera os requisitos de comunicação da aplicação e a carga de rede do *cluster*, (iii) W_3 também chamado de peso da grade para a utilização da rede, referindo-se à utilização da *Wide Área Network* (WAN) e o tamanho da aplicação.

Foram realizados testes de simulação para ajustar o coeficiente desses pesos na agregação da equação da prioridade. A fórmula final obtida para calcular a prioridade de um *cluster* em relação a uma tarefa é dada pela Equação (16):

$$Prioridade = 0,5W_1 + 0,2W_2 + 0,3W_3 \tag{16}$$

Depois de calcular a prioridade de todos os *clusters*, o escalonador atribui a tarefa ao *cluster* com prioridade máxima.

A carga dos nós do *cluster* é um atributo dinâmico e é calculado pela média das cargas relatadas atualmente pelo uso da CPU no nó. O escalonador divide os nós de um *cluster* em nós de baixa carga, média e alta. Um nó cuja carga é menor que 0,3 é considerado de baixa carga, um nó com carga entre 0,3 e 0,6 é visto de carga média, e um nó com carga maior que 0,6 é classificado como um nó de alta carga. Para o i-ésimo *cluster*, L_i, M_i, H_i mostram o número de nós de carga baixa, média e alta, respectivamente. O pseudocódigo apresentado no Algoritmo 1 foi o método usado para calcular o peso do *cluster* em relação ao número de máquinas (W_1).

Quando uma aplicação possui alta taxa de comunicação, a mesma deve ser escalonada para um *cluster* com carga de rede baixa. Para isso, é usada a LF para atribuir um peso ao *cluster*, o que determina a adequação da execução da tarefa no *cluster*. Essa atribuição considera os requisitos de comunicação das tarefas da aplicação, e da carga de rede do *cluster*. O sistema fuzzy baseado em regras é constituído por dois parâmetros de entrada: (i) relação de comunicação das tarefas, (ii) e largura de

Algoritmo 1: Algoritmo para calcular o peso do cluster

se $(L_i > numofTask)$ **então**
  $W_1 = 1 - (\frac{L_i - numofTask}{L_i})^3$
  se $(W_i < 0,5)$ **então**
    $W_i = 0,5$
  fim
fim
senão se $(L_i + M_i) > numofTask$ **então**
  $W_1 = 0,5 - (\frac{numofTask - L_i}{numofTask})^3$
  se $(W_i) < 0,2$ **então**
    $W_i = 0,2$
  fim
fim
senão se $(L_i + M_i + H_i) > numofTask$ **então**
  $W_i = 0,2 - (\frac{numofTask - (L_i + M_i)}{numofTask})^3$
  se $(W_i < 0)$ **então**
    $W_i = 0$
  fim
fim

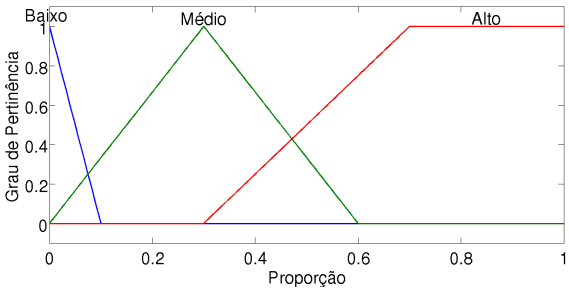


Figura 15: Comunicação e Taxa de Execução

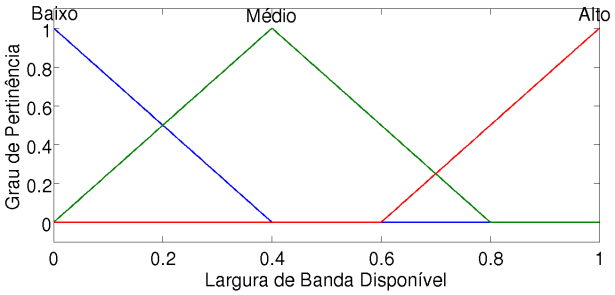


Figura 16: Largura de Banda Disponível

banda disponível do *cluster*, gerando como saída: o peso do *cluster* para carga de rede (W_2).

Quando a relação de comunicação das tarefas da aplicação for próxima de um, isso significa que a aplicação requer alta taxa de comunicação, então um pequeno peso próximo de zero deve ser atribuído a um *cluster* para largura de banda disponível alta, um peso alto perto de um deve ser atribuído ao *cluster* para largura de banda disponível baixa.

As Figuras 15 e 16 retratam os exemplos das funções de pertinência para as entradas, e para a saída, a Figura 17 mostra o peso atribuído à carga de rede do *cluster*. A Tabela 4 resume as regras que mapeiam as entradas para a saída. Foram usados o método de inferência do produto, o método de fuzzificação de singleton, e de defuzzificação média do centro.

Quando uma tarefa é enviada de um *cluster* para outro, a utilização da rede afeta

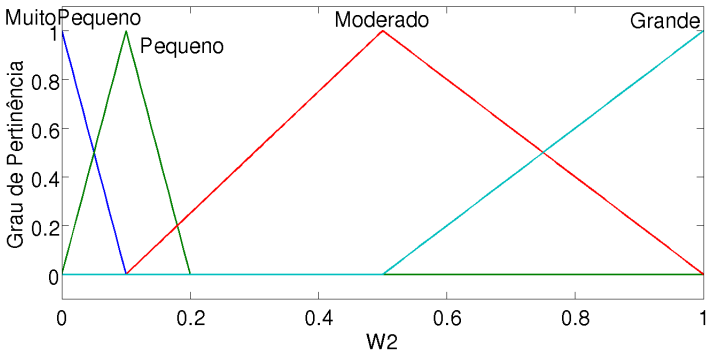


Figura 17: Funções de Pertinência do Peso do Cluster

diretamente o tempo de transmissão. Especificamente, durante o escalonamento global, quando uma tarefa é escalonada de um *cluster* para outro. Para determinar a utilização da rede entre dois *clusters*, é empregado o tempo para a chamada de método remoto. Então considera-se um método remoto de referência que simplesmente retorna um tipo de dado primitivo do escalonador de um *cluster*.

```
int x;
public int getNumber() {return x;}
```

O método *System.currentTimeMillis()* é usado para medir o tempo decorrido durante a chamada *Remote Method Invocation* (RMI) em milissegundos (ms). Na medição, é realizada a verificação do tempo necessário para executar o método remoto, que é em torno de 40 a 50 ms quando a utilização da rede é baixa, e se torna mais longa quando a carga da rede aumenta. Portanto, o método remoto de referência pode refletir aproximadamente a carga da rede da grade. O conjunto fuzzy que representa o tempo de invocação do método remoto é determinado como “Baixo”, “Médio” e “Longo”. A Figura 18 apresenta o gráfico de associação para o tempo de invocação do método remoto.

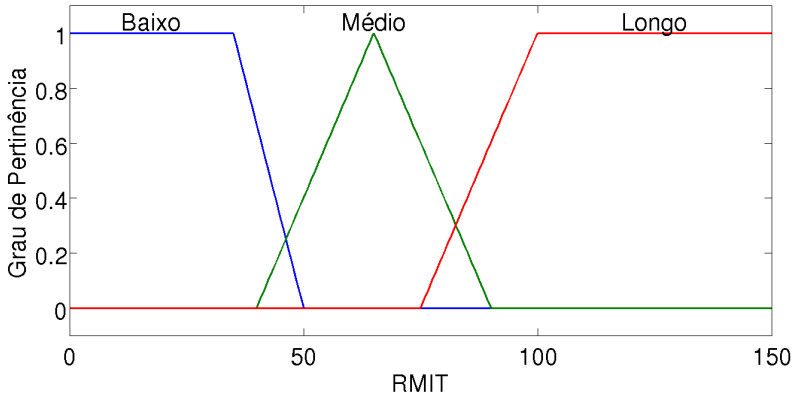


Figura 18: FP da RMIT

Outro fator que influencia o tempo de transmissão da aplicação é o seu tamanho. O tamanho da aplicação é mensurado em megabytes. O conjunto fuzzy para retratar

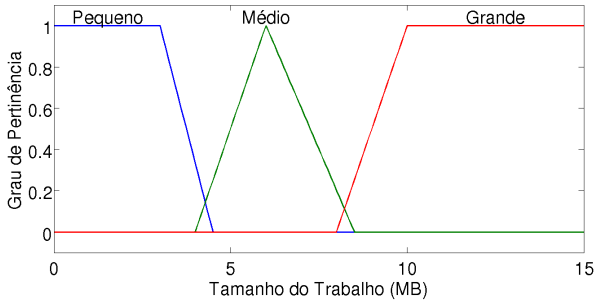


Figura 19: Tamanho do Trabalho (TT)

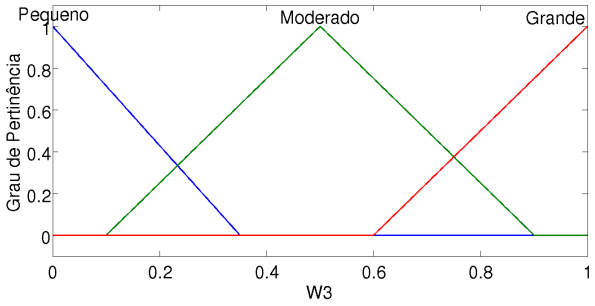


Figura 20: Carga da Rede da Grade

esta medida é definido como “Pequeno”, “Médio” e “Grande”. As Figuras 19 e 20 expõem os graus de pertinência para os tamanhos da aplicação, e o peso da grade para a utilização da rede, respectivamente. A Tabela 4 resume as regras que mapeiam as entradas para a saída (W_3). Foi utilizado o método de inferência do produto, o método de fuzzificação de singleton, e o de defuzzificação média do centro.

Tabela 4: Regras fuzzy usadas na abordagem (VAHDAT-NEJAD; MONSEFI, 2008).

RMIT/TT	Pequeno	Médio	Grande
Baixo	Grande	Grande	Grande
Médio	Grande	Moderado	Pequeno
Longo	Moderado	Pequeno	Pequeno

3.2 Escalonadores de Tarefas Dependentes para Grades

No trabalho de (DANIEL MACEDO BATISTA, 2010) com o título de Escalonadores de Tarefas Dependentes para Grades Robustos às Incertezas das Informações de Entrada, foi proposto o escalonador *IP-FULL-FUZZY*, projetado para lidar tanto com incertezas referente ao tempo de processamento nos computadores, quanto com incertezas referente ao tempo de transferência de dados via rede, tal foco deve-se ao fato de existirem ferramentas de estimativa de largura de banda disponível que fornecem as estimativas através de intervalos.

O escalonador *IP-FULL-FUZZY* baseia-se no escalonador *IPDT-FUZZY*. A diferença entre eles está no fato do *IP-FULL-FUZZY* considerar que a disponibilidade de processamento dos computadores, T_I, e a disponibilidade dos enlaces da grade, T_B, são dadas por números fuzzy, assim como são I e B. O escalonador *IP-FULL-FUZZY* devolve como saída uma lista dos computadores alocados para cada tarefa, os instantes de tempo em que as tarefas devem ser executadas e os instantes de tempo em que os dados de dependência entre as tarefas devem ser transferidos via rede.

Considera-se que os números fuzzy referentes aos valores de $\widetilde{TI}$ e de $\widetilde{TB}$ são números fuzzy triangulares, para reproduzir situações em que o erro na estimativa do peso pode ser tanto positiva como negativa. Assim, a capacidade de processamento disponível em um computador k na grade é representada, utilizando a notação de números fuzzy, por $[\underline{TI_K}, TI_K, \overline{TI_K}]$ onde $\underline{TI_K} = TI_K(1-\frac{\chi}{100})$ e $\overline{TI_K} = TI_K(1+\frac{\chi}{100}).\chi\%$ representa a incerteza em torno da capacidade de processamento disponível. De modo similar, a largura de banda disponível entre os computadores k e l é representada por $[\underline{TB_{k,l}}, TB_{k,l}, \overline{TB_{k,l}}]$ com uma incerteza de $\omega\%$. $\underline{TB_{k,l}}(1-\frac{\omega}{100})$ e $\overline{TB_{k,l}}(1+\frac{\omega}{100})$.

Os valores de χ e de ω traduzem as incertezas a cerca da disponibilidade dos recursos da grade. Quanto menor a variação na disponibilidade de processamento dos computadores menor o valor de χ. De forma similar, quanto menor a variação na largura de banda disponível entre os computadores da grade, menor o valor de ω.

Por conveniência, na formulação do problema resolvido pelo escalonador, foi utilizada a notação $\tau'' = \{1, \ldots, T''_{max}\}$, onde $\tau_{max}(1+\frac{\sigma}{100})(1+\frac{\chi}{100})$ e T_{max} é o maior *makespan* possível para o DAG. Na descrição, é utilizado também o valor T''_{min}, onde $T''_{min} = T_{min}(1-\frac{\sigma}{100})(1-\frac{\chi}{100})$ e T_{min} é o menor *makespan* possível para o DAG. Observa-se que T''_{max} e T''_{min} podem ser calculados diretamente pelos pesos do DAG e do grafo da grade. O escalonador *IP-FULL-FUZZY* é expresso pela formulação matemática apresentada a seguir.

$$1 - \frac{f_n - T''_{min}}{T''_{max} - T''_{min}} \geq \lambda \qquad \text{(FF1)}$$

$$\sum_{t \in \mathcal{T}''} \sum_{k \in V_H} x_{j,t,k} = 1 \qquad \text{para } j \in V_D; \qquad \text{(FF2)}$$

$$x_{j,t,k} = 0 \qquad \text{para } j \in V_D, k \in V_H, \quad t \in \{1, \ldots, \lceil \underline{I_j} \underline{TI_k} \rceil\}; \qquad \text{(FF3)}$$

$$\sum_{k \in \delta(l)} \sum_{t}^{\lceil t - \underline{I_j} \ \overline{TI_l} - \overline{B_{i,j}} \ \overline{TB_{k,l}} \rceil} x_{i,s,k} \geq \sum_{s=1} x_{j,s,l} \qquad \text{para } j \in V_D, ij \in A_D, \text{ para } l \in V_H, t \in \mathcal{T}''; \qquad \text{(FF4)}$$

$$\sum_{j \in V_D} \sum_{s=t}^{\lceil t + I_j \underline{TI_k} - 1 \rceil} x_{j,s,k} \leq 1 \qquad \text{para } k \in V_H, t \in \mathcal{T}'', \quad t \leq \lceil T''_{max} - \underline{I_j} \underline{TI_k} \rceil; \qquad \text{(FF5)}$$

$$x_{j,t,k} \in \{0,1\} \qquad \text{para } j \in V_D, l \in V_H, \quad t \in \mathcal{T}''. \qquad \text{(FF6)}$$

A função objetivo e as restrições do problema resolvido pelo escalonador *IP-FULL-FUZZY* são similares àquelas do *IPDT-FUZZY*, As restrições (FF_i) do escalonador

IP-FULL-FUZZY equivalem às restrições (F_i) do *IPDT-FUZZY*.

A diferença entre as formulações está na utilização dos novos limites para o tempo de execução da aplicação na restrição (FF1) (T''_{max}) e (T''_{min}) ao invés de T'_{max} e T'_{min} e na utilização dos novos limites dos valores de TI e TB nas restrições (FF3), (FF4) e (FF5).

3.3 Escalonador Multi Critérios Fuzzy para Grades

No projeto de (SANTIAGO et al., 2012), denominado de Escalonador Multi Critérios Fuzzy para Tarefas Independentes na Computação em Grade, é exposto um Meta Escalonador Fuzzy (MEF) para grades computacionais que não necessita de qualquer informação sobre o tamanho das tarefas, ou tempo de chegada das tarefas para a realização do escalonamento. Este MEF é do tipo multi-critérios porque se propõe resolver dois objetivos: (i) minimizar o *makespan* de um conjunto de tarefas, e (ii) realizar a atribuição das tarefas de forma equilibrada entre os recursos da grade.

A arquitetura do MEF é constituída no interior de uma camada denominada de Centro de Gerenciamento de Recursos (CGR), tendo em vista que o escalonamento ocorre em três fases principais (SCHOPF, 2002). A primeira fase é a descoberta de recursos (BAGHERI; NAGHIBZADEH, 2007), o que gera uma lista de recursos potenciais. A segunda envolve a publicação de informações sobre esses recursos e a seleção do melhor para enviar uma tarefa. Na terceira, a execução da tarefa. As fases constituintes desta arquitetura proposta são apresentadas na Figura 21.

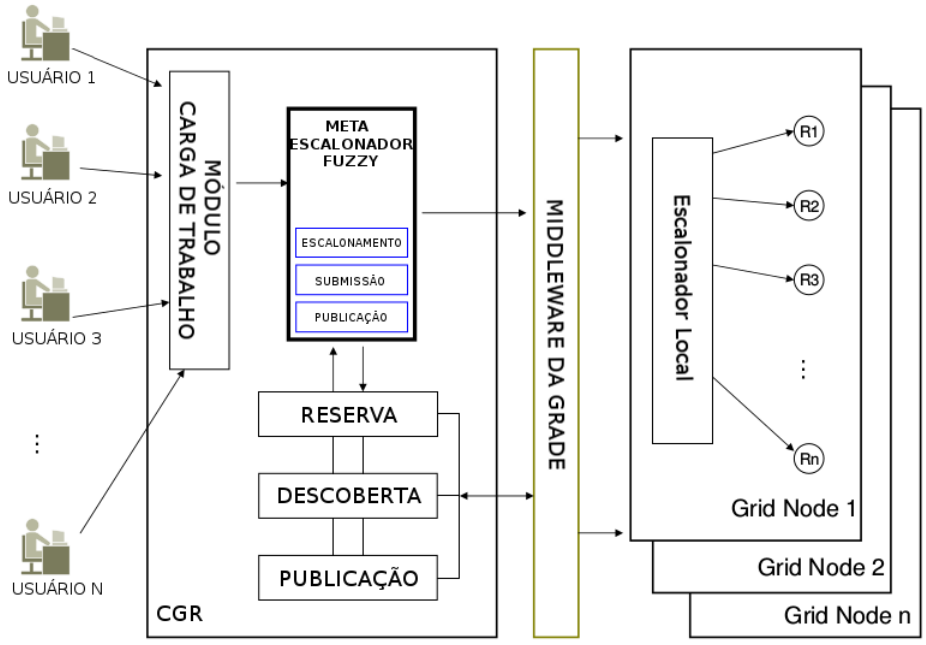


Figura 21: Sistema de Grade com Escalonamento em dois níveis.

O CGR é aplicado na primeira fase, descoberta de recursos, reserva e publicação,

escondendo do usuário a complexidade do sistema de rede. Na segunda fase, o MEF envia as tarefas para os componentes Grid Nodes (GN), ou seja, vários nós e recursos computacionais heterogêneos registrados no sistema que, por sua vez, interagem com os Escalonadores Locais (EL) para implantação das tarefas aos recursos, e com outros componentes do CGR como, por exemplo, monitoramento. O sistema de rede é composto por um conjunto de GN, onde cada GN é composto por vários nós, e recursos computacionais heterogêneos. O MEF interage com o EL de cada GN. Esta interação é fornecida pelo componente CGR, vide Figura 21, que é baseado em um escalonador de dois níveis.

Na formulação do problema, é considerado T exposto na Equação (17), como um conjunto de tarefas BoT a serem atribuídas ao G_i, caracterizado na representação da Equação (18), GN heterogêneo para dupla finalidade: (i) minimizar o *makespan*, e (ii) utilizar eficientemente os recursos da grade visando o equilíbrio na utilização dos mesmos.

$$T_j = (j \in \{1, 2, 3, \ldots, m\}) \tag{17}$$

$$G_i = (i \in \{1, 2, 3, \ldots, n\}) \tag{18}$$

O primeiro objetivo deste trabalho é melhorar o *makespan* no escalonamento de aplicações do tipo BoT. O *makespan* de T é definido como o tempo desde o início da execução da primeira tarefa, até o fim da execução da última tarefa da aplicação BoT. A Equação (19) apresenta a formulação para avaliar o *makespan* de T, em que M é o *makespan*, e S_1 é o tempo inicial de execução da primeira tarefa, e T_m é o tempo final da tarefa.

$$M = max(Tempo_{final} - Tempo_{S_1}) \tag{19}$$

O segundo objetivo, Balanceamento de Carga (BC) é definido através da Equação (20). Com esta equação é verificado qual GN executará o trabalho de forma equilibrada. O BC é avaliado para cada GN quando uma BoT for executada, os valores próximos de zero são os ideais.

$$BC = \frac{RecursoDoNo}{TotalDeRecursos} - \frac{CargaDoNo}{TotalDeCarga} - \frac{1}{n} \tag{20}$$

- RecursoDoNo: O número de recursos de um nó em relação;
- TotalDeRecursos: ao Total de Recursos do sistema de rede;
- CargaDoNo: O número de tarefas executadas no nó em relação;
- TotalDeCarga: ao Total de Tarefas executas no sistema da Grade; e

- n: Número de GN do sistema de rede.

Na execução de uma BoT são necessárias algumas informações sobre o estado atual de cada GN, de posse das informações obtidas através de cada EL para o CGR, o MEF infere para qual GN a BoT deve ser enviada. No GN selecionado, o EL do mesmo atribui a tarefa para o melhor recurso livre. A arquitetura do MEF proposto é exposta na Figura 22.

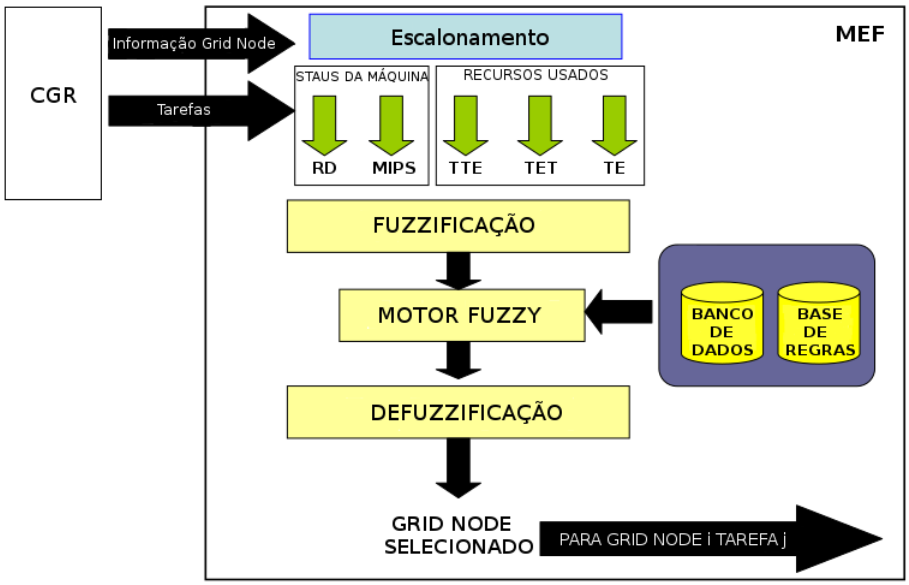


Figura 22: Arquitetura do Meta Escalonador Fuzzy

- Taxa de Recursos Disponível (RD): esta entrada representa o número de recursos disponíveis livres em um GN, formulado através da Equação (21).

$$RD(i) = \frac{RecursosLivres(i)}{TotalRecursos(i)} \tag{21}$$

  Onde i representa um GN do sistema.

- Taxa ociosa de MIPS: esta entrada indica o poder de computação de recursos ociosos de um GN, formalizado na Equação (22).

$$MIPS(i) = \frac{MipsOciosos(i)}{TotalMips(i)} \tag{22}$$

- Tamanho de Tarefas Executadas (TTE): esta entrada é utilizada para indicar o tamanho das tarefas executadas em um GN, em relação a todas as tarefas que foram executadas em todos os nós. A normalização é dada pela Equação (23).

$$TTE(i) = \frac{TamanhoTarefasExecutadas(i)}{\sum_{i=1}^{n} TotalTamanhoTarefasExecutadas} \tag{23}$$

- Tempo de Execução das Tarefas (TET): este parâmetro está relacionado com o tempo total gasto por um nó para executar tarefas BoT em curso, ou o tempo total gasto por todas as tarefas em todos os nós. A Equação (24) é a normalização para esta entrada.

$$TET(i) = \frac{TempoGasto(i)}{\sum_{i=1}^{n} TotalTempoGastoTodosNos} \tag{24}$$

- Tarefas Executadas (TE): representação dos números de tarefas BoT executadas e concluídas por um GN, em relação com o total de tarefas executadas por todos os nós da Grade. Esta formalização é obtida pela Equação (25).

$$TE(i) = \frac{Count(TarefasExecutadas(i))}{\sum_{i=1}^{n} Count(TotalTarefasExecutadas)} \tag{25}$$

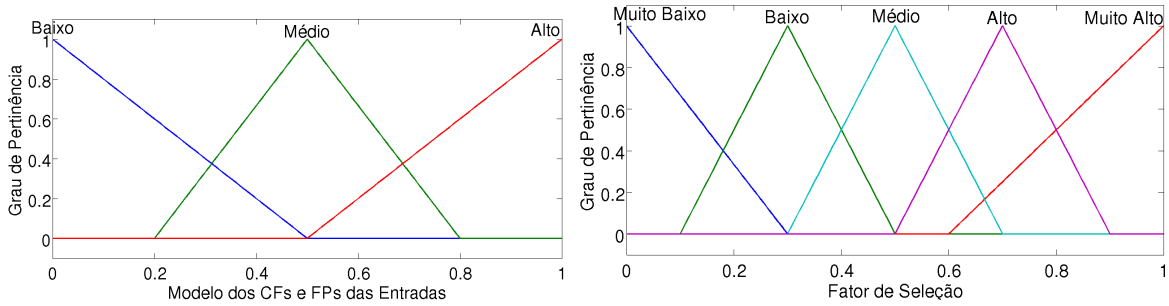
Para realização do balanceamento de carga vários parâmetros são utilizados: (i) o número de recursos de um nó em relação ao total de recursos da grade, (ii) o número de tarefas executadas neste nó, com respeito ao total das tarefas executadas sobre a grade, e (iii) o número de GN do sistema de rede.

O MEF verifica o status atual de todos os nós da grade. De acordo com este status, cada nó tem um Fator de Seleção (FS). O FS do nó é obtido com cinco medidas relativas à situação atual e histórica do nó. Estes parâmetros são definidos no intervalo de 0 e 1, e são classificados em dois grupos, parâmetros relacionados ao estado atual das máquinas: (i) taxa de recursos disponível, (ii) taxa de recursos ociosos, e parâmetros associados ao histórico do uso dos recursos da grade: (i) tamanho de tarefas executadas, (ii) tempo de execução das tarefas, e (iii) histórico de tarefas executadas.

Para estas cinco entradas são calculadas o grau de pertinência para cada Conjunto Fuzzy. As Funções de Pertinência são definidas por Conjuntos Fuzzy na forma triangular. Na Equação (26) é exposto o formalismo utilizado na obtenção do grau de pertinência variando no intervalo de $[0, 1]$. A variável de saída do sistema fuzzy é definida como um FS de um GN. A função de pertinência das entradas e as variáveis de saída são definidas na Figura 23.

$$\mu(x) = \begin{cases} 0, & x \leq a, \\ \frac{(x-a)}{(m-a)} & x \in (a, m], \\ \frac{(b-x)}{(b-m)}, & x \in (m, b], \\ 0, & x \geq 0. \end{cases} \tag{26}$$

O método de inferência utilizado foi o de Mamdani, já para a defuzzificação foi o centroide.



(a) Entradas (b) Saída

Figura 23: Funções de Pertinência para (a) Entrada e (b) Saída

3.4 Algoritmo Distribuído para Alocação de Recursos em Grades

Na pesquisa de (RIBACIONKA, 2013), intitulada de Algoritmo Distribuído para Alocação de Múltiplos Recursos é proposto um algoritmo para o escalonamento em GC que explora a LF sempre que um servidor está impossibilitado de atender a uma solicitação feita por um cliente, encaminhando esta solicitação a um servidor remoto. O algoritmo utiliza o conceito de relógio lógico (LAMPORT, 1978) para garantir o atendimento das solicitações feitas em todos os servidores que compartilham recursos. Este algoritmo segue o modelo distribuído, onde uma cópia do algoritmo é executada em cada servidor que compartilha recursos para seus clientes, e todos os servidores tomam parte das decisões com relação à alocação destes recursos.

A estratégia desenvolvida tem como objetivo minimizar o tempo de resposta na alocação de recursos, funcionando como um balanceador de carga em um ambiente cliente-servidor com alto índice de solicitações de recursos pelos clientes. A eficiência do algoritmo desenvolvido neste trabalho foi comprovada através da implementação e comparação com outros algoritmos tradicionais, mostrando a possibilidade de utilização de recursos que pertencem a distintos servidores por uma mesma solicitação de recursos, com a garantia de que esta requisição será atendida, e em um tempo finito.

Os seguintes aspectos são tratados neste trabalho: (i) utilizar os recursos para execução das aplicações de forma eficiente, (ii) na concorrência a estes recursos, os mesmos são selecionados em função do poder computacional, e (iii) as aplicações são alocadas de forma a evitar *deadlock*[1] e *starvation*[2].

Nesse trabalho, é apresentado um algoritmo totalmente distribuído, onde cada nó participa do funcionamento do sistema executando localmente uma cópia do gerenciador de alocação, recebendo e provendo a alocação e execução das requisições. Cada

[1] Situação onde dois ou mais processos estão esperando por um evento que só pode ser gerado por algum dos mesmos processos em espera.

[2] Quando um processo espera indefinidamente por algum recurso.

requisição pode solicitar um ou múltiplos recursos. Aqueles que são alocados poderão ser de um único *cluster* ou vários servidores e *clusters*, os quais serão selecionados, considerando as suas cargas de processamento e as distâncias em relação ao nó em que foi solicitada a requisição.

A aplicação deste algoritmo em um sistema que precisa alocar múltiplos recursos distribuídos em ambientes de grades computacionais busca prover as necessidades de desempenho quanto ao atendimento de aplicações com e sem dependência de comunicação.

Se vários servidores podem atender a solicitação, utiliza-se de um mecanismo de decisão baseado em LF para seleção, que considera a carga de cada nó. Na alocação dos recursos é também considerado o custo de comunicação decorrente das distâncias entre eles e o nó em que foi solicitada a requisição, e o poder computacional de cada servidor, reduzindo o tempo de resposta.

As metas destacadas para alcançar os objetivos neste trabalho são: (i) desenvolver um algoritmo que explore uma melhor utilização de recursos que estão distribuídos por *clusters* ligados por uma rede de computadores. Este algoritmo tem como característica ser descentralizado, e deve ser executado em cada servidor que compartilha recursos; (ii) implementar neste algoritmo um mecanismo de escolha aos melhores candidatos a ceder recursos; (iii) fazer com que este algoritmo não permita a ocorrência de *starvation* e *deadlock*; (iv) implementar o algoritmo apresentado e realizar testes para a verificação de seu funcionamento e análise de desempenho, utilizando o *framework* de simulação para sistemas distribuídos de grades computacionais SimGrid.

As variáveis observadas para formar as proposições fuzzy são: a (i) latência entre os servidores, e (ii) a potência de processamento de cada servidor. Quando um servidor recebe uma requisição de recursos e não tem recursos próprios para atender esta requisição, o controlador fuzzy é utilizado quando mais de um servidor que participa do grupo de servidores que compartilham recursos pode atender a requisição. A ação fuzzy ajuda a decidir qual é a melhor opção de escolha para qual servidor enviar a solicitação de recursos.

3.5 Abordagem Fuzzy Distribuída para Escalonamento em Grades

Neste trabalho de (KHALILIAN; MIRABEDINI, 2014), denominado de Uma Nova Abordagem Descentralizada Baseada em Lógica Fuzzy para Escalonamento de Tarefas em Grade Computacionais, com o auxílio da teoria fuzzy é apresentado um novo método para o escalonamento de tarefas em sistemas de grades computacionais. O método utiliza a carga média dos nós de todos os clusters, a média do poder computacional determina as funções de pertinência do nó e os parâmetros de entrada das

tarefas do sistema de computação fuzzy, em relação ao valor de saída do sistema fuzzy, com isto, os nós mais adequados são determinados.

Neste esquema, a teoria fuzzy é utilizada por 3 parâmetros para selecionar os nós mais adequados para o escalonamento: o (i) poder de processamento dos nós, (ii) o tempo de duração do trabalho atual em execução nos nós, e o (iii) tempo de entrada da primeira tarefa no sistema.

Para concepção deste modelo, quando o nó denominado de super-cluster receber um pedido de um usuário, é calculada a Velocidade Média de Processamento (VMP) e o Tamanho Médio da Tarefa (TMT) de todos os seus nós. Em seguida, o escalonador do cluster atribui a primeira posição ao pedido. O primeiro super-cluster aciona os seus nós subjacentes com a ajuda da teoria fuzzy e, em seguida, ele envia o primeiro pedido a todos os super-clusters. O valor de saída do sistema fuzzy situa-se no intervalo $[0, 1]$.

Valores não-fuzzy são as entradas do sistema de inferência fuzzy, que são usados na etapa de raciocínio fuzzy. O valor de saída do sistema de inferência fuzzy tem um valor não-fuzzy que apresenta se o *cluster* tem o nó adequado para o pedido ou não. O modelo do sistema de inferência utilizado nesta abordagem é o de Mamdani. Na Figura 24 é exposta a arquitetura do Sistema de Inferência Fuzzy empregada neste trabalho.

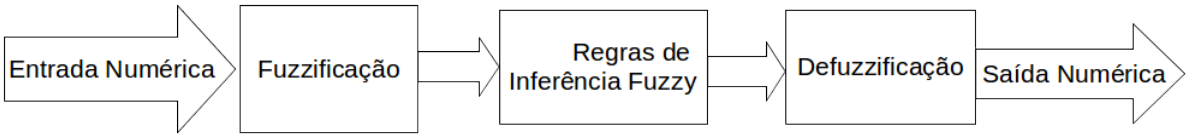
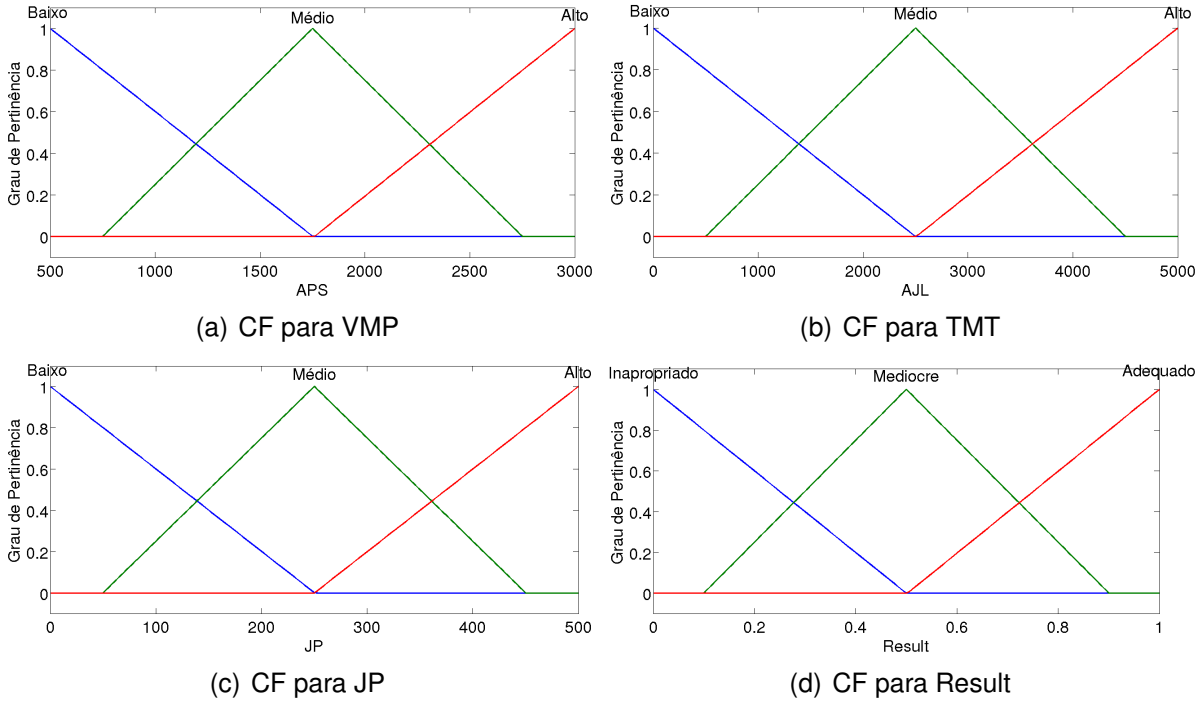


Figura 24: Abordagem do Sistema de Inferência Fuzzy

Para selecionar o conjunto de clusters adequados, as seguintes características são levadas em consideração, primeiramente a VMP, e em seguida, deve se obter um baixo valor para TMT. Esses dois parâmetros são as entrada para Sistema Fuzzy. Nas Figuras 25(a), 25(b), 25(c) as entradas são apresentadas, e na Figura 25(d) a saída do sistema fuzzy é exposta com a modelagem gráfica dos Conjuntos Fuzzy para as variáveis VMP, TMT e JP.

As regras disparadas devem ser integradas de forma que a decisão é realizada de acordo com a agregação das mesmas. O foco das regras de agregação é intensificar os Conjuntos Fuzzy que representam as saídas de cada regra, e são integrados em um único Conjunto Fuzzy. Este único Conjunto Fuzzy é a entrada para a etapa de defuzzificação. A saída da etapa de defuzzificação é um número não-fuzzy que determina o *cluster* mais adequado para o recebimento do pedido. Na Figura 26, é exposto o processo de agregação, e defuzzificação das regras.

No que trata da concepção da base de regras do sistema fuzzy, a mesma foi elaborada levando em consideração as regras apresentadas na Tabela 5.



(a) CF para VMP (b) CF para TMT

(c) CF para JP (d) CF para Result

Figura 25: Funções de Pertinência Entradas (a) VMP, (b) TMT, (c) JP e saída Result

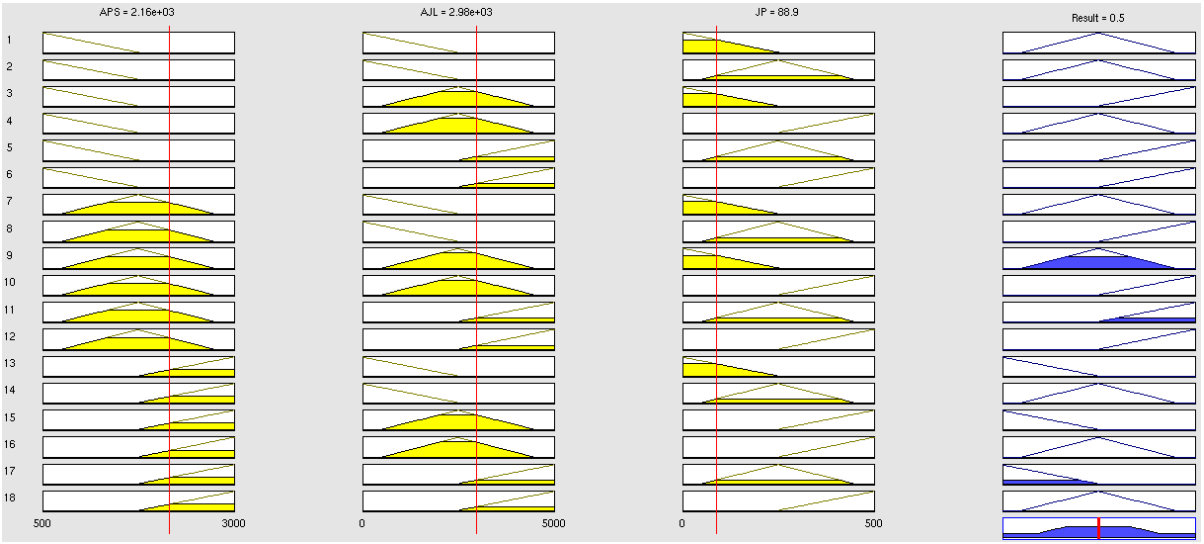


Figura 26: Processo de Agregação e Defuzzificação

Este capítulo relacionou as principais pesquisas conectadas à proposta desta dissertação. Na Seção 4.4, é apresentada a análise referente aos trabalhos relacionados.

Tabela 5: Regras usadas na abordagem de (KHALILIAN; MIRABEDINI, 2014)

VMP	TMT	JP	Result
Baixa	Baixa	Baixa	Mediocre
Baixa	Baixa	Média	Mediocre
Baixa	Média	Baixa	Adequado
Baixa	Média	Alta	Mediocre
Baixa	Alta	Média	Adequado
Baixa	Alta	Alta	Adequado
Média	Baixa	Baixa	Mediocre
Média	Baixa	Média	Adequado
Média	Média	Baixa	Mediocre
Média	Média	Alta	Adequado
Média	Alta	Média	Adequado
Média	Alta	Alta	Adequado
Alto	Baixa	Baixa	Inapropriado
Alto	Baixa	Média	Mediocre
Alto	Média	Baixa	Inapropriado
Alto	Média	Alta	Mediocre
Alto	Alta	Média	Inapropriado
Alto	Alta	Alta	Mediocre

4 FUZZ-GRID: VISÃO GERAL E MODELAGEM FUZZY

Este capítulo trata dos aspectos de concepção, modelagem e prototipação da proposta desta dissertação, a qual foi desenvolvida buscando uma abordagem pouco intrusiva para gerenciamento do escalonamento.

4.1 Tratamento de Incertezas no ETGC via Lógica Fuzzy Tipo 1

Esta seção apresenta os aspectos de concepção, modelagem e prototipação do *fGrid*, um módulo para auxiliar na tomada de decisão no escalonamento de tarefas em grades computacionais baseado em LF tipo 1, e caracterizado principalmente pela capacidade de auxiliar o escalonador diante de incertezas referentes às informações inerentes às grades computacionais.

4.1.1 Modelagem do Sistema Fuzzy para Tratamento das Incertezas

O *fGrid* foi projetado para lidar tanto com incertezas referente ao poder computacional nos computadores disponíveis, quanto com incertezas referentes ao tempo de transferência de dados via rede. Esta seção enfatiza os ganhos que o escalonador com auxílio do *fGrid* alcança em situações as quais ocorram incertezas relacionadas com a disponibilidade de largura de banda ou com a disponibilidade de recursos computacionais.

O *fGrid* considera que a disponibilidade do poder computacional dos computadores e a disponibilidade da largura de banda da grade é dada por números fuzzy. O componente *fGrid* devolve como saída uma lista dos computadores disponíveis e suas respectivas prioridades para auxiliar no escalonamento de tarefas.

Considera-se que os números fuzzy utilizam funções trapezoidais, para reproduzir situações em que o erro na medida extraída da grade pode ser tanto positiva como negativa. Assim, o poder computacional de um computador na grade é representado, utilizando a notação de números fuzzy, que representa a incerteza em torno da capacidade de processamento disponível. De modo similar, a largura de banda disponível entre os computadores também utiliza a notação de números fuzzy, que representa a

incerteza em torno da capacidade de comunicação disponível.

Acerca das incertezas da disponibilidade dos recursos da grade, quanto menor a variação na disponibilidade de processamento dos computadores, menor é a incerteza presente no poder computacional da grade. De forma similar, quanto menor a variação na largura de banda disponível entre os computadores da grade, menor é a incerteza presente no custo de comunicação da grade.

4.1.2 Base de Dados e Definição das Funções de Pertinência

Antes de começar a trabalhar com as informações de entrada, fez-se necessário estudá-las juntamente com um especialista da área, com a finalidade de modelar de forma adequada o comportamento da variável de saída, de acordo com as variáveis de entrada.

Nesta etapa, ocorre o processo de transformação de sentenças linguísticas em Conjuntos Fuzzy, também são determinados quantos e quais CFs cada variável do sistema possuirá, ocorrendo desta forma, o processo de representação gráfico ou funcional do comportamento das variáveis fuzzy, tanto de entrada como de saída.

Conforme a Figura 27, neste projeto a modelagem dos CFs das variáveis são formadas por funções de pertinência trapezoidais, devido ao fato de representarem adequadamente o comportamento das variáveis dentro do sistema fuzzy do tipo 1.

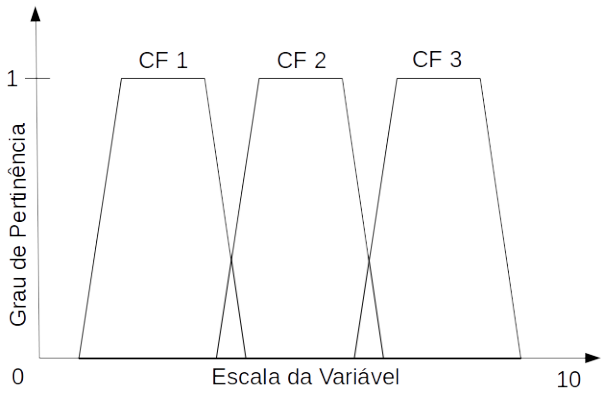


Figura 27: Modelo adotado para as Funções de Pertinência dos Conjuntos Fuzzy.

Outra questão que também teve que ser estudada com auxílio de especialistas, foi definir os pontos de intersecção entre os CFs de uma mesma variável, conforme Figura 27, repetindo este processo em cada variável, de forma que o comportamento do sistema fosse representado corretamente.

Para desenvolver o tratamento de informações da grade, foi feito um levantamento de quais métodos tornariam possível esta funcionalidade ser implementada, de maneira que tornasse mais robusta a etapa de escalonamento. Dentre os métodos estudados, destacou-se a LF pelo fato de lidar com incertezas do mundo real, e como as informações obtidas da grade podem oscilar em um curto espaço de tempo, esta

incerteza também se encontra presente nestes ambientes, conforme é apresentado na Figura 28.

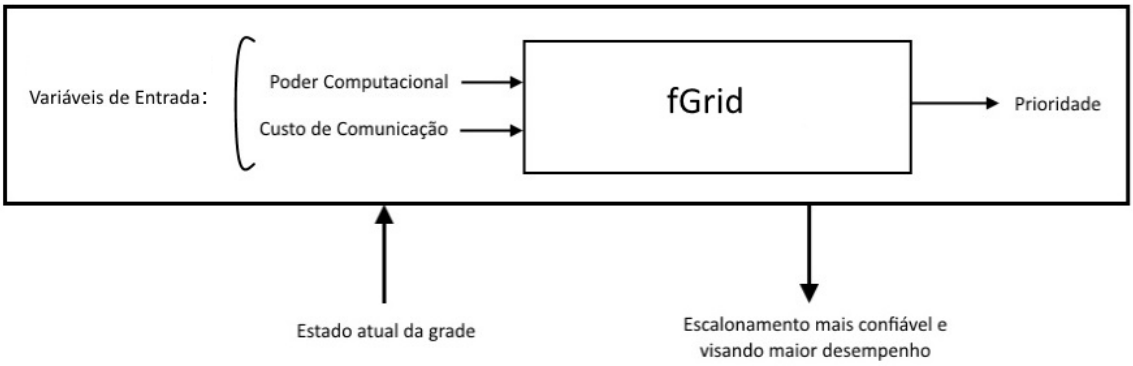


Figura 28: Visão geral das entradas e saídas do *fGrid*.

Por meio deste tratamento utilizando a LF, foi possível dar maior robustez às informações obtidas, tornando a saída do escalonador mais confiável contribuindo para o aumento do desempenho do ambiente da grade. Outros métodos também foram considerados, porém, grande parte deles utilizam migração e reescalonamento de tarefas, e um dos objetivos deste projeto é evitar este tipo de ações pelo fato de inserir uma sobrecarga no recurso de comunicação da grade, o que aumentaria a incerteza sobre informações destes recursos.

Os processos executados pelo *fGrid*, expostos mo fluxograma da Figura 29, representam a forma como ocorre a seleção dos recursos, na seguinte sequência: (i) na primeira etapa é realizada a coleta de informações sobre os recursos, (ii) depois estas informações são tratadas com uso da LF, que por fim (iii) retorna a prioridade dos nodos disponíveis na grade.

Posteriormente, cabe ao escalonador da grade realizar o mapeamento entre as tarefas a serem executadas e os recursos computacionais disponíveis, de acordo com as prioridades apresentadas pelo *fGrid*, ou seja, qual tarefa irá para qual máquina escolhendo sempre da maior prioridade para a menor.

Como os ambientes das grades computacionais possuem uma série de recursos a serem mensurados, este projeto focou em dois recursos considerados os principais nesse contexto, destacados através da Figura 28, que são eles a serem analisados no *fGrid*, as variáveis: PC e CC.

O *fGrid* é responsável pelo escalonamento de aplicações constituídas por tarefas homogêneas do tipo BoT e, para isso, considera-se um sistema fuzzy com uma base de regras que atua em três etapas: (i) Fuzzificação, (ii) Inferência e (iii) Defuzzificação, retornando como saída a Prioridade (P) de cada máquina para recebimento das tarefas.

Através do estudo das variáveis junto a um especialista, foram transformadas as

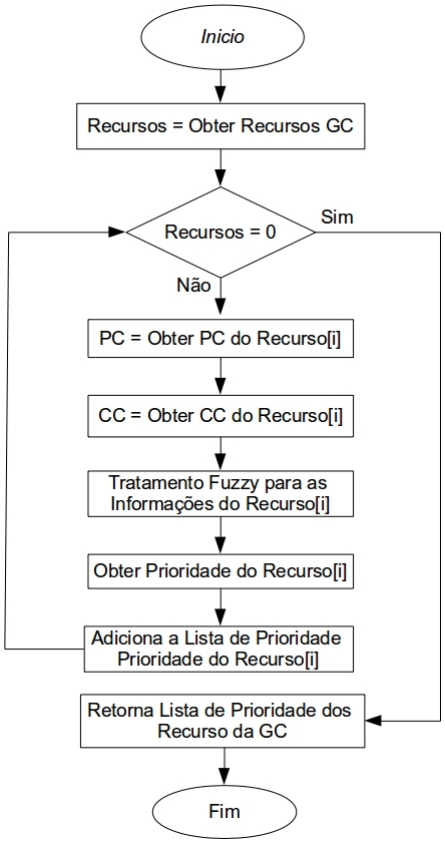


Figura 29: Visão geral das funcionalidades do *fGrid*.

VLs relativas a cada uma das variáveis em CFs, usando a representação gráfica trapezoidal para suas funções de pertinência.

Para mensurar o PC e o CC é realizada uma leitura das configurações do ambiente de grade computacional simulado. Esses valores são então ajustados a uma escala padrão adotada, conforme apresenta a Figura 30 para PC, e para CC como mostrado na Figura 31. Os Termos Linguísticos (TLs) definidos para os CFs da variável PC são: "Limitado" (PCL), "Razoável" (PCR) e "Elevado" (PCE - melhor caso). Sendo $PC = a$ e $a \in [0; 10]$, têm-se as Funções de Pertinência da Tabela 6.

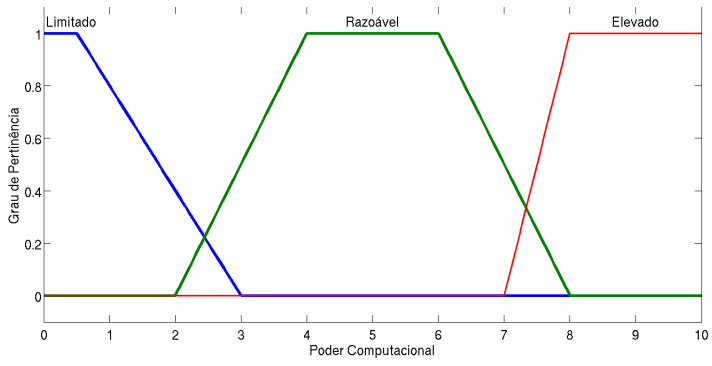


Figura 30: PC na escala padrão

PC	Equação
PCL	$\begin{cases} 1, \text{ se } 0 < a \leq 0,5 \\ \frac{-a+3}{2,5}, \text{ se } 0,5 < a \leq 3 \\ 0, \text{caso contrário.} \end{cases}$
PCR	$\begin{cases} \frac{a-2}{2}, \text{ se } 2 \leq a < 4 \\ 1, \text{ se } 4 \leq a < 6 \\ \frac{-a+8}{2}, \text{ se } 6 \leq a < 8 \\ 0, \text{caso contrário.} \end{cases}$
PCE	$\begin{cases} a - 7, \text{ se } 7 \leq a < 8 \\ 1, \text{ se } 8 \leq a < 10 \\ 0, \text{caso contrário.} \end{cases}$

Tabela 6: PC - Funções de Pertinência

A comunicação é mensurada entre a máquina que contém a aplicação a ser enviada à grade e uma máquina de cada *cluster* contido na grade, sendo retornado o valor, que é então adaptado para a escala padrão, conforme a Figura 31. Os TLs para os CFs definidos para a variável CC são: “Pequeno” (CCP - melhor caso), “Médio” (CCM) e “Grande” (CCG). Sendo $CC = b$ e $b \in [0; 10]$, têm-se as Funções de Pertinência da Tabela 7. É considerado que o custo de comunicação dos vários processadores de um mesmo *cluster* são iguais.

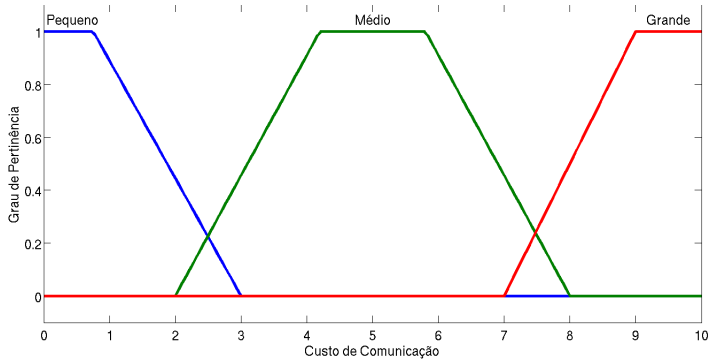


Figura 31: CC na escala padrão

CC	Equação
CCP	$\begin{cases} 1, \text{ se } 0 \leq \text{b} < 0,75 \\ \frac{-\text{b}+3}{2,25}, \text{ se } 0,75 \leq \text{ b } \leq 3 \\ 0, \text{caso contrário.} \end{cases}$
CCM	$\begin{cases} \frac{\text{b-2}}{2,2}, \text{ se } 2 \leq \text{ b } \leq 4,2 \\ 1, \text{ se } 4,2 \leq \text{b} < 5,8 \\ \frac{-\text{b}+8}{1,6}, \text{ se } 5,8 \leq \text{ b } < 8 \\ 0, \text{caso contrário.} \end{cases}$
CCG	$\begin{cases} \frac{\text{b}-7}{2}, \text{ se } 7 \leq \text{ b } < 9 \\ 1, \text{ se } 9 \leq \text{b} < 10 \\ 0, \text{caso contrário.} \end{cases}$

Tabela 7: CC - Funções de Pertinência

A saída P das máquinas também é adaptada para uma escala padrão, conforme exposto na Figura 32. Os TLs para os CFs usados nesse caso são: “Baixa” (PB), “Média” (PM) e “Alta” (PA - melhor caso). Sendo $P = c$ e $c \in [0; 10]$, têm-se as Funções de Pertinência da Tabela 8.

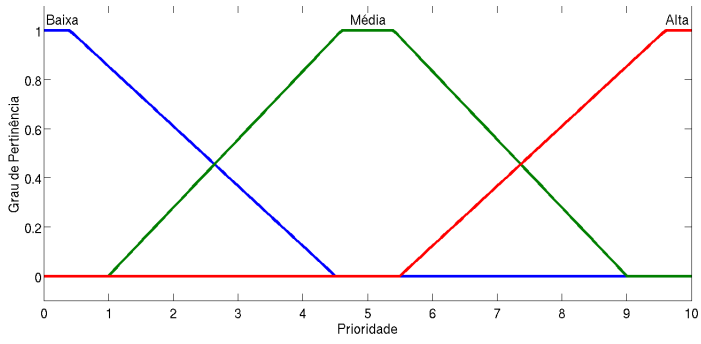


Figura 32: P - escala padrão

P	Equação
PB	$\begin{cases} 1, \text{ se } 0 \leq \text{c} < 0,4 \\ \frac{-\text{c+4,5}}{4,1}, \text{ se } 0,4 \leq \text{ c } < 4,5 \\ 0, \text{caso contrário.} \end{cases}$
PM	$\begin{cases} \frac{\text{c}-1}{3,6}, \text{ se } 1 \leq \text{ c } < 4,6 \\ 1, \text{ se } 4,6 \leq \text{c} < 5,4 \\ \frac{-\text{c}+9}{3,6}, \text{ se } 5,4 \leq \text{ c } < 9 \\ 0, \text{caso contrário.} \end{cases}$
PA	$\begin{cases} \frac{\text{c}-5,5}{4,1}, \text{ se } 5,5 \leq \text{ c } < 9,6 \\ 1, \text{ se } 9,6 \leq \text{c} < 10 \\ 0, \text{caso contrário.} \end{cases}$

Tabela 8: P - Funções de Pertinência

4.1.3 Fuzzificação

Nessa etapa, ocorre o mapeamento dos valores de entrada (já ajustados para escala observada na Seção 4.1.1) para o domínio fuzzy, como pode ser observado Figura 33.

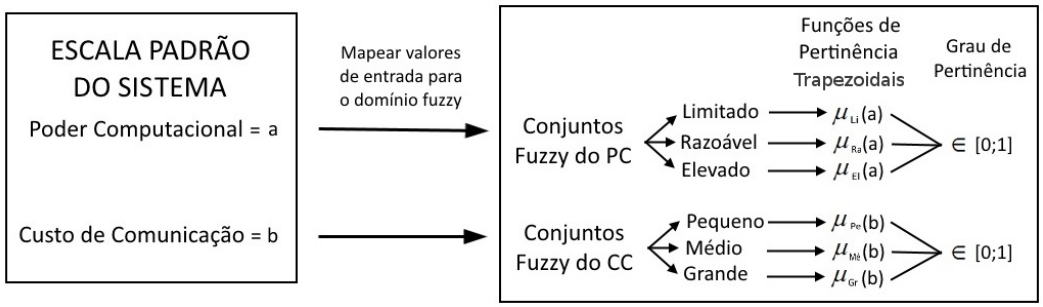


Figura 33: Processo de Fuzzificação do *fGrid*

4.1.4 Base de Regras

A Base de Regras (BR) do *fGrid* é desenvolvida com o intuito de ser facilmente compreensível e editável, visto que não há dificuldade em adicionar novas regras, caso seja desejado manipular outras variáveis de entrada. A BR, observada na Figura 34, leva em conta que o bom desempenho do sistema fuzzy está condicionado às regras que descrevem a estratégia de controle de forma consistente (KLIR, 2005). Leva-se em conta três fatores para sua construção:

- As VLs nomeiam os CFs, tornando a modelagem do sistema mais próxima do mundo real;
- São utilizadas conexões lógicas do tipo “AND” para criar a relação entre as variáveis de entrada, pela simplicidade existente entre as relações, não havendo necessidade de utilizar conexões lógicas do tipo “OR”.
- As implicações são do tipo $modus\ ponens$ (modo afirmativo):

Se ((x é A) e (y é B)) então z é C.

```
RULE 1 : IF poder_computacional   IS limitado AND custo_comunicacao IS pequeno THEN prioridade IS media ;
RULE 2 : IF poder_computacional   IS limitado AND custo_comunicacao IS medio   THEN prioridade IS baixa ;
RULE 3 : IF poder_computacional   IS limitado AND custo_comunicacao IS grande  THEN prioridade IS baixa ;
RULE 4 : IF poder_computacional   IS razoavel AND custo_comunicacao IS pequeno THEN prioridade IS alta  ;
RULE 5 : IF poder_computacional   IS razoavel AND custo_comunicacao IS medio   THEN prioridade IS media ;
RULE 6 : IF poder_computacional   IS razoavel AND custo_comunicacao IS grande  THEN prioridade IS baixa ;
RULE 7 : IF poder_computacional   IS elevado  AND custo_comunicacao IS pequeno THEN prioridade IS alta  ;
RULE 8 : IF poder_computacional   IS elevado  AND custo_comunicacao IS medio   THEN prioridade IS alta  ;
RULE 9 : IF poder_computacional   IS elevado  AND custo_comunicacao IS grande  THEN prioridade IS media ;
```

Figura 34: Base de Regras do *fGrid*

4.1.5 Inferência

No processo de Inferência, ocorrem as operações entre os CFs, combinação dos antecedentes das regras e implicações utilizando o operador *modus ponens generalizado*. Conforme a Figura 35, o processo ocorre em três etapas:

(i) Aplicação da Operação Fuzzy: nesta etapa, ocorre a aplicação dos operadores fuzzy sendo que a entrada consta de dois valores resultantes da fuzzificação. Como as regras são formadas pelo operador fuzzy “AND”, a aplicação utiliza o método MIN (mínimo) sobre os dois valores retornados da fuzzificação;

(ii) Aplicação do Método de Implicação Fuzzy: nesta etapa, é realizada uma combinação entre o valor obtido na aplicação do operador fuzzy e os valores do CF de saída da regra, utilizando o método MIN (mínimo) sobre estas combinações;

(iii) Aplicação do Método de Agregação Fuzzy: nesta etapa, ocorre a composição dos resultados fuzzy de saída de cada regra, utilizando o método MAX (máximo), assim criando uma única região fuzzy para ser analisada pelo próximo processo do módulo fuzzy.

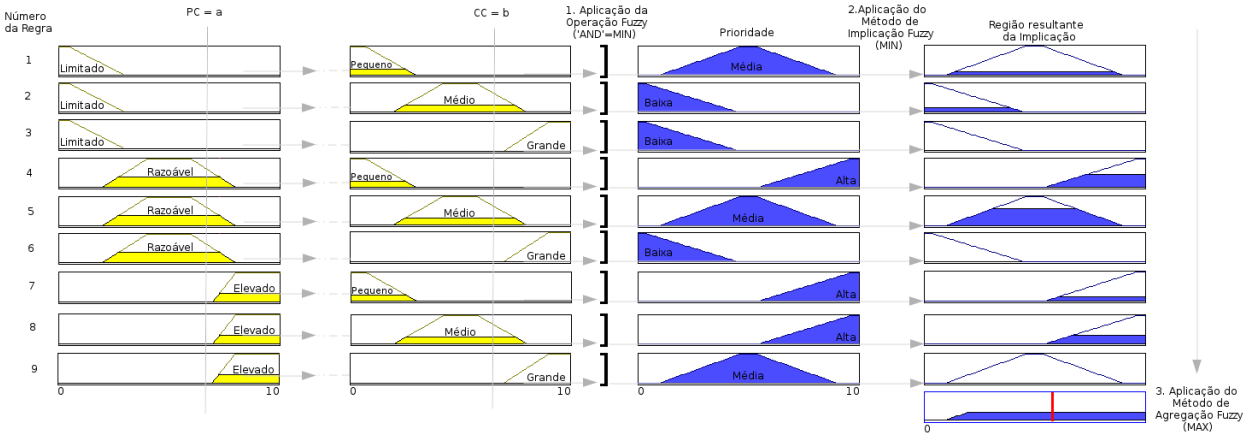


Figura 35: Processo de Inferência do *fGrid*

4.1.6 Defuzzificação

Nessa etapa, ocorre a transformação da região resultado da Inferência em um valor discreto (que representa a Prioridade). Essa transformação foi feita no *fGrid* através do emprego do método centro da área.

Esse método calcula o centroide (x) da área composta pela saída fuzzy do sistema de Inferência (união de todas as contribuições de regras demonstradas nas seções 4.1.4 e 4.1.5), como observado na Figura 36. O centroide é calculado pela seguinte fórmula:

$$u = \frac{\sum_{i=1}^{N} u_i \mu_{OUT}(u_i)}{\sum_{i=1}^{N} \mu_{OUT}(u_i)} \tag{27}$$

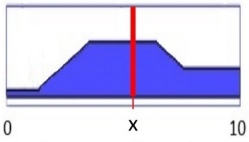


Figura 36: Processo de Defuzzificação utilizando o método Centro da Área

4.1.7 Interpretação Geométrica para o Modelo Fuzzy Tipo 1

A seguir, é ilustrada graficamente a relação da saída Prioridade, a superfície obtida para as variáveis de entrada PC e CC mantém uma relação com a saída Prioridade. Isso deve-se aos intervalos das funções de pertinência utilizadas para PC e CC, assim como a relação direta com a saída Prioridade, que tende a aumentar com o incremento da variável PC e o decremento da entrada CC.

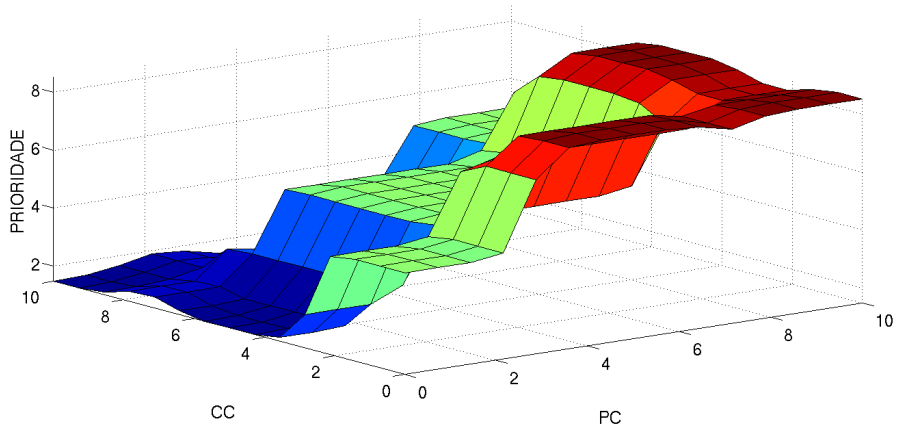


Figura 37: Superfície para PC e CC Modelo Fuzzy Tipo 1

4.2 Tratamento de Incertezas no ETGC via Lógica Fuzzy Tipo 2

A lógica fuzzy do tipo 2 tem sido uma área muito pesquisada nos últimos anos. Este crescimento vem acompanhado de uma potencialidade desta estratégia no tratamento de incertezas em modelos e/ou informações provenientes de especialistas. Podem-se encontrar trabalhos nas áreas de Engenharia, Ciência da Computação, Medicina, Biologia, Economia, Matemática, dentre outras, como pode ser visto em (VAHDANI et al., 2012; KUO; LIANG, 2012; NAJAFI et al., 2016; CHAVAN; SAMBARE; JOSHI, 2016; NAYAK; VATHASAVAI, 2017; CHUANG et al., 2017; HERMAN; PRASAD; MCGINNITY, 2017), evidenciando diversidade e amplitude de aplicação desta metodologia, bem como a eficácia desta extensão em relação à lógica fuzzy do tipo 1 (LIMA; FONTES; SCHNITMAN, 2007).

A lógica fuzzy do tipo 2 trata as incertezas associadas aos conjuntos fuzzy, estendendo a lógica fuzzy do tipo 1, viabilizando portanto, a manipulação de termos imprecisos em toda sua extensão, inclusive na definição das funções de pertinência (MENDEL, 2003).

Além do mais, no ETGC a estimativa da largura de banda disponível nos enlaces das grades exerce um papel fundamental na qualidade do escalonamento, e diferente de informações estáticas como a capacidade máxima dos enlaces, o CC é uma métrica dinâmica que precisa ser frequentemente mensurada. Dentro de um intervalo de tempo, o valor que representa a largura de banda é incerto dada a dinâmica do ambiente e as imprecisões das variáveis que compõem o CC (latência e largura de banda), assim como as imprecisões das ferramentas utilizadas na captura destes valores. Ignorar tais incertezas pode motivar o aumento do *makespan* no escalonamento. Portanto, encontrar um intervalo de valores que represente a largura de banda disponível nos enlaces em certos instantes de tempo mostra-se oportuno para uso diretamente nos escalonadores. Com isso, justifica-se o emprego da LF tipo 2 nesta dissertação.

O *Int-fGrid* é responsável pelo escalonamento de aplicações compostas por tarefas homogêneas do tipo BoT. Para o *Int-fGrid*, de mesmo modo que no *fGrid*, considera-se um sistema fuzzy com uma base de regras que atua em três etapas: (i) Fuzzificação, (ii) Inferência e (iii) Defuzzificação, retornando como saída a Prioridade de cada máquina para recebimento das tarefas. A modelagem do sistema fuzzy do tipo 2 foi realizada com emprego do módulo *Interval Type-2 Fuzzy Logic System Toolbox* (IT2FLT) para o Matlab (CASTRO; CASTILLO; MARTÍNEZ, 2007), no diagrama de bloco da Figura 38, tem-se a apresentação da estrutura do sistema modelado.

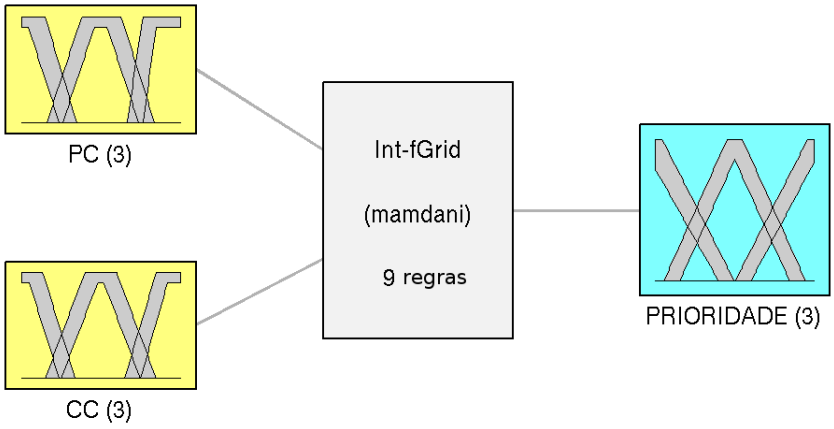


Figura 38: Diagrama de Bloco Sistema Fuzzy Tipo 2

Uma característica importante das aplicações em ambientes distribuídos está relacionada às dependências na comunicação dentre as tarefas. Em aplicações do tipo BoT (TERZOPOULOS; KARATZA, 2016), cada tarefa é independente das demais e a comunicação ocorre somente ao final da sua execução com o retorno dos resultados. Portanto, a ordem na qual as tarefas são executadas não afeta o resultado final do sistema.

4.2.1 Base de Dados e Definição das Funções de Pertinência

Através do estudo das variáveis junto a um especialista, foram transformadas as VLs relativas a cada uma das variáveis de incerteza em CFs, usando a representação gráfica trapezoidal para suas funções de pertinência, tornando assim a modelagem simplificada através do emprego do IT2FLT, ao mesmo tempo que também refletiu a realidade em relação à abordagem aplicada anteriormente e aos valores aplicados no *framework* de simulação.

Para mensurar o PC e o CC, de mesmo modo que empregado no *fGrid*, é realizada a leitura das configurações do ambiente de grade computacional simulado com o SimGrid.

Os TLs que definem os Conjuntos Fuzzy Tipo 2 da variável PC são os seguintes: “Limitado” (PCL), “Razoável” (PCR) e “Elevado” e considerando (PCE - como melhor caso). Sendo $PC = a$ e $a \in [0; 10]$, obtemos as Funções de Pertinência apresentadas na Tabela 9, e em modo gráfico as mesmas são expostas na Figura 39.

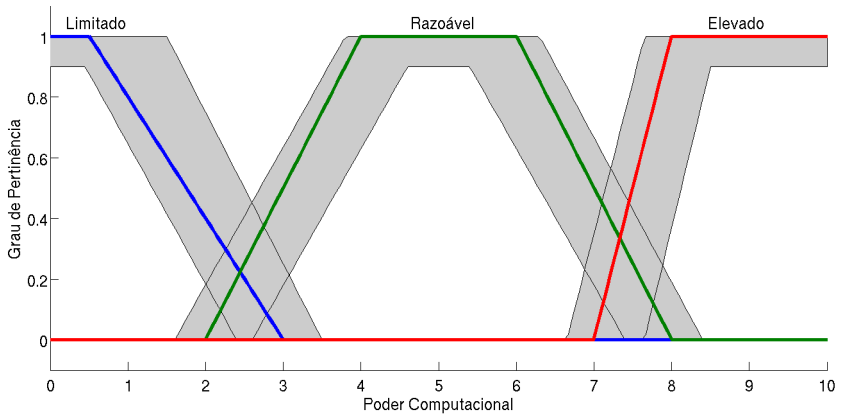


Figura 39: PC na escala padrão

Tabela 9: Funções de Pertinência PC

PC	$\overline{\mu_A(x)}$	$\underline{\mu_{A(x)}}$
PCL	$\begin{cases} 1, \text{ se } 0 \leq x < 1.5 \\ \frac{-x+3.5}{2}, \text{ se } 1.5 \leq x < 3.5 \\ 0, \text{caso contrário.} \end{cases}$	$\begin{cases} 0.9, \text{ se } 0 \leq x < 0.45 \\ \frac{-0.9x+2.4}{2.4}, \text{ se } 0.45 \leq x < 2.4 \\ 0, \text{caso contrário.} \end{cases}$
PCR	$\begin{cases} \frac{x-1.6}{2.2}, \text{ se } 1.6 \leq x < 3.8, \\ 1, \text{ se } 3.8 \leq x < 6.3, \\ \frac{-x+8.4}{2.1}, \text{ se } 6.3 \leq x < 8.4, \\ 0, \text{caso contrário.} \end{cases}$	$\begin{cases} 0.45x - 1,17, \text{ se } 2.6 \leq x < 4.6, \\ 0.9, \text{ se } 4.6 \leq x < 5.4, \\ -0.45x + 3.33, \text{ se } 5.4 \leq x < 7.4, \\ 0, \text{caso contrário.} \end{cases}$
PCE	$\begin{cases} x - 6.65, \text{ se } 6.65 \leq x < 7.65, \\ 1, \text{ se } 7.65 \leq x < 10, \\ 0, \text{caso contrário.} \end{cases}$	$\begin{cases} \frac{18x}{17} - 8.1, \text{ se } 7.65 \leq x < 8.5, \\ 0.9, \text{ se } 8.5 \leq x < 10, \\ 0, \text{caso contrário.} \end{cases}$

A comunicação também é medida entre a máquina que contém a aplicação a ser enviada à grade e uma máquina de cada *cluster* contido na grade. Os TLs para os CFs definidos para essa variável são: “Pequeno” (CCP - melhor caso), “Médio” (CCM) e “Grande” (CCG). Sendo $CC = b$ e $b \in [0; 10]$, têm-se as Funções de Pertinência da Tabela 10 e, graficamente na Figura 40. Também é considerado que o custo de comunicação dos vários processadores de um mesmo *cluster* são iguais.

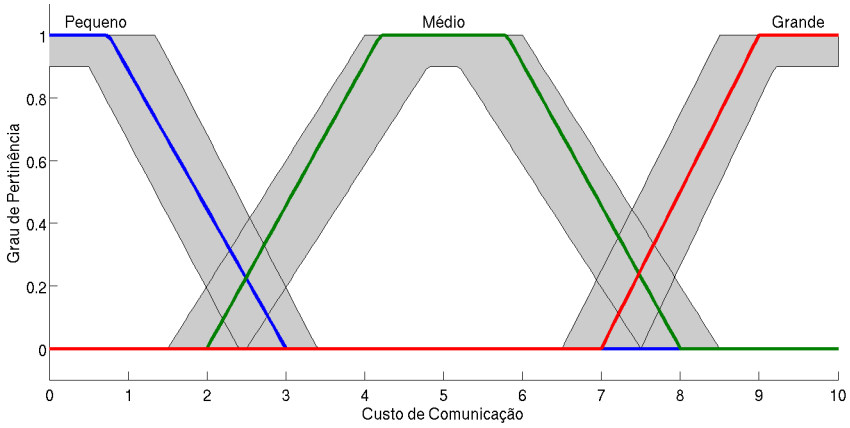


Figura 40: CC na escala padrão

Tabela 10: Funções de Pertinência CC

CC	$\overline{\mu_{B\tilde{X}}}$	$\underline{\mu_{B\tilde{X}}}$
CCP	$\begin{cases} 1, \text{ se } 0 \leq x < 1.35, \\ \frac{-x+3.4}{2.05}, \text{ se } 1.35 \leq x < 3.4, \\ 0, \text{caso contrário.} \end{cases}$	$\begin{cases} 0.9, \text{ se } 0 \leq x < 0.5, \\ \frac{-0.9x+2.16}{1.9}, \text{ se } 0.5 \leq x < 2.4, \\ 0, \text{caso contrário.} \end{cases}$
CCM	$\begin{cases} \frac{x-1.5}{2.5}, \text{ se } 1.5 \leq x < 2.5, \\ 1, \text{ se } 4 \leq x < 6, \\ \frac{-x+8.5}{2.5}, \text{ if } 6 \leq x < 8.5, \\ 0, \text{caso contrário.} \end{cases}$	$\begin{cases} \frac{0.9x-2.25}{2.3}, \text{ se } 2.5 \leq x < 4.8, \\ 0,9, \text{ se } 4.8 \leq x < 5.2, \\ \frac{-0.9x+6.75}{2.3}, \text{ se } 5.2 \leq x < 7.5, \\ 0, \text{caso contrário.} \end{cases}$
CCG	$\begin{cases} \frac{x-6.5}{2}, \text{ se } 6.5 \leq x < 8.5, \\ 1, \text{ se } 8.5 \leq x < 10, \\ 0, \text{caso contrário.} \end{cases}$	$\begin{cases} \frac{18x-13.5}{3.4}, \text{ se } 7.5 \leq x < 9.2, \\ 0.9, \text{ se } 9.2 \leq x < 10, \\ 0, \text{caso contrário.} \end{cases}$

A saída P das máquinas também é adaptada para uma escala padrão, conforme Figura 41. Os TLs para os CFs usados nesse caso são: “Baixa” (PB), “Média” (PM) e “Alta” (PA - melhor caso). Sendo $Pr = c$ e $c \in [0; 10]$, têm-se as Funções de Pertinência da Tabela 11 e, na Figura 41, são apresentadas de forma gráfica as Funções de Pertinência aplicadas a escala padrão.

4.2.2 Fuzzificação

Nesta etapa, ocorre o mapeamento dos valores de entrada (já ajustados para escala observada na Seção 4.1) para o domínio fuzzy, como foi mostrado através da Figura 33.

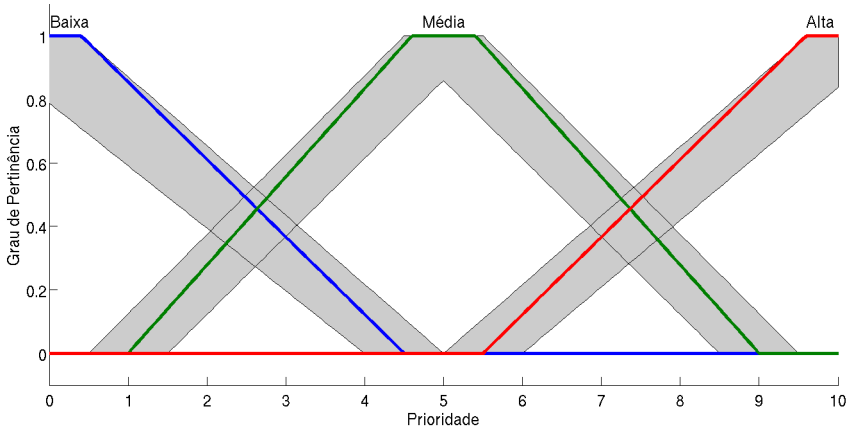


Figura 41: Prioridade na escala padrão

Tabela 11: Funções de Pertinência P

P	$\overline{\mu_C}_{\tilde{X}}$	$\underline{\mu_C}_{\tilde{X}}$
PB	$\begin{cases} \frac{x}{1{,}16}, \text{ se } 0 \leq x < 1.16, \\ 1, \text{ se } 1.16 \leq x < 2.16, \\ \frac{-x+3.5}{1.34}, 2.16 \leq x < 3.5. \\ 0, \text{caso contrário.} \end{cases}$	$\begin{cases} \frac{0.9x-0.45}{1.333}, \text{ se } 0.5 \leq x < 1.66, \\ \frac{-0.9x+2.52}{1.3}, \text{ se } 1.66 \leq x < 2.8, \\ 0, \text{caso contrário.} \end{cases}$
PM	$\begin{cases} \frac{-0.9x-3.447}{1.333}, \text{ se } 3.15 \leq x < 4.5, \\ 1, \text{ se } 4.5 \leq x < 5.5, \\ \frac{-x+6.83}{1.327}, \text{ se } 5.5 \leq x < 6.83, \\ 0, \text{caso contrário.} \end{cases}$	$\begin{cases} \frac{0.9x-3.447}{1.33}, \text{ se } 3.83 \leq x < 4.99, \\ \frac{-0.9x+5.544}{1.12}, \text{ se } 4.99 \leq x < 6.16, \\ 0, \text{caso contrário.} \end{cases}$
PA	$\begin{cases} \frac{x-6.5}{1.33}, \text{ se } 6.5 \leq x < 7.83, \\ 1, \text{ se } 7.83 \leq x < 8.83, \\ \frac{-x+10}{1.17}, \text{ se } 8.83 \leq x < 10. \\ 0, \text{caso contrário.} \end{cases}$	$\begin{cases} \frac{0.9x-6.444}{1.34}, \text{ se } 7.16 \leq x < 8.33, \\ \frac{-0.9x+8.55}{1.34}, \text{ se } 8.33 \leq x < 9.5, \\ 0, \text{caso contrário.} \end{cases}$

4.2.3 Base de Regras

A BR do *Int-fGrid* foi desenvolvida com o intuito de ser facilmente compreensível e editável e, levando em consideração os mesmos critérios que no *fGrid*, tendo em vista que não há dificuldade em adicionar novas regras, caso seja desejado agregar ao modelo outras variáveis de entrada.

4.2.4 Inferência

No processo de Inferência, ocorrem as operações entre os CFs, combinação dos antecedentes das regras e implicações utilizando o operador *modus ponens generalizado*. Este processo ocorre em três etapas detalhadas a seguir. A Figura 42, expõe a modelagem deste processo com emprego do Matlab no módulo IT2FLT.

(i) Aplicação da Operação Fuzzy: nesta etapa, ocorre a aplicação dos operadores fuzzy sendo que a entrada consta de dois valores resultantes da fuzzificação.

Como as regras são formadas pelo operador fuzzy “AND”, a aplicação utiliza o método MIN (mínimo) sobre os dois valores retornados da fuzzificação;

(ii) Aplicação do Método de Implicação Fuzzy: nesta etapa, é realizada uma combinação entre o valor obtido na aplicação do operador fuzzy e os valores do CF de saída da regra, utilizando o método MIN (mínimo) sobre estas combinações;

(iii) Aplicação do Método de Agregação Fuzzy: nesta etapa, ocorre a composição dos resultados fuzzy de saída de cada regra, utilizando o método MAX (máximo), assim criando uma única região fuzzy para ser analisada pelo próximo processo do módulo fuzzy.

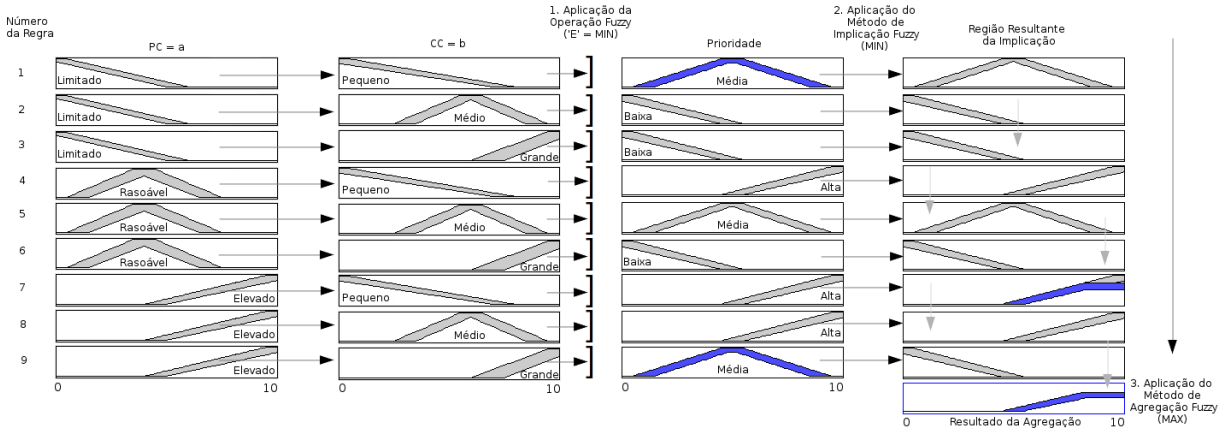


Figura 42: Processo de Inferência do Int-fGrid

4.2.5 Defuzzificação

Nessa etapa, ocorre a transformação da região resultando da Inferência em um valor (que representa a Prioridade). Para que essa transformação fosse realizada ao encontro do *fGrid* foi aplicada a técnica do centro da área para o sistema *Int-fGrid*. Esse método calcula o centroide (x) da área composta pela saída fuzzy do sistema de Inferência (união de todas as contribuições de regras demonstradas nas seções 4.1.4 e 4.1.5). O centroide é calculado através da fórmula da Equação (28).

$$u = \frac{\sum_{i=1}^{N} u_i \mu_{OUT}(u_i)}{\sum_{i=1}^{N} \mu_{OUT}(u_i)} \tag{28}$$

4.2.6 Interpretação Geométrica para o Modelo Fuzzy Tipo 2

Assim como na Seção 4.1.7, a superfície é apresentada para ilustrar a relação entre as entradas e a saída do sistema fuzzy do tipo 2 desenvolvido, as variáveis de entrada PC e CC foram combinadas uma com a outra para gerar a superfície, que do

SBRF2 de *Int-fGrid* em relação ao SBRF1 de *fGrid* apresentada na Figura 37 é muito semelhante, isso deve-se aos intervalos considerados na concepção das funções de pertinência do SBRF2. A Figura 43 ilustra a superfície para a combinação PC e CC, retrata bem o efeito na divisão para a entrada PC em três intervalos, a relação entre o decréscimo do seu valor e o aumento do custo de comunicação fica mais evidente para toda a gama de valores.

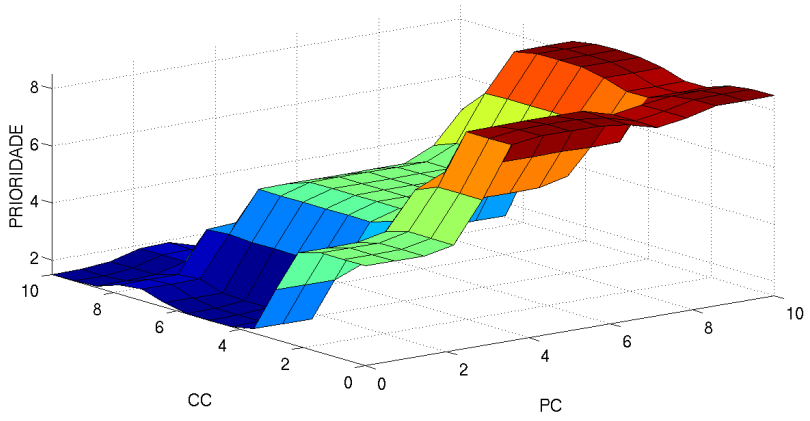


Figura 43: Superfície para PC e CC

4.3 Visão Geral dos Procedimentos Operacionais do FUZZ-GRID

Nesta seção, será apresentada uma visão geral dos procedimentos relacionados à operação do FUZZ-GRID, de modo a representar de forma integrada todas as etapas descritas nas Seções 4.1 e 4.2. Desta forma, facilitando a compreensão do sistema.

O fluxograma apresentado na Figura 44, descreve uma visão geral dos processos envolvidos na operação do *fGrid* e *Int-fGrid*. Os mesmos recebem como entrada uma aplicação constituída por tarefas do tipo BoT e o estado atual dos recursos da grade.

Então, é identificado o estado atual da grade, quais são as máquinas disponíveis para uso, e armazenado esse resultado em uma lista. A seguir, são coletadas as informações de CC e PC de cada máquina disponível adaptando para uma Escala Padrão, onde estarão prontas para ingressarem nos Módulos Fuzzy que, por sua vez, passam por todas as etapas da LF de: (i) Fuzzificação, (ii) Inferência e (iii) Defuzzificação, onde então é obtida a prioridade de cada máquina disponível, armazenando esta em uma Lista de Prioridades e, após, é verificado se ainda há alguma máquina disponível não analisada. Enquanto houver, analisa os dados e armazena na lista de prioridades, ou seja, obtêm a lista de prioridades dos recursos da grade.

Esta lista é enviada ao Tomador de Decisão do Escalonador que, por sua vez irá mapear as tarefas a serem escalonadas nas máquinas disponíveis, de acordo com a lista de prioridades criadas, isto é, quanto maior a prioridade de uma máquina, melhor

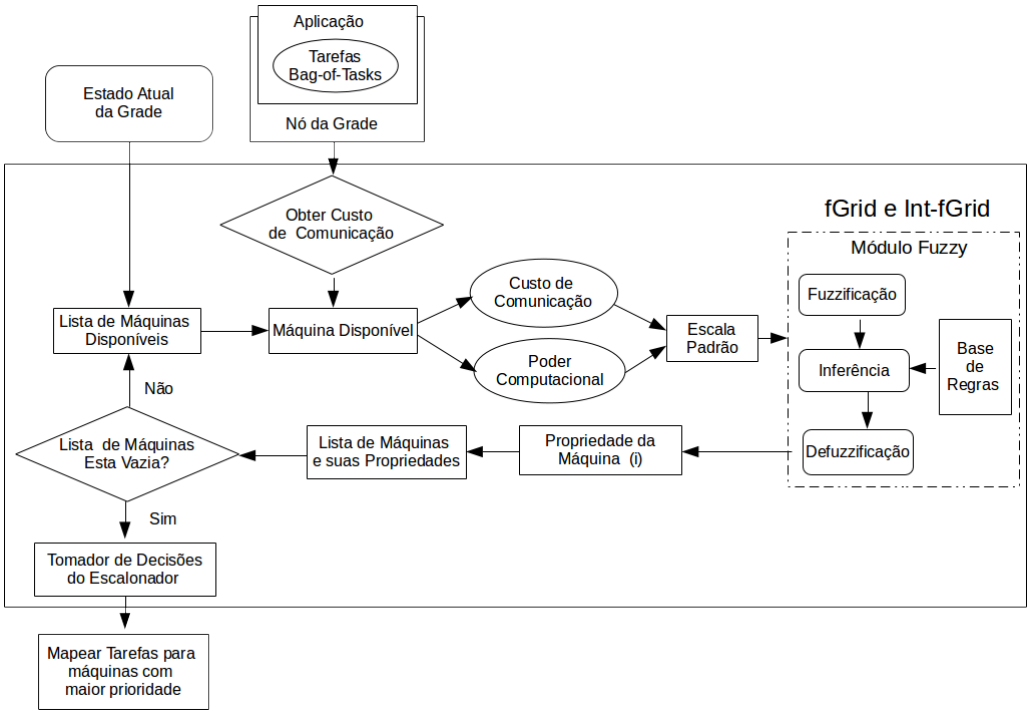


Figura 44: Fluxograma dos processos operacionais do FUZZ-GRID.

as condições de ela receber a tarefa a ser escalonada. Desta forma, as máquinas disponíveis que possuírem as prioridades mais altas serão as primeiras a receberem tarefas.

Visando o menor intervalo de tempo para a execução das tarefas na grade, o escalonador começa mapeando as tarefas para os melhores recursos disponíveis. O tempo de execução da BoT costuma ser utilizado de forma indireta para avaliar a qualidade de um dado escalonamento.

4.4 Análise dos Trabalhos Relacionados

Esta seção apresenta uma breve análise sobre os trabalhos relacionados que foram considerados nesta dissertação.

Na pesquisa de (VAHDAT-NEJAD; MONSEFI, 2008), a técnica de simulação foi utilizada com emprego do Matlab para avaliar o desempenho do algoritmo de escalonamento proposto. A grade simulada consistiu em vários *clusters*, cada um composto no intervalo de 1 a 100 nós. O tempo de intercalação das tarefas foi considerado como satisfazendo um processo com parâmetro de 8 segundos. Cada aplicação foi submetida considerando 3 atributos: (i) hora de chegada, (ii) número de tarefas, e (iii) relação entre comunicação e execução. Cada aplicação submetida foi constituída por um número aleatório de tarefas que variou no intervalo entre 1 a 50 tarefas, e comunicação aleatória entre 1 e 20 Megabytes.

O método foi comparado com um processo denominado Política Distribuída de

Melhor Ajuste, que ignora os requisitos de comunicação das tarefas como largura de banda disponível dos *clusters* e da grade.

Nas execuções, foram considerados dois cenários distintos, são eles: (i) 10 *clusters* e 20 tarefas, (ii) 30 *clusters* e 50 tarefas;

No que trata o tempo de conclusão de processamento, o algoritmo fuzzy mostrou-se melhor em relação a execução com a Política Distribuída de Melhor Ajuste. Como o algoritmo fuzzy emprega os requisitos de comunicação das tarefas e a carga de rede do *cluster* para auxiliar na tomada de decisões quanto ao escalonamento, apresentou resultados mais eficientes e, com isso, melhor *makespan*.

No trabalho de (DANIEL MACEDO BATISTA, 2010), a proposta do escalonador *IP-FULL-FUZZY* apresentou mecanismos que garantem um bom funcionamento das grades independente das incertezas inerentes ao ambiente. Utilizou exemplos numéricos para demonstrar o impacto negativo de se ignorar as incertezas associadas às variáveis dos ambientes das grades. Durante os testes, foram comprovados aumentos de até 75% nos *makespans* de aplicações que foram atribuídas por escalonadores que receberam informações incorretas a cerca da disponibilidade da grade.

Também foram resumidas propostas existentes na literatura para lidar com as incertezas. A maioria dessas propostas tratam delas através de mecanismos que realizam um ciclo contínuo de monitoramento, reescalonamento e migração de tarefas, ou seja, eles agem de forma reativa na tentativa de diminuir o impacto negativo das incertezas.

A motivação para se implementar escalonadores robustos para grades levou à proposta de dois escalonadores de tarefas nesse trabalho. Foram também investigados dois estimadores de largura de banda disponível, a fim de se avaliar a possibilidade de serem utilizados para fornecer estimativas de disponibilidade de largura de banda para um dos escalonadores propostos.

O trabalho destaca primeiramente o *IPDT-FUZZY*, um escalonador robusto às incertezas nas descrições das aplicações. Ele utiliza técnicas de otimização fuzzy e resolve um problema de programação inteira $0-1$ que propõe escalonamentos de aplicações formadas por tarefas dependentes, a fim de minimizar o *makespan*. Experimentos de simulação realizados com três DAGs executados em várias grades comprovaram a eficácia do escalonador em situações onde há uma expectativa alta de incerteza na descrição dos requisitos de comunicação.

O escalonador mostrou-se robusto em termos de *speedup* produzindo tempo de execução para fornecer o escalonamento e utilização dos recursos de comunicação. Os resultados alcançados pelo *IPDT-FUZZY* foram comparados com os resultados do escalonador *IPDT*, que não implementa técnicas de otimização fuzzy e que serviu de base para sua construção. Através das comparações realizadas, o escalonador IPDT-FUZZY produziu escalonamentos com *speedups* até 29,55% maiores, enquanto o tempo de execução foi até 95,67% menor.

Mesmo quando comparado com o *IPDT*, quando este tinha acesso a informações corretas sobre as demandas de comunicação das aplicações, o escalonador *IPDT-FUZZY* mostrou-se vantajoso. Os *speedups* dos escalonamentos propostos por ele foram tão bons quanto aqueles propostos pelo escalonador *IPDT*, com a vantagem do tempo de execução ter sido até 97,77% inferior.

Os resultados alcançados com a implementação do *IPDT-FUZZY* levaram à proposta do escalonador *IP-FULL-FUZZY*, que visa o tratamento tanto das incertezas nas demandas das aplicações, quanto das incertezas na disponibilidade dos recursos. O *IP-FULL-FUZZY*, assim como o *IPDT-FUZZY*, resolve um problema de programação inteira $0-1$.

Experimentos de simulação realizados com vários DAGs e várias grades comprovaram a eficácia do escalonador em situações onde há uma expectativa alta de incerteza na largura de banda disponível entre os computadores. O escalonador *IP-FULL-FUZZY* mostrou-se robusto em termos de *speedup* produzido e apresentou, em média, um tempo de processamento equivalente àquele do escalonador *IPDT-FUZZY*.

Em situações onde foram simuladas incertezas na disponibilidade dos recursos, o *speedup* do escalonador *IP-FULL-FUZZY* chegou a ser, em média, até 41% maior do que o *IPDT-FUZZY* e até 34% maior do que o *RANDOM*, um escalonador que aloca os recursos de forma pseudo-aleatória e ignora todas as fontes de incerteza. Além disso, observou-se que o tempo de execução do *IP-FULL-FUZZY* pode ser, em média, até 13,25% menor do que aquele do *RANDOM*.

No entanto, foi concluído que é importante definir limites de tempo para a execução do *IP-FULL-FUZZY* para evitar que o tempo necessário para derivar o escalonamento invalide a solução. Os escalonamentos produzidos pelo *IP-FULL-FUZZY* foram também comparados com aqueles devolvidos pelos escalonadores clássicos *Heterogeneous Earliest Finish Time* (HEFT) e *Critical Path On a Processor* (CPOP) e foi observado, respectivamente, ganhos de até e 25% e 17% no *speedup*. Os resultados encontrados comprovam a utilidade de tratar as incertezas diretamente nos escalonadores de tarefas em grades.

Para que o *IP-FULL-FUZZY* possa ser utilizado na prática é necessário que as ferramentas de monitoramento forneçam estimativas sobre a disponibilidade dos recursos da grade como intervalos e, além disso, apresentem bom desempenho.

A avaliação realizada em (SANTIAGO et al., 2012), foi com base em conjuntos de aplicações constituídas de tarefas BoT, por serem utilizadas em uma variedade de cenários, incluindo varredura de parâmetros, biologia computacional, imagens, mineração de dados, cálculo de fractal e simulações. Além disso, motivado pela independência entre as tarefas, as aplicações BoT podem ser executadas com mais sucesso em ambientes distribuídos como as grades computacionais.

Neste trabalho, a quantificação das tarefas variou em um intervalo de $[500, 1500]$.

Com o intuito de simplificar o cenário para a análise, o estudo simulou casos com números fixos de GN com 5 nós e 240 recursos.

Os seguintes pressupostos são detalhados neste estudo: (i) as tarefas de uma aplicação são previamente divididas por uma unidade lógica, (ii) todas as tarefas são independentes uma da outra, e (iii) os parâmetros de comunicação entre as tarefas não são necessários.

O modelo de simulação empregado é derivado a partir de cenários de heterogeneidade das tarefas classificadas em dois tipos: (i) alta, variando em $[1, 300000]$, e (ii) baixo, variando em $[1, 10000]$.

Tratando a heterogeneidade de recursos também foi classificada em dois tipos, são eles: (i) alta, variando em $[1, 10000]$, e (ii) baixa, variando em $[1, 1000]$. Por conseguinte, os quatro cenários são apresentados na Tabela 12.

Tabela 12: Quatro cenários no modelo de simulação

Heterogeneidade	Tarefas	Recursos
AA	Alta [1, 300 000]	Alta [1, 10 000]
AB	Alta [1, 300 000]	Baixa [1, 1000]
BA	Baixa [1, 10 000]	Alta [1, 10 000]
BB	Baixa [1, 10 000]	Baixa [1, 1000]

Devido às dificuldades associadas ao uso de ambientes reais, a avaliação foi realizada com emprego de simulação com GridSim (BUYYA; MURSHED, 2002) um *kit* de ferramentas de simulação para a modelagem de recursos e escalonamento de aplicações em sistemas de computação paralela e distribuída.

A abordagem foi comparada com três heurísticas de escalonamento do modo em lote para a chegada das tarefas, que são elas *Max-Min* (LEE; LIANG; CHANG, 2006), *Min-Min* (YU; CHEN, 2008), e *Sufferage* (YU; CHEN, 2008) e uma extensão da *Min-Min*.

No trabalho de (RIBACIONKA, 2013), o algoritmo apresentado permite a utilização de recursos pertencentes a diferentes servidores por uma mesma requisição, e a escolha deles segue um critério que define a melhor combinação de recursos para atender a solicitação e também garante que uma requisição será atendida e em um tempo finito.

O controle efetuado é realizado com a combinação de um relógio lógico e um mapa de disponibilidade, a estratégia adotada oferece um mecanismo de alocação que mostrou ser mais eficiente do que estratégias comuns utilizadas atualmente.

A utilização da LF apresentou resultados eficientes para o auxílio na tomada de decisões para a seleção do servidor de destino de uma requisição, quando mais de

um servidor pode receber a solicitação e, também, quando os servidores estão sobrecarregados com muitas solicitações de recursos.

Através da implementação do algoritmo desenvolvido neste trabalho no ambiente SimGrid foi apresentado que ele é mais eficiente em comparação com o algoritmo *Round Robin* e *Small Latency* quando os servidores ficam sobrecarregados.

Como desvantagem, foi destacado que, para o funcionamento da estratégia, um grande número de mensagens são trocadas pelos servidores e, quando estes não estão sobrecarregados, os algoritmos tradicionais têm ganho de desempenho comparado com o algoritmo desenvolvido neste trabalho.

Nas avaliações realizadas na pesquisa de (KHALILIAN; MIRABEDINI, 2014), a abordagem proposta foi comparada com outros mecanismos, fazendo uso de simulação empregando as ferramentas GridSim e Matlab. No primeiro experimento, o modelo foi avaliado em termos de *makespan* com o *Hierarchical Load Balanced Algorithm* (HLBA) e IHLBA, para este último, os parâmetros do sistema fuzzy são empregados em ordem diferente ao HLBA (LEE; LEU; CHANG, 2011) e, aplicando as configurações apresentadas na Tabela 13, para comparar em termos de *makespan*. A abordagem proposta apresentou melhor desempenho considerando tanto o valor de velocidade de processamento, quanto a carga atual de cada nó da grade.

Tabela 13: Parâmetros de simulação da primeira avaliação

Números de Nós do Clusters	Número de clusters	Número de Tarefas	Poder Computacional dos Nós (MIPS)	Taxa de Transmissão (bps)	Tamanho das Tarefas (MI)
10	10	1000	500-5000	1500	200000-400000

Na segunda avaliação, o modelo foi comparado em termos de *makespan* com outros algoritmos como *Ant Colony Optimization* (ACO) (XU; HOU; SUN, 2003), e o *Most Fit Task First* (MFTF) (WANG; HSU; HUANG, 2005) e o *RANDON*. Na Tabela 14, são apresentados os parâmetros de simulação usados e, como anteriormente, neste caso o método proposto obteve *makespan* menor em relação às outras técnicas.

Tabela 14: Parâmetros de simulação da segunda avaliação

Tamanho da Tarefa (MI)	Taxa de Transmissão	Poder Computacional (MIPS)	Número de Tarefas	Número de Nós por Cluster
300000-500000	1500	2000	10	10

Na Tabela 15, é apresentada uma comparação dos trabalhos relacionados frente a proposta desta dissertação, em relação à (i) taxonomia do escalonamento, (ii) a tipo de ferramentas de simulação utilizadas, (iii) ao número de variáveis de entrada do sistema fuzzy e (iv) aos métodos de fuzzificação empregados em cada trabalho.

Tabela 15: Comparação dos trabalhos relacionados considerando taxonomia, simulação, números de variáveis de entrada do sistema fuzzy, e métodos de fuzzificação usado em cada aplicação.

Trabalho	Taxonomia	Simulação	Variáveis	Fuzzificação
1	♠	MatLab	3	Singlenton
2	♢	GridSim	2	Otimização
3	♢	GridSim	5	Triangular MF
4	♠	SimGrid	2	Triangular MF
5	♠	GridSim/Matlab	3	Triangular MF
fGrid	♢	SimGrid/Matlab	2	Trapezoidal MF
Int-fGrid	♢	SimGrid/Matlab	2	Trapezoidal MF

♠ = Global/Estático/Distribuído
♢ = Global/Estático/Centralizado

De maneira semelhante, na Tabela 16, os trabalhos são comparados em relação ao (i) sistema de inferência, (ii) método de defuzzificação, (iii) conectivos fuzzy, (iv) e tipo de tarefas das aplicações executadas em cada proposta.

Tabela 16: Comparação dos trabalho relacionados considerando sistema de inferência, defuzzificação, conectivos fuzzy, e tarefas executadas em cada aplicação.

Trabalho	Inferência	Defuzificação	Conectivos	Tarefas
1	Product	Average Center	AND	BoT*
2	Optimization	Optimization	Optimization	DAG**
3	Mandani	Centroid	AND	BoT*
4	Mandani	CoG$*$	AND	DAG/BoT**
5	Mandani	Centroid	AND	BoT*
fGrid	Mandani	CoA	AND	BoT*
Int-fGrid	Mandani	Centroid	AND	BoT*

$*$ Center of Gravity (CoG)
$**$ Center of Area (CoA)
Directed Acyclic Graphs (DAG)
Bag of Tasks (BoT)

Já a Tabela 17, expõe uma comparação em relação ao modo de mapeamento das tarefas para realização do escalonamento, e aos métodos usados nos trabalhos relacionados para obtenção de resultados nas avaliações, frente a abordagem de *fGrid* e *Int-fGrid*.

Neste capítulo, foram apresentados os módulos para auxiliar no escalonamento de tarefas em grades computacionais *fGrid* e *Int-fGrid*, que destinam-se ao tratamento

Tabela 17: Comparação dos trabalhos relacionados considerando modo de chegada das tarefas para o escalonamento, e métodos usados na comparação.

Trabalho	Tarefas	Avaliação
1	em lote	Politica Distribuída de Melhor Ajuste
2	em lote	IPDT-FUZZY, IP-FULL-FUZZY, RANDON, HEFT, CPOP
3	em lote	Max-Min, Min-Min, Sufferage, Ext-Min-Min
4	em lote	Round-Robin, Small Latency
5	em lote	HLBA, IHLBA, ACO, MFTF, RANDON
FUZZ-GRID	em lote	RANDON, Round-Robin, XSufferage

HEFT = *Heterogeneous Earliest Finish Time*
CPOP = *Critical Path On a Processor*
HLBA = *Hierarchical Load Balanced Algorithm*
ACO = *Ant Colony Optimization*

das incertezas a cerca das variáveis PC e CC obtidas através da infraestrutura das grades, foi exposto o detalhamento que trata da modelagem e concepção dos mesmos. Também foi abordada uma análise referente aos trabalhos relacionados. No próximo Capítulo 5, são discutidos os assuntos sobre as avaliações realizas empregando os SBRF1 e SBRF2 no contexto do escalonamento de tarefas em grades computacionais com uso do *framework* SimGrid.

5 FUZZ-GRID: TESTES E ANÁLISE DE DESEMPENHO

Este capítulo apresenta as avaliações realizadas durante o andamento da pesquisa. As prioridades dos recursos obtidas através do emprego dos módulos fuzzy *fGrid* com SBRF1, e *Int-fGrid* com SBRF2 foram aplicadas na tomada de decisão quando da realização do escalonamento de aplicações BoT no contexto do ETGC. Para tanto, as avaliações empregaram, nesta dissertação, simulações realizadas com uso das funcionalidades do *framework* SimGrid.

Mais especificamente, na etapa de elaboração das simulações foram utilizados recursos da API MSG do SimGrid versão 3.13, com o objetivo de verificar a eficiência e o *makespan*, comparando a outros algoritmos de uso comum como Randon, Round-Robin, XSufferage.

A principal motivação para seleção dos algoritmos Round-Robin e XSufferage para avaliação de desempenho foram fatos dos mesmos serem algoritmos tradicionais, apesar de não serem orientados para ambientes com incerteza, eles são utilizados, tanto para mostrar o impacto negativo de não considerar incertezas, quanto para servir de referência na avaliação de desempenho, dado que a utilização desses escalonadores como referência de comparação é bastante comum na literatura como pode ser visto em (YU et al., 2007; GHEREGA; PUPEZESCU, 2011; RIBACIONKA, 2013; AGARWAL; JAIN et al., 2014; VIRIYAPANT; SMANCHAT, 2016; GUPTA; KATIYAR, 2017).

O XSufferage (CASANOVA et al., 2000; SANTOS NETO, 2004), uma extensão do Sufferage (YU; CHEN, 2008), é um algoritmo de escalonamento baseado nas informações sobre o desempenho dos recursos e da rede que os interliga. Este algoritmo aborda o impacto das grandes transferências em aplicações que usam grandes quantidades de dados. O XSufferage é uma extensão do algoritmo Sufferage (IBARRA; KIM, 1977; YU; CHEN, 2008), no qual a ideia básica é determinar quanto cada tarefa seria prejudicada se não fosse atribuída ao processador que a executaria de forma mais eficiente. Este valor é denominado sufferage e é definido como a diferença entre o melhor e o segundo melhor tempo de execução previsto para a tarefa, considerando todos os processadores da grade.

A principal diferença entre o Sufferage e o XSufferage é o método usado para calcular o valor de sufferage. O algoritmo XSufferage leva em consideração também a disponibilidade corrente do link de rede e o tempo de transferência dos dados utilizados pela tarefa, diferente do Sufferage que necessita somente das informações relacionadas a CPU e o tempo estimado de execução da tarefa.

Um ponto importante a ser observado é que este algoritmo considera somente os recursos livres no momento em que vai escalonar uma tarefa, pois, caso contrário sempre o recurso mais rápido e com a melhor conexão de rede receberia todas as tarefas.

Além disso, estes algoritmos apresentaram resultados através dos trabalhos relacionados que motivaram a seleção dos mesmos para realização das avaliações nesta dissertação.

O uso do *framework* SimGrid permite simular ambientes que se aproximam do mundo real nas áreas de rede e processamento distribuído e tem sido utilizado por pesquisadores em centros de pesquisa e universidades para prova de conceito; além do mais, possui uma comunidade ativa de usuários e desenvolvedores. Estes fatos foram a motivação para utilização deste *framework* de simulação na etapa de coleta de resultados nesta dissertação.

A repetida utilização dos algoritmos aplicados em um programa desenvolvido no SimGrid teve como objetivo gerar dados que possibilitaram a análise do desempenho da proposta desta dissertação, em comparação com outras soluções existentes.

5.1 fGrid: Modelagem Fuzzy Tipo 1

Esta seção apresenta os resultados obtidos com o emprego do módulo fuzzy *fGrid* com SBRF1 para ETGC.

5.1.1 Estudo de Caso 1

Nesta primeira avaliação do módulo *fGrid* com resultados publicados em (MOURA et al., 2016a), foram desenvolvidas situações de teste considerando a infraestrutura da GridRS. A estrutura da GridRS é composta por *clusters* de quatro universidades do Rio Grande do Sul, que são elas: UFRGS, PUCRS, UFPEL e UFSM.

Cada um desses *clusters* contêm um conjunto de máquinas homogêneas, mas as características de um *cluster* para outro variam, no que diz respeito a poder computacional e canal de comunicação, tornando assim a estrutura como um todo heterogênea. Heterogeneidade essa que motiva a necessidade do tratamento de incertezas quando do escalonamento de tarefas.

Para esta avaliação, são utilizados os valores expostos na Tabela 18 para a configuração dos canais de comunicação considerados nas execuções da Tabela 20.

Tabela 18: Características dos Canais de Comunicação

DE	PARA	Bytes por segundo	Latência segundos
UFPEL	UFPEL	5,25	0,028
UFPEL	UFRGS	2,88	4,943
UFPEL	PUCRS	1,25	4,852
UFPEL	UFSM	0,53	9,493

Na Tabela 19, é listado o PC de cada *cluster* totalizando em 23,77 GFLOPS caso todas as máquinas da grade estiverem disponíveis, os valores são convertidos para a escala padrão utilizada no *fGrid* $(10 * \text{PC}/2000)$, considerando no Estudo de Caso 1 (o valor máximo sendo 2000 MFLOPS).

Tabela 19: PC da GridRS

Instituição	Nº. Máquinas	PC por Máquina (MFLOPS)	PC (Escala Padrão)
UFPEL	10	1889	9,445
UFRGS	10	348	1,74
PUCRS	10	644	3,22
UFSM	10	1397	6,985

Na realização deste experimento, foram consideradas variações de 15 e 30 tarefas homogêneas para 50 Mflop/s, 100 Mflop/s e 200 Mflop/s para demanda computacional das tarefas respectivamente e, em relação ao custo de comunicação e volume de dados a serem transmitidos, foi empregado o valor de 1 MB/s.

Posteriormente, as etapas de Fuzzificação, Inferência e Defuzzificação (explicadas nas Seções 4.1.3, 4.1.5 e 4.1.6) e o emprego do *framework* SimGrid nas execuções das avaliações, os resultados obtidos são apresentados na Tabela 20, onde são expostos os tempos de execução em segundos com a aplicação do *fGrid*, comparado ao método *Randon* de seleção de recursos, e nas Figuras 45 e 46 são mostrados os gráficos referente a estas execuções.

Na primeira execução deste estudo de caso, observa-se que o *fGrid* obteve *makespan* 38,19% melhor em relação ao método *Randon*, logo na terceira execução o *fGrid* conquistou 37,18% menor comparado ao *Randon*, já para a última execução o *fGrid* atingiu até 54,31% menor *makespan*.

5.1.2 Estudo de Caso 2

Para estas execuções com resultados publicados em (MOURA et al., 2016b) e, diferentemente do que foi aplicado anteriormente, as avaliações foram realizadas considerando outra formatação de arquitetura para a GridRS, simultaneamente com outros quantificadores de tarefas. Para a caracterização dos canais de comunicação,

Tabela 20: Avaliação Numérica das Resultados

			Tempo (Segundos)	
Execução	**Nº de Tarefas**	**CC**	**fGrid**	**Randon**
1	15	50	56133	146957
2	15	100	109071	290635
3	15	200	214947	577992
4	30	50	81054	147019
5	30	100	158694	290697
6	30	200	313973	578053

CC = Custo Computacional das tarefas em Mflop/s.

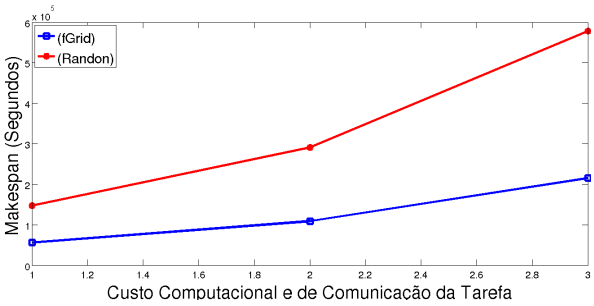


Figura 45: *Makespan* para 15 Tarefas

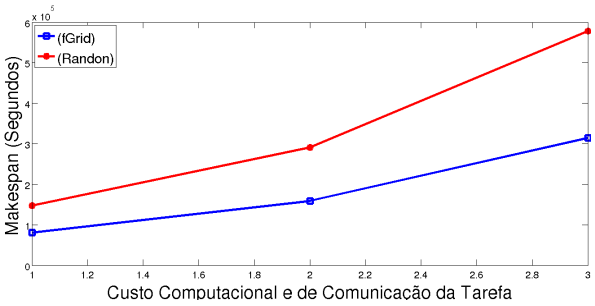


Figura 46: *Makespan* para 30 Tarefas

foram utilizados os valores exibidos na Tabela 21 e empregando flutuação no CC com o SimGrid.

Tabela 21: Características dos Canais de Comunicação

DE	**PARA**	**Bytes por segundo**	**Latência segundos**
UFPEL	UFPEL	10000	0,0011
UFPEL	UFRGS	598	0,089
UFPEL	PUCRS	496	0,023
UFPEL	UFSM	368	4,924

Na Tabela 22, são apresentados o PC de cada *cluster*, ao mesmo tempo que os valores convertidos para a escala padrão utilizada no *fGrid* $(10 * \text{PC}/3000)$, considerando para este caso 3000 MFLOPS como valor máximo.

Neste estágio experimental, foram consideradas aplicações constituídas por conjuntos de 5, 10, 15, 20, 30 e 40 tarefas de custo computacional homogêneo de 60 Mflop/s e custo de comunicação de 5 MB/s. Estas configurações são aplicadas no SimGrid após as etapas de Fuzzificação, Inferência e Defuzzificação (explicadas nas Seções 4.1.3, 4.1.5 e 4.1.6), os resultados obtidos são apresentados na Tabela 23 e através do gráfico da Figura 47.

Nestas execuções junto ao SimGrid, pode ser observado que o *fGrid* obteve

Tabela 22: Descrição da Plataforma da GridRS

Instituição	Nº. Máquinas	PC por Máquina (MFLOPS)	PC (Escala Padrão)
UFPEL	12	2489	8,297
UFRGS	12	2987	9,957
PUCRS	10	1453	4,843
UFSM	10	1266	4,22

Tabela 23: Avaliação Numérica dos Resultados

		Tempo (Segundos)	
Execução	Nº de Tarefas	fGrid	Randon
1	5	21287,8	45594,2
2	10	21829,1	58205,2
3	20	64985,3	75155,9
4	30	83173,1	86300,7
5	40	87520,4	99068,1

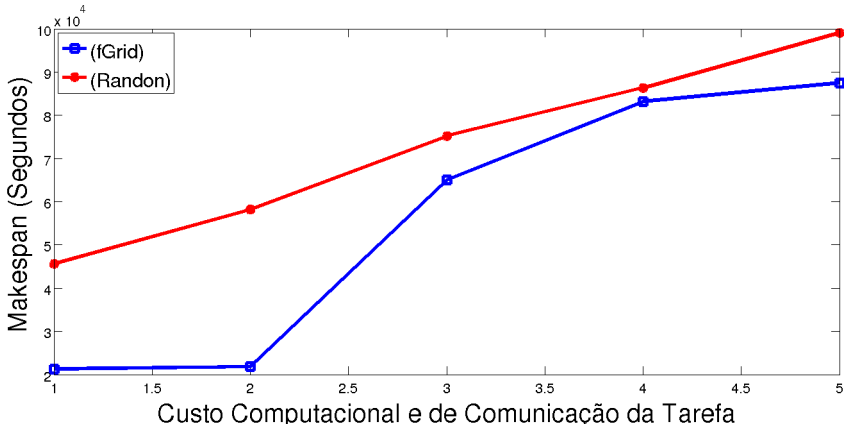


Figura 47: *Makespan* para fGrid, Randon

46,68%, 37,5%, 86,46%, 96,37% e 88,34% melhor eficiência no *makespan* respectivamente, em relação ao método Randon de seleção de recursos.

5.1.3 Estudo de Caso 3

Para a terceira etapa de experimentos com *fGrid*, foram considerados conjuntos de 10, 20, 30, 40 e 50 tarefas homogêneas com o custo computacional de 60 Mflop/s e com custo de comunicação de 5 MB/s, divididas entre os *clusters* apresentados na Tabela 24, com as respectivas características para os CC como é exposto na Tabela 25, considerando que as tarefas são escalonadas a partir do *cluster* da UFPEL.

Distintamente do que foi feito em (MOURA et al., 2016a,b), nesta avaliação foi definida uma variação da carga do volume de tráfego dos canais de comunicação, com intuito de avaliar as execuções simulando a utilização dos mesmos no momento

Tabela 24: Descrição da Plataforma da GridRS

Instituição	CC	Nº de Máquinas	PC (MFLOPS)	P – Lista de Prioridades
UFPEL	Link 1	10	2834	8,35
UFRGS	Link 2	15	1259	7,98
PUCRS	Link 3	15	1698	6,08
UFSM	Link 4	10	1865	5,29

Tabela 25: Características dos Canais de Comunicação

Apelido	DE	PARA	MB/s	Latência segundos
Link 1	UFPEL	UFPEL	1.860	0,000018
Link 2	UFPEL	UFRGS	50	0,000028
Link 3	UFPEL	PUCRS	30	0,000142
Link 4	UFPEL	UFSM	20	0,000219

MB/s = Megabyte por segundo.

da execução do escalonamento e que variou com valores acima de 50% de suas capacidades.

A avaliação é apresentada através da Tabela 26 e no gráfico da Figura 48, onde foi realizada através de comparação da execução com o algoritmo Randon de seleção de recursos computacionais e com emprego da P apresentada na Tabela 24, obtida através da aplicação do módulo *fGrid*.

Tabela 26: Avaliação Numérica dos Resultados

		Tempo (Segundos)	
Execução	Nº de Tarefas	fGrid	Randon
1	10	21,45	7737339,34
2	20	35,16	30293120,68
3	30	19050237,89	54721200,16
4	40	34902443,03	60274861,33
5	50	48804214,9	102571233,95

Para a terceira fase de avaliações empregando o SimGrid o módulo fuzzy *fGrid* nas primeiras duas execuções alcançou *makespans* expressivamente menores de 21,45 e 35,16 segundos em relação ao método Randon, mostrando o impacto negativo gerado por escalonadores que não consideram as incertezas associadas ao PC e ao CC no escalonamento de tarefas em grades computacionais e, após, para as próximas execuções o mesmo atingiu *makespans* de 34,81%, 57,9% e 47,58% em confronto ao método Randon de seleção de recursos.

Esta seção apresentou os resultados obtidos com emprego das prioridades obtidas através do módulo fuzzy fundamentado no SBRF1 denominado *fGrid* para ETGC.

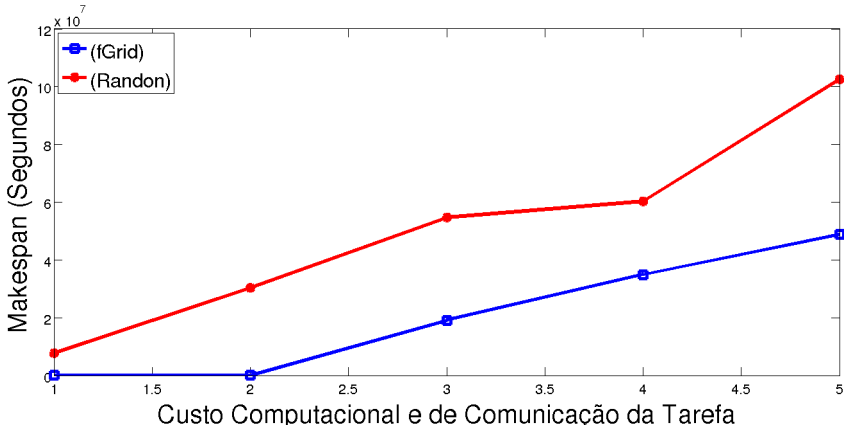


Figura 48: *Makespan* para fGrid, Randon

5.2 Int-fGrid: Modelagem Fuzzy Tipo 2

Esta seção apresenta os resultados obtidos na avaliação do módulo fuzzy para ETGC empregando SBRF2 nomeado *Int-fGrid*.

5.2.1 Estudo de Caso 4

Na Tabela 27, os valores utilizados na configuração da grade são apresentados: (i) quantificadores de máquinas de cada cluster, (ii) PC Mflop/s, (iii) PC convertido na escala padrão $(10 * \text{PC}/3000)$, (iv) apelido dos canais de comunicação utilizados na interconexão da infraestrutura da GridRS simulada.

Tabela 27: Características da GridRS

Instituição	Máquinas	PC (MFLOPS)	PC Padronizado	Canais
UFPEL	10	2073,095	6,91	Link 1
UFRGS	8	693,021	2,31	Link 2
PUCRS	16	1038,052	3,46	Link 3
UFSM	12	1467,067	4,89	Link 4

Para aplicação dos valores referentes às características do Canais de Comunicação, foram usados os valores da Tabela 28, para (i) Largura de Banda, (ii) Latência.

Tabela 28: Características dos Canais de Comunicação

Link	De	Para	Largura de Banda (Mbps)	Latência (ms)
Link 1	UFPEL	UFPEL	92	0,00216
Link 2	UFPEL	UFRGS	60	0,00423
Link 3	UFPEL	PUCRS	41	0,00169
Link 4	UFPEL	UFSM	35	0,00241

Na realização da avaliação, foram consideradas variações para os quantificadores de: (i) tarefas que constituíram as aplicações, (ii) custo computacional das tarefas e (iii) custo de comunicação das tarefas exibidos na Tabela 29.

Visando reproduzir perturbações na entrada do sistema fuzzy, foram aplicadas flutuações nos canais de comunicação da GridRS simulada que variaram no intervalo de 10% a 20% de suas capacidades; de maneira semelhante, a perturbação gerada para o PC foi de 5% considerando a carga de intervenção do *middleware* no gerenciamento da grade.

Tabela 29: Configuração das Tarefas

Execução	Tarefas	PC Tarefas (MFLOPS)	CC das Tarefas (MBps)
1	300	30	10
2	400	10	1
3	500	20	5
4	1000	30	10
5	1500	10	1
6	2000	20	5
7	2500	30	10
8	3000	10	1
9	3500	20	5
10	4000	30	10

As execuções das avaliações são simuladas no SimGrid tendo como base que as tarefas são escalonadas a partir do *cluster* da UFPEL, atribuídas com o uso do algoritmo Round-Robin e XSufferage e, por último, o emprego do módulo *Int-fGrid*, para o qual é obtida uma lista de prioridades através da aplicação das etapas de Fuzzificação, Inferência e Defuzzificação do SBRF2 e, consequentemente, gerada a lista de nodos com suas respectivas prioridades.

De acordo com a maioria das execuções, o método com emprego da abordagem fuzzy tipo 2 obteve melhor *makespan* em relação ao Round-Robin e XSufferage. Os resultados obtidos nas execuções da avaliação são expostos no gráfico da Figura 49, onde os tempos médios das execuções são confrontados.

Os resultados obtidos com o emprego do *Int-fGrid* para auxiliar o escalonador na tomada de decisões estão de acordo com os resultados gerados através do *fGrid*, produzindo como saída valores mais aproximados em referência as configurações aplicadas no *framework* SimGrid.

Entretanto, nas avaliações não estão destacados os tempos de execução para o *fGrid* em relação ao *Int-fGrid*, pois os mesmos apresentaram-se iguais em todas execuções, gerando como saída uma lista de prioridade que é fornecida através da aplicação do *Int-fGrid*, porém com valores diferentes.

Associado a isto, os tempos de execuções obtidos nas simulações são apresentados na Tabela 30 para *Int-fGrid*, Round-Robin e XSufferage, considerando 10% e 20% nas variações da flutuação dos canais de comunicação da rede.

Tabela 30: Avaliação Numérica dos Resultados

Execução	Tempo (Segundos)					
	Int-fGrid		Round-Robin		XSufferage	
	Baixa 10%	Média 20%	Baixa 10%	Média 20%	Baixa 10%	Média 20%
1	22,9597	24,9529	63,1664	69,5522	31,357	34,3172
2	2,29162	2,29162	8,55396	8,91204	4,47636	4,47636
3	11,0866	11,8169	52,9459	58,329	26,5004	28,7369
4	22,9597	24,9529	209,027	233,867	105,508	116,557
5	2,29162	2,29162	31,6976	34,6885	15,9165	17,1501
6	11,0866	11,8169	210,377	232,873	105,336	116,375
7	2,30126	24,9529	522,41	584,789	263,767	292,596
8	2,29162	2,29162	63,4701	69,9228	31,7666	34,8008
9	11,0866	11,8169	369,035	408,698	184,915	204,626
10	22,9597	24,9529	834,774	934,908	421,871	467,807

A Figura 49 descreve a média do *makespan* em segundos para as execuções considerando a configuração dos casos descritos na Tabela 29, aplicando Round-Robin, XSufferage e *Int-fGrid*. Cada coluna mostra a média de todas as execuções para um determinado caso de teste do (eixo x) usando um dos escalonadores.

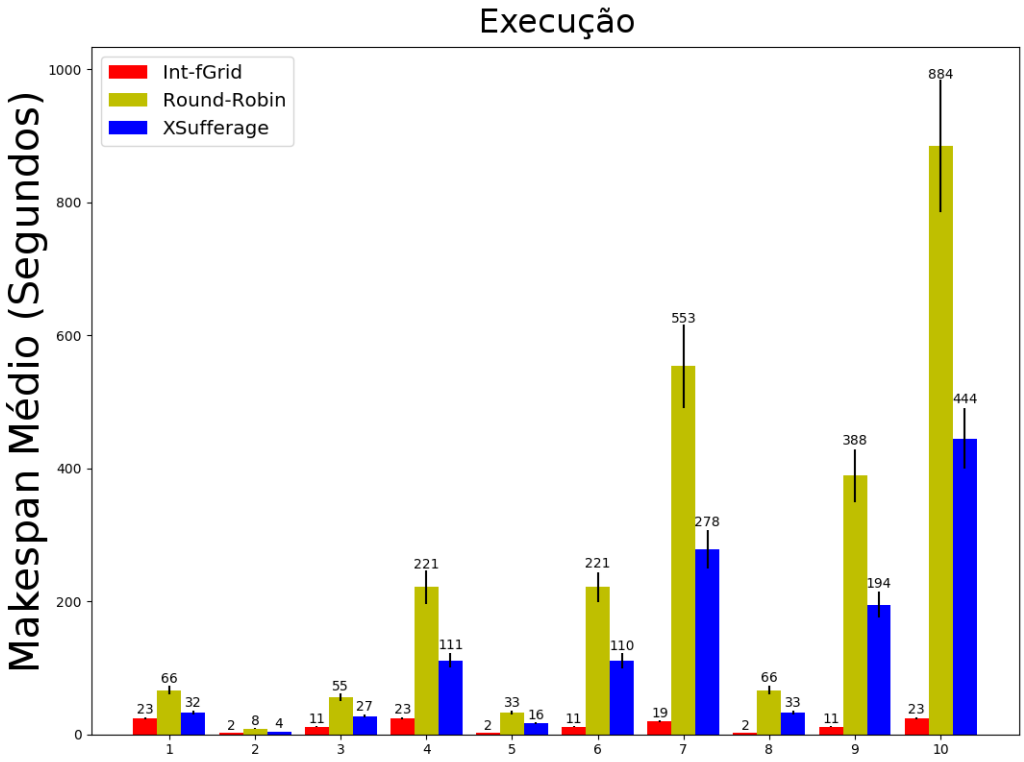


Figura 49: *Makespan* para Int-fGrid, Round-Robin e XSufferage.

A barra de erro mostra a diferença entre os resultados com 10 e 20% das flutuações. Os resultados obtidos na abordagem com SBRF2 de *Int-fGrid* produziu

makespan menor do que os outros dois escalonadores. O módulo *Int-fGrid* fornece um *makespan* 36,9 vezes menor do que a média para Round-Robin e 18,5 vezes menor do que a média para XSufferage para o 12º caso de teste. A variação entre os testes com 10 e 20% foi ligeiramente melhor com *Int-fGrid* com variação de 8,3%, no pior caso, enquanto para Round Robin foi de 11,3% e para XSufferage, 10,3%. Motivando com isso, a continuidade do esforço de estudo e pesquisa na área.

Este capítulo tratou sobre as avaliações realizadas empregando a proposta desta dissertação. No Capítulo 6, são apresentadas as considerações finais, principais conclusões e pretensões de trabalhos futuros.

6 CONSIDERAÇÕES FINAIS

Esta dissertação apresentou a modelagem, implementação e consolidação do módulo fuzzy *fGrid* para o ETGC, que emprega a LF no tratamento das incertezas inerentes às variáveis extraídas da infraestrutura computacional das grades PC e CC. Seu intuito é auxiliar o escalonador na tomada de decisões quanto à realização do escalonamento com objetivo de não executar ações como reescalonamento e migração de tarefas, visto que são ações que oneram os recursos das grades, de mesma forma que os canais de comunicação.

Com objetivo de ampliar o tratamento das incertezas associadas as variáveis PC e CC, também foi proposta a extensão do módulo *fGrid* para o *Int-fGrid*, empregando para este a formalização da LFI (LF tipo 2) e, agregando na abordagem para o tratamento das incertezas, também o tratamento das imprecisões associadas às variáveis PC e CC.

Para a etapa da realização dos testes, foram utilizados ambientes virtuais de grades computacionais com o *framework* SimGrid, tornando rápido o processo de avaliação dos modelos em diferentes configurações de ambientes distribuídos de grades computacionais.

6.1 Principais Conclusões

A heterogeneidade, dinâmica dos recursos e as incertezas nas demandas computacionais e de comunicação das aplicações dificultam a obtenção de informações precisas para o ETGC. Esta dissertação apresentou mecanismos que proporcionam um bom funcionamento das grades independe das incertezas associadas ao PC e ao CC.

Também foram apresentadas algumas propostas existentes na literatura para lidar com as incertezas no contexto do ETGC; em sua maioria, tratam-nas através de mecanismos que realizam um ciclo contínuo de monitoramento, reescalonamento e migração de tarefas, ou seja, agindo de forma reativa na tentativa de minimizar o impacto negativo das incertezas. A carga extra e a intrusão gerada na grade por estas

abordagens justificam a implementação de sistemas que tratem as incertezas diretamente nos escalonadores de tarefas.

A motivação para implementar sistemas robustos as incertezas das variáveis extraídas das grades levou à proposta de dois módulos para auxiliar na tomada de decisões quanto ao escalonamento de tarefas nestes ambientes.

Avaliações com emprego de simulações realizadas em diferentes configurações de aplicações constituídas por tarefas BoT e em várias grades comprovaram a eficácia do tratamento das incertezas associadas ao PC e ao CC. Os escalonamentos obtidos através dos módulos *fGrid* e *Int-fGrid* mostraram-se robustos em termos de *makespans* produzidos, apresentando em média tempos melhores que alcançaram até 36,9 vezes em relação ao Round-Robin, e 18,5 vezes comparado ao XSufferage.

6.2 Publicações Realizadas

Os resultados alcançados através do estudo apresentado nesta dissertação geraram nove publicações. A abordagem empregada na investigação compreende os assuntos referentes aos seguintes aspectos: (i) Lógica Fuzzy, (ii) Sistemas Fuzzy para o Escalonamento de Tarefas em Grades Computacionais, (iii) Grades Computacionais, e (iv) Ambientes de Simulação de Grades Computacionais e Heurísticas de Escalonamento para Ambientes Distribuídos. Em resumo, o andamento da pesquisa destaca que os resultados gerados foram publicados em:

- L. Sampaio, Y. Soares, B. Moura, R. Reiser, A. Yamin and M. Pilla. **fGrid: Modelagem de Variáveis de Incerteza para Escalonamento em Grades Computacionais usando Lógica Fuzzy**. In Workshop-Escola de Informática Teórica (WEIT), p.128-137, Porto Alegre, RS, Outubro 2015.

- B. Moura, R. Reiser and A. Yamin. **Escalonamento de Tarefas em Grades Computacionais sob Regime de Incerteza e Imprecisão de Dados**. In XVI Escola Reginal de Alto Desempenho (ERAD), p.128-135, Porto Alegre, RS, Abril 2016.

- B. Moura, R. Reiser and A. Yamin. **Uma Abordagem Fuzzy Intervalar para Escalonamento de Tarefas em Grades Computacionais**. In XXXVI Congresso Nacional de Matemática Aplicada e Computacional (CNMAC), p.1-2, Gramado, RS, Setembro 2016.

- L. Sampaio, Y. Soares, B. Moura, R. Reiser A. Yamin and M. Pilla. **Escalonamento em Grades Computacionais usando Lógica Fuzzy**. In XXXVI Congresso Nacional de Matemática Aplicada e Computacional (CNMAC), p.1-2, Gramado, RS, Setembro 2016.

- B. Moura, R. Reiser, A. Yamin. **Uma Avaliação do Modelo Fuzzy fGrid para Escalonamento de Tarefas em Grades Computacionais**. In IV Congresso Brasileiro de Sistemas Fuzzy (CBSF), p.92-95, Campinas, SP, Outubro 2016.

- B. Moura, R. Reiser, A. Yamin. **Modelagem e Avaliação para o Escalonamento de Tarefas em Grades Computacionais Empregando a Lógica Fuzzy e o SimGrid**. In Encontro de Pós-Graduação, Universidade Federal de Pelotas (ENPOS), p.1-4, Pelotas, RS, Setembro 2016.

Foram também elaborados 3 artigos que foram submetidos a eventos internacionais, que são eles:

- B. Moura, Y. Soares, L. Sampaio, R. Reiser, A. Yamin and M. Pilla. **fGrid: Uncertainty Variables Modeling for Computational Grids using Fuzzy Logic**. In International Conference on Fuzzy Systems (FUZZ-IEEE 2016), pages 2249-2256, Vancouver,CA, July 2016.

- B. Moura, Y. Soares, L. Sampaio, R. Reiser, A. Yamin, and M. Pilla. **Fuzzy System Modeling for Task Scheduling in Computational Grids**. In International Conference on Uncertainty Modelling in Knowledge Engineering and Decision Making (FLINS 2016), pages 806-811. Word Scientific, 2016.

- Bruno M. P. Moura, Guilherme B. Schneider, Adenauer C. Yamin, Renata H. S. Reiser and Mauricio L. Pilla. **Int-fGrid: a Type-2 Fuzzy Approach for Scheduling Tasks of Computational Grids**. In 13^{th} International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD 2017), pages 1-8, Guilin, China, July 2017.

6.3 Trabalhos Futuros

Na listagem a seguir, são apresentados alguns itens relacionados a esta dissertação que merecem investigação futura:

- Agregar ao modelo o tratamento de aplicações constituídas por tarefas com dependência, usualmente representas por DAGs;

- Para tanto será explorada a API SimDag do SimGrid;

- Empregar ferramentas baseadas em Java para LF tipo 1 e tipo 2 como o projeto Juzzy (WAGNER, 2013; WAGNER; PIERFITT; MCCULLOCH, 2014);

- Explorar a jMSG API Java do SimGrid, unificando o desenvolvimento de software empregando a linguagem Java;

- Comparar a outros métodos de escalonamento da literatura;
- Implementar as soluções em um ambiente real de grade computacional.

REFERÊNCIAS

AGARWAL, D.; JAIN, S. et al. Efficient optimal algorithm of task scheduling in cloud computing environment. **arXiv preprint arXiv:1404.2076**, [S.l.], 2014.

AMADOR-ANGULO, L.; CASTILLO, O.; CASTRO, J. R. A Generalized Type-2 Fuzzy Logic System for the dynamic adaptation the parameters in a Bee Colony Optimization algorithm applied in an autonomous mobile robot control. In: IEEE INTERNATIONAL CONFERENCE ON FUZZY SYSTEMS (FUZZ-IEEE), 2016., 2016. **Anais...** [S.l.: s.n.], 2016. p.537–544.

ATANASSOV, K. T.; PASI, G.; YAGER, R. R.; ATANASSOVA, V. Intuitionistic fuzzy graph interpretations of multi-person multi-criteria decision making. In: EUSFLAT CONF., 2003. **Anais...** [S.l.: s.n.], 2003. p.177–182.

BACZYŃSKI, M. Residual implications revisited. Notes on the Smets–Magrez theorem. **Fuzzy Sets and Systems**, [S.l.], v.145, n.2, p.267–277, 2004.

BACZYŃSKI, M.; JAYARAM, B. **An Introduction to Fuzzy Implications**. [S.l.]: Springer, 2008.

BAGHERI, E.; NAGHIBZADEH, M. A New Approach to Resource Discovery and Dissemination for Pervasive Computing Environments Based on Mobile Agents. **Scientia Iranica**, [S.l.], v.14, n.6, p.612–624, 2007.

BAJAJ, C.; DOGRA, A.; SINGH, G. REVIEW AND ANALYSIS OF TASK SCHEDULING ALGORITHMS. **International Research Journal of Engineering and Technology (IRJET)**, [S.l.], 2015.

BALASUBRAMANIAM, J. Yager?s new class of implications J f and some classical tautologies. **Information Sciences**, [S.l.], v.177, n.3, p.930–946, 2007.

BARROS, L. e Bassanezi, RC “Tópicos de Lógica Fuzzy e Biomatemática”. **Primeira edição. Coleção IMECC Textos Didáticos**, [S.l.], v.5, 2010.

BARROS L. C.; BASSANEZI, R. C. **Tópicos de Lógica Fuzzy com Aplicações em Biomatemática**. [S.l.]: SP: UNICAMP/IMECC, 2010.

BARROS, L. C. de; BASSANEZI, R. C. **Tópicos de lógica fuzzy e biomatemática**. [S.l.]: Unicamp-Imecc, 2006.

BEDREGAL, B. C. On interval fuzzy negations. **Fuzzy Sets and Systems**, [S.l.], v.161, n.17, p.2290–2313, 2010.

BELL, W. H.; CAMERON, D. G.; MILLAR, A. P.; CAPOZZA, L.; STOCKINGER, K.; ZINI, F. Optorsim: A grid simulator for studying dynamic data replication strategies. **International Journal of High Performance Computing Applications**, [S.l.], v.17, n.4, p.403–416, 2003.

BENTKOWSKA, U.; KRÓL, A. Preservation of fuzzy relation properties based on fuzzy conjunctions and disjunctions during aggregation process. **Fuzzy Sets and Systems**, [S.l.], 2015.

BOLZE, R.; CAPPELLO, F.; CARON, E.; DAYDÉ, M.; DESPREZ, F.; JEANNOT, E.; JÉGOU, Y.; LANTERI, S.; LEDUC, J.; MELAB, N. et al. Grid'5000: a large scale and highly reconfigurable experimental grid testbed. **International Journal of High Performance Computing Applications**, [S.l.], v.20, n.4, p.481–494, 2006.

BUSTINCE, H.; BARRENECHEA, E.; MOHEDANO, V. Intuicionistic Fuzzy Implication Operators - An Expression and Main Properties. **Uncertainty, Fuzziness and Knowledge-Based Systems**, [S.l.], v.12, p.387–406, 2004.

BUSTINCE, H.; BURILLO, P.; SORIA, F. Automorphisms, negations and implication operators. **Fuzzy Sets and Systems**, [S.l.], v.134, n.2, p.209–229, 2003.

BUYYA, R.; MURSHED, M. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. **Concurrency and computation: practice and experience**, [S.l.], v.14, n.13-15, p.1175–1220, 2002.

CARLSSON, C. F. R. **Fuzzy Reasoning in Decision Making and Optimization**. [S.l.]: Heidelberg: Physiva-Verlag Springer, 2002.

CASANOVA, H. Simgrid: A toolkit for the simulation of application scheduling. In: CLUSTER COMPUTING AND THE GRID, 2001. PROCEEDINGS. FIRST IEEE/ACM INTERNATIONAL SYMPOSIUM ON, 2001. **Anais...** [S.l.: s.n.], 2001. p.430–437.

CASANOVA, H.; GIERSCH, A.; LEGRAND, A.; QUINSON, M.; SUTER, F. Versatile, Scalable, and Accurate Simulation of Distributed Applications and Platforms. **Journal of Parallel and Distributed Computing**, [S.l.], v.74, n.10, p.2899–2917, June 2014.

CASANOVA, H.; LEGRAND, A.; QUINSON, M. Simgrid: A generic framework for large-scale distributed experiments. In: COMPUTER MODELING AND SIMULATION, 2008. UKSIM 2008. TENTH INTERNATIONAL CONFERENCE ON, 2008. **Anais...** [S.l.: s.n.], 2008. p.126–131.

CASANOVA, H.; LEGRAND, A.; ZAGORODNOV, D.; BERMAN, F. Heuristics for Scheduling Sweep Applications in Grid environments. In: HETEROGENEOUS COMPUTING SYSTEMS WORKSHOP, 9., 2000. **Anais...** [S.l.: s.n.], 2000.

CASANOVA, H.; MARCHAL, L. A network model for simulation of grid application. **Research Report**, [S.l.], 2002.

CASAVANT, T. L.; KUHL, J. G. A taxonomy of scheduling in general-purpose distributed computing systems. **Software Engineering, IEEE Transactions on**, [S.l.], v.14, n.2, p.141–154, 1988.

CASTRO, J. R.; CASTILLO, O.; MARTÍNEZ, L. G. Interval Type-2 Fuzzy Logic Toolbox. **Engineering Letters**, [S.l.], v.15, n.1, p.89–98, 2007.

CHAVAN, S. V.; SAMBARE, S. S.; JOSHI, A. Diet recommendation based on Prakriti and season using Fuzzy ontology and Type-2 Fuzzy Logic. In: INTERNATIONAL CONFERENCE ON COMPUTING COMMUNICATION CONTROL AND AUTOMATION (ICCUBEA), 2016., 2016. **Anais...** [S.l.: s.n.], 2016. p.1–6.

CHUANG, C.-Y.; CHEN, P.-S.; HSU, C.-C.; LI, J.-Y.; CHEN, J.-F.; LIN, C.-L. Novel maximum power point tracker for PV systems using interval type-2 fuzzy logic controller. In: IEEE 3RD INTERNATIONAL FUTURE ENERGY ELECTRONICS CONFERENCE AND ECCE ASIA (IFEEC 2017 - ECCE ASIA), 2017., 2017. **Anais...** [S.l.: s.n.], 2017. p.1505–1507.

CHUN, B.; CULLER, D.; ROSCOE, T.; BAVIER, A.; PETERSON, L.; WAWRZONIAK, M.; BOWMAN, M. Planetlab: an overlay testbed for broad-coverage services. **ACM SIGCOMM Computer Communication Review**, [S.l.], v.33, n.3, p.3–12, 2003.

CIRNE, W. Grids Computacionais: Arquiteturas, Tecnologias e Aplicações (Computational Grids: Architectures, Tecnologies and Applications). In: TERCEIRO WORKSHOP EM SISTEMAS COMPUTACIONAIS DE ALTO DESEMPENHO (PROCEEDINGS OF THE THIRD WORKSHOP ON HIGH PERFOMANCE COMPUTING SYSTEMS), 2002. **Anais...** [S.l.: s.n.], 2002.

CIRNE, W.; BRASILEIRO, F.; PARANHOS, D.; GÓES, L. F. W.; VOORSLUYS, W. On the efficacy, efficiency and emergent behavior of task replication in large distributed systems. **Parallel Computing**, [S.l.], v.33, n.3, p.213–234, 2007.

CLAUSS, P.-N.; STILLWELL, M.; GENAUD, S.; SUTER, F.; CASANOVA, H.; QUINSON, M. Single node on-line simulation of mpi applications with smpi. In: PARALLEL & DISTRIBUTED PROCESSING SYMPOSIUM (IPDPS), 2011 IEEE INTERNATIONAL, 2011. **Anais...** [S.l.: s.n.], 2011. p.664–675.

DA SILVA, D. P.; CIRNE, W.; BRASILEIRO, F. V. Trading cycles for information: Using replication to schedule bag-of-tasks applications on computational grids. In: **Euro-Par 2003 Parallel Processing**. [S.l.]: Springer, 2003. p.169–180.

DANIEL MACEDO BATISTA, N. L. S. d. F. **Escalonadores de Tarefas Dependentes para Grades Robustosas Incertezas das Informaçoes de Entrada**. 2010. Tese (Doutorado em Ciência da Computação) — Instituto de Computaçao.

DE BAETS, B. Coimplicators, the forgotten connectives. **Tatra Mt. Math. Publ.**, [S.l.], v.12, p.229–240, 1997.

DENGFENG, L.; CHUNTIAN, C. New similarity measures of intuitionistic fuzzy sets and application to pattern recognitions. **Pattern Recognition Letters**, [S.l.], v.23, n.1, p.221–225, 2002.

DESCHRIJVER, G.; KERRE, E. Implicators based on binary aggregation operators in interval-valued fuzzy set theory. **Fuzzy Sets and Systems**, [S.l.], v.153, n.2, p.229–248, 2005.

DUBOIS, D.; PRADE, H. **Fundamentals of Fuzzy Sets**. Boston: Kluwer Academic Publishers, 2000.

FODOR, J. C. On fuzzy implication operators. **Fuzzy Sets and Systems**, [S.l.], v.42, n.3, p.293–300, 1991.

FODOR, J. C. Contrapositive symmetry of fuzzy implications. **Fuzzy Sets and Systems**, [S.l.], v.69, n.2, p.141–156, 1995.

FODOR, J. C.; ROUBENS, M. **Fuzzy preference modelling and multicriteria decision support**. [S.l.]: Springer Science & Business Media, 1994. v.14.

FODOR, J.; ROUBENS, M. **Fuzzy Preference Modelling and Multicriteria Decision Support**. Dordrecht: Kluwer Academic Publisher, 1994.

FONT, J. M.; HÁJEK, P. On Łukasiewicz's four-valued modal logic. **Studia Logica**, [S.l.], v.70, n.2, p.157–182, 2002.

FOSTER, I.; ZHAO, Y.; RAICU, I.; LU, S. Cloud Computing and Grid Computing 360-Degree Compared. In: GRID COMPUTING ENVIRONMENTS WORKSHOP, 2008., 2008. **Anais...** [S.l.: s.n.], 2008. p.1–10.

GEORGE J, K.; BO, Y. Fuzzy sets and fuzzy logic, theory and applications. **-**, [S.l.], 2008.

GHANEM, A. M. A.; SALEH, A.; ALI, H. A. et al. High performance adaptive framework for scheduling Grid Workflow applications. In: COMPUTER ENGINEERING AND SYSTEMS (ICCES), 2010 INTERNATIONAL CONFERENCE ON, 2010. **Anais...** [S.l.: s.n.], 2010. p.52–57.

GHEREGA, A.; PUPEZESCU, V. Multi-agent Resource Allocation Algorithm Based on the XSufferage Heuristic for Distributed Systems. In: INTERNATIONAL SYMPOSIUM ON SYMBOLIC AND NUMERIC ALGORITHMS FOR SCIENTIFIC COMPUTING, 2011., 2011. **Anais...** [S.l.: s.n.], 2011. p.313–320.

GULLAPALLI, S. Atlas: An Intelligent, Performant Framework for Web-based Grid Computing. In: ACM SIGSOFT INTERNATIONAL SYMPOSIUM ON FOUNDATIONS OF SOFTWARE ENGINEERING, 2016., 2016, New York, NY, USA. **Proceedings...** ACM, 2016. p.1154–1156. (FSE 2016).

GUPTA, K.; KATIYAR, V. Survey of Resource Provisioning Heuristics in Cloud and their Parameters. **International Journal of Computational Intelligence Research**, [S.l.], v.13, n.5, p.1283–1300, 2017.

HERMAN, P. A.; PRASAD, G.; MCGINNITY, T. M. Designing an Interval Type-2 Fuzzy Logic System for Handling Uncertainty Effects in Brain – Computer Interface Classification of Motor Imagery Induced EEG Patterns. **IEEE Transactions on Fuzzy Systems**, [S.l.], v.25, n.1, p.29–42, Feb 2017.

HERRERA, F.; MARTINEZ, L.; SÁNCHEZ, P. J. Managing non-homogeneous information in group decision making. **European Journal of Operational Research**, [S.l.], v.166, n.1, p.115–132, 2005.

HORČÍK, R.; NAVARA, M. Validation sets in fuzzy logics. **Kybernetika**, [S.l.], v.38, n.3, p.319–326, 2002.

HOWE, D. et al. The free on-line dictionary of computing. **FOLDOC, Retrieved Aug**, [S.l.], v.30, p.2010, 1993.

IBARRA, O. H.; KIM, C. E. Heuristic algorithms for scheduling independent tasks on nonidentical processors. **Journal of the ACM (JACM)**, [S.l.], v.24, n.2, p.280–289, 1977.

JIANG, H.; NI, T. PB-FCFS-a task scheduling algorithm based on FCFS and backfilling strategy for grid computing. In: PERVASIVE COMPUTINGHANEM2010HIGHG

(JCPC), 2009 JOINT CONFERENCES ON, 2009. **Anais...** [S.l.: s.n.], 2009. p.507–510.

KHALILIAN, Z.; MIRABEDINI, S. J. A Novel Decentralized Fuzzy Based Approach for Grid Job. **Journal of Telecommunication, Electronic and Computer Engineering (JTEC)**, [S.l.], v.6, n.1, p.21–26, 2014.

KLEMENT E. MESIAR R., P. E. **Triangular Norms.** [S.l.]: Kluwer Academic Publisher,, 2003.

KLEMENT, E. P.; MESIAR, R.; PAP, E. **Triangular Norms**. Dordrecht: Kluwer Academic Publisher, 2000.

KLIR, G. J. Developments in uncertainty-based information. **Advances in computers**, [S.l.], v.36, p.255–332, 1993.

KLIR, G. J. **Uncertainty and Information**: Foundations of Generalized Information Theory. [S.l.]: Wiley-Interscience Malden, USA, 2005.

KRAUTER, K.; BUYYA, R.; MAHESWARAN, M. A taxonomy and survey of grid resource management systems for distributed computing. **Software-Practice and Experience**, [S.l.], v.32, n.2, p.135–64, 2002.

KUO, M.-S.; LIANG, G.-S. A soft computing method of performance evaluation with MCDM based on interval-valued fuzzy numbers. **Applied Soft Computing**, [S.l.], v.12, n.1, p.476 – 485, 2012.

LAMPORT, L. Time, Clocks, and the Ordering of Events in a Distributed System. **Commun. ACM**, New York, NY, USA, v.21, n.7, p.558–565, July 1978.

LEE, L. t.; LIANG, C. h.; CHANG, H. y. An Adaptive Task Scheduling System for Grid Computing. In: THE SIXTH IEEE INTERNATIONAL CONFERENCE ON COMPUTER AND INFORMATION TECHNOLOGY (CIT'06), 2006. **Anais...** [S.l.: s.n.], 2006. p.57–62.

LEE, Y.-H.; LEU, S.; CHANG, R.-S. Improving job scheduling algorithms in a grid environment. **Future generation computer systems**, [S.l.], v.27, n.8, p.991–998, 2011.

LEGRAND, A.; MARCHAL, L.; CASANOVA, H. Scheduling distributed applications: the simgrid simulation framework. In: CLUSTER COMPUTING AND THE GRID, 2003. PROCEEDINGS. CCGRID 2003. 3RD IEEE/ACM INTERNATIONAL SYMPOSIUM ON, 2003. **Anais...** [S.l.: s.n.], 2003. p.138–145.

LEITE, D.; SANTANA, M.; BORGES, A.; GOMIDE, F. Fuzzy Granular Neural Network for incremental modeling of nonlinear chaotic systems. In: IEEE INTERNATIONAL CONFERENCE ON FUZZY SYSTEMS (FUZZ-IEEE), 2016., 2016. **Anais...** [S.l.: s.n.], 2016. p.64–71.

LEROUGE, J.; LEGRAND, A. MetaSimGrid: Towards realistic scheduling simulation of distributed applications. **ENS-LIP Research Report**, [S.l.], v.28, p.2002, 2002.

ŁESKI, J. ε–insensitive learning techniques for approximate reasoning systems. **Int. J. Comput. Cognit**, [S.l.], v.1, n.1, p.21–77, 2003.

LIMA, M. P.; FONTES, C. H. d. O.; SCHNITMAN, L. **A lógica fuzzy do tipo 2 e um estudo de caso aplicado ao controle de tráfego aéreo**. [S.l.]: Universidade Federal da Bahia-UFBA, 2007.

MENDEL, J. M. **Uncertain rule-based fuzzy logic systems**: introduction and new directions. [S.l.]: Prentice Hall PTR Upper Saddle River, 2001.

MENDEL, J. M. Fuzzy sets for words: a new beginning. In: FUZZY SYSTEMS, 2003. FUZZ'03. THE 12TH IEEE INTERNATIONAL CONFERENCE ON, 2003. **Anais...** [S.l.: s.n.], 2003. v.1, p.37–42.

MITCHELL, H. B. Pattern recognition using type-II fuzzy sets. **Information Sciences**, [S.l.], v.170, n.2, p.409–418, 2005.

MOURA, B.; SOARES, Y.; SAMPAIO, L.; REISER, R.; YAMIN, A.; PILLA, M. fGrid: Uncertainty variables modeling for computational grids using fuzzy logic. In: IEEE INTERNATIONAL CONFERENCE ON FUZZY SYSTEMS (FUZZ-IEEE), 2016., 2016. **Anais...** [S.l.: s.n.], 2016. p.2249–2256.

MOURA, B.; SOARES, Y.; SAMPAIO, L.; REISER, R.; YAMIN, A.; PILLA, M. FUZZY SYSTEM MODELING FOR TASK SCHEDULING IN COMPUTATIONAL GRIDS. In: UNCERTAINTY MODELLING IN KNOWLEDGE ENGINEERING AND DECISION MAKING, 2016. **Anais...** WORLD SCIENTIFIC, 2016. p.806–811.

NAJAFI, A.; AMIRKHANI, A.; MOHAMMADI, K.; NAIMI, A. A novel soft computing method based on interval type-2 fuzzy logic for classification of celiac disease. In: IRANIAN CONFERENCE ON BIOMEDICAL ENGINEERING AND 2016 1ST INTERNATIONAL IRANIAN CONFERENCE ON BIOMEDICAL ENGINEERING (ICBME), 2016., 2016. **Anais...** [S.l.: s.n.], 2016. p.257–262.

NAYAK, P.; VATHASAVAI, B. Energy Efficient Clustering Algorithm for Multi-Hop Wireless Sensor Network Using Type-2 Fuzzy Logic. **IEEE Sensors Journal**, [S.l.], v.17, n.14, p.4492–4499, July 2017.

NEMETZ, R. **Otimizando o fluxo de tarefas em sistemas distribuídos de impressão**: um algoritmo de escalonamento dinâmico não preemptivo baseado em mecanismo de previsão. 2011. Dissertação (Mestrado em Ciência da Computação) — .

OJHA, V. K.; ABRAHAM, A.; SNášEL, V. Metaheuristic tuning of type-II fuzzy inference systems for data mining. In: IEEE INTERNATIONAL CONFERENCE ON FUZZY SYSTEMS (FUZZ-IEEE), 2016., 2016. **Anais...** [S.l.: s.n.], 2016. p.610–617.

OSTERMANN, S.; PRODAN, R.; FAHRINGER, T. Dynamic Cloud provisioning for scientific Grid workflows. In: GRID COMPUTING (GRID), 2010 11TH IEEE/ACM INTERNATIONAL CONFERENCE ON, 2010. **Anais...** [S.l.: s.n.], 2010. p.97–104.

PACHECO, P. Parallel Programming with MPI. Morgan Kau man, San Francisco. **CA, October**, [S.l.], 1996.

PINEDO, M. L. **Scheduling**: theory, algorithms, and systems. [S.l.]: Springer Science & Business Media, 2012.

QUINSON, M. GRAS: a research and development framework for grid and P2P infrastructures. In: IASTED INTERNATIONAL CONFERENCE ON PARALLEL AND DISTRIBUTED COMPUTING AND SYSTEMS, 18., 2006. **Anais...** [S.l.: s.n.], 2006.

QUINSON, M.; BOBELIN, L.; SUTER, F. Synthesizing generic experimental environments for simulation. In: P2P, PARALLEL, GRID, CLOUD AND INTERNET COMPUTING (3PGCIC), 2010 INTERNATIONAL CONFERENCE ON, 2010. **Anais...** [S.l.: s.n.], 2010. p.222–229.

REIS, V. Q. d. **Escalonamento em grids computacionais**: estudo de caso. 2005. Tese (Doutorado em Ciência da Computação) — Universidade de São Paulo.

RIBACIONKA, F. **Algoritmo distribuído para alocação de múltiplos recursos em ambientes distribuídos.** 2013. Tese (Doutorado em Ciência da Computação) — Universidade de São Paulo.

RIZOL, P.; MESQUITA, L.; SAOTOME, O. Lógica Fuzzy Tipo-2. **Revista Sodebras**, [S.l.], v.6, p.27–46, 2011.

RONG, H.; ZHIGANG, H. A scheduling algorithm aimed at time and cost for meta-tasks in grid computing using fuzzy applicability. In: HIGH-PERFORMANCE COMPUTING IN ASIA-PACIFIC REGION, 2005. PROCEEDINGS. EIGHTH INTERNATIONAL CONFERENCE ON, 2005. **Anais...** [S.l.: s.n.], 2005. p.6–pp.

ROSS, T. J. Fuzzy Control Systems. **Fuzzy Logic with Engineering Applications, Third Edition**, [S.l.], p.437–500, 2004.

RUAN, D.; KERRE, E. E. Fuzzy implication operators and generalized fuzzy method of cases. **Fuzzy Sets and systems**, [S.l.], v.54, n.1, p.23–37, 1993.

SAMPAIO, L.; SOARES, Y.; MOURA, B.; RIBEIRO, S.; REISER, R.; YAMIM, A.; PILLA, M. fGrid: Modelagem de Variáveis de Incerteza para Escalonamento em Grades Computacionais usando Lógica Fuzzy. In: WORKSHOP ESCOLA INFORMáTICA TEóRICA (WEIT 2015), 3., 2015. **Anais...** [S.l.: s.n.], 2015. p.128–137.

SANTIAGO, A. J. S.; YUSTE, A. J.; EXPOSITO, J. E. M.; GALÁN, S. G.; PRADO, R. P. d. A Multi-Criteria Meta-Fuzzy-Scheduler for Independent Tasks in Grid Computing. **Computing and Informatics**, [S.l.], v.30, n.6, p.1201–1223, 2012.

SANTOS NETO, E. L. dos. **Escalonamento de Aplicações que Processam Grandes Quantidades de Dados em Grids Computacionais**. 2004. Tese (Doutorado em Ciência da Computação) — Master's thesis, Coordenaçao de Pós-Graduaçao em Informática-Universidade Federal de Campina Grande.

SCHOPF, J. M. A General Architecture for Scheduling on the Grid. **Special issue of JPDC on Grid Computing**, [S.l.], 2002.

SENGER, H. et al. Improving scalability of Bag-of-Tasks applications running on master–slave platforms. **Parallel Computing**, [S.l.], v.35, n.2, p.57–71, 2009.

SHAW, I. S. **Controle e Modelagem Fuzzy**. [S.l.]: Edgard Blücher LTDA, 1999.

SILER, W.; BUCKLEY, J. J. **Fuzzy expert systems and fuzzy reasoning**. [S.l.]: John Wiley & Sons, 2005.

SOMASUNDARAM, T. S.; GOVINDARAJAN, K.; KIRUTHIKA, U.; BUYYA, R. Semantic-enabled CARE Resource Broker (SeCRB) for managing grid and cloud environment. **The Journal of Supercomputing**, [S.l.], v.68, n.2, p.509–556, 2014.

SOOD, D.; KOUR, H.; KUMAR, S. Survey of computing technologies: distributed, utility, cluster, grid and cloud computing. **Journal of Network Communications and Emerging Technologies (JNCET) www. jncet. org**, [S.l.], v.6, n.5, 2016.

SOUZA, P. d. AMIGO: Uma Contribuição para a Convergência na Área de Escalonamento de Processos. **Doutorado). ICMC-USP, São Carlos, Brasil**, [S.l.], 2000.

SUN, W.; ZHU, Y.; SU, Z.; JIAO, D.; LI, M. A priority-based task scheduling algorithm in grid. In: PARALLEL ARCHITECTURES, ALGORITHMS AND PROGRAMMING (PAAP), 2010 THIRD INTERNATIONAL SYMPOSIUM ON, 2010. **Anais...** [S.l.: s.n.], 2010. p.311–315.

TERZOPOULOS, G.; KARATZA, H. D. Bag-of-Tasks Load Balancing on Power-Aware Clusters. In: EUROMICRO INTERNATIONAL CONFERENCE ON PARALLEL, DISTRIBUTED, AND NETWORK-BASED PROCESSING (PDP), 2016., 2016. **Anais...** [S.l.: s.n.], 2016. p.135–142.

VAHDANI, B.; TAVAKKOLI-MOGHADDAM, R.; MOUSAVI, S. M.; GHODRATNAMA, A. Soft computing based on new interval-valued fuzzy modified multi-criteria decision-making method. **Applied Soft Computing**, [S.l.], n.0, p.–, 2012.

VAHDAT-NEJAD, H.; MONSEFI, R. Static Parallel Job Scheduling in Computational Grids. In: COMPUTER AND ELECTRICAL ENGINEERING, 2008. ICCEE 2008. INTERNATIONAL CONFERENCE ON, 2008. **Anais...** [S.l.: s.n.], 2008. p.548–552.

VIRIYAPANT, K.; SMANCHAT, S. A deadline-constrained scheduling for dynamic multi-instances parameter sweep workflow. In: COMPUTER AND INFORMATION SCIENCE (ICIS), 2016 IEEE/ACIS 15TH INTERNATIONAL CONFERENCE ON, 2016. **Anais...** [S.l.: s.n.], 2016. p.1–6.

VON ALTROCK, C. **Fuzzy logic and neurofuzzy applications in business and finance**. [S.l.]: Prentice-Hall, Inc., 1996.

WAGNER, C. Juzzy - A Java based toolkit for Type-2 Fuzzy Logic. In: IEEE SYMPOSIUM ON ADVANCES IN TYPE-2 FUZZY LOGIC SYSTEMS (T2FUZZ), 2013., 2013. **Anais...** [S.l.: s.n.], 2013. p.45–52.

WAGNER, C.; PIERFITT, M.; MCCULLOCH, J. Juzzy online: An online toolkit for the design, implementation, execution and sharing of Type-1 and Type-2 fuzzy logic systems. In: IEEE INTERNATIONAL CONFERENCE ON FUZZY SYSTEMS (FUZZ-IEEE), 2014., 2014. **Anais...** [S.l.: s.n.], 2014. p.2321–2328.

WANG, S.-D.; HSU, I.-T.; HUANG, Z.-Y. Dynamic scheduling methods for computational grid environments. In: INTERNATIONAL CONFERENCE ON PARALLEL AND DISTRIBUTED SYSTEMS (ICPADS'05), 11., 2005. **Anais...** [S.l.: s.n.], 2005. v.1, p.22–28 Vol. 1.

WANG, W.; XIN, X. Distance measure between intuitionistic fuzzy sets. **Pattern Recognition Letters**, [S.l.], v.26, n.13, p.2063–2069, 2005.

XU, Z.; HOU, X.; SUN, J. Ant algorithm-based task scheduling in grid computing. In: ELECTRICAL AND COMPUTER ENGINEERING, 2003. IEEE CCECE 2003. CANADIAN CONFERENCE ON, 2003. **Anais...** [S.l.: s.n.], 2003. v.2, p.1107–1110.

YAGER, R. R. On the implication operator in fuzzy logic. **Information Sciences**, [S.l.], v.31, n.2, p.141–164, 1983.

YAGER, R. R. On global requirements for implication operators in fuzzy modus ponens. **Fuzzy Sets and Systems**, [S.l.], v.106, n.1, p.3–10, 1999.

YAGER, R. R. On some new classes of implication operators and their role in approximate reasoning. **Information Sciences**, [S.l.], v.167, n.1, p.193–216, 2004.

YE, J. Cosine similarity measures for intuitionistic fuzzy sets and their applications. **Mathematical and Computer Modelling**, [S.l.], v.53, n.1, p.91–97, 2011.

YU, K. M.; CHEN, C. K. An Adaptive Scheduling Algorithm for Scheduling Tasks in Computational Grid. In: SEVENTH INTERNATIONAL CONFERENCE ON GRID AND COOPERATIVE COMPUTING, 2008., 2008. **Anais...** [S.l.: s.n.], 2008. p.185–189.

YU, K.-M.; LUO, Z.-J.; CHOU, C.-H.; CHEN, C.-K.; ZHOU, J. A fuzzy neural network based scheduling algorithm for job assignment on computational grids. In: INTERNATIONAL CONFERENCE ON NETWORK-BASED INFORMATION SYSTEMS, 2007. **Anais...** [S.l.: s.n.], 2007. p.533–542.

ZADEH, L. A. Fuzzy sets. **Information and control**, [S.l.], v.8, n.3, p.338–353, 1965.

ZADEH, L. A. Outline Of A New Approach To The Analysis Of Complex Systems And Decision Processes. **Systems, Man and Cybernetics, IEEE Transactions on**, [S.l.], n.1, p.28–44, 1973.

ZADEH, L. A. Fuzzy logic and approximate reasoning. **Synthese**, [S.l.], v.30, n.3-4, p.407–428, 1975.

ZADEH, L. A. Fuzzy logic, neural networks, and soft computing. **Communications of the ACM**, [S.l.], v.37, n.3, p.77–84, 1994.

ZANOTELLI, R. M.; REISER, R. H. S.; CAVALHEIRO, S. C.; FOSS, L.; BEDREGAL, B. Towards Robusteness and Duality Analysis of Intuitionistic Fuzzy Aggregations. In: IEEE INTERNATIONAL CONFERENCE ON FUZZY SYSTEMS (FUZZ IEEE 2015) - ISTANBUL TURKEY, 2015. **Anais...** [S.l.: s.n.], 2015.