UNIVERSIDAD DE JAÉN

ESCUELA POLITÉCNICA SUPERIOR DE LINARES
DEPARTAMENTO DE INGENIERÍA DE TELECOMUNICACIÓN

TESIS DOCTORAL

PLANIFICACIÓN LOCAL DE TAREAS CON SISTEMAS BASADOS EN REGLAS BORROSAS. APLICACIÓN A LA DETECCIÓN DE DEFECTOS EN TABLEROS DE FIBRA

PRESENTADA POR:
ANTONIO JAVIER SÁNCHEZ SANTIAGO

DIRIGIDA POR:
DR. D. SEBASTIÁN GARCÍA GALÁN
DR. D. JOSÉ ENRIQUE MUÑOZ EXPÓSITO

JAÉN, 16 DE NOVIEMBRE DE 2012

ISBN 978-84-8439-693-2

A todos aquellos quienes
en un momento dado
decidan ponerlo
en práctica

Agradecimientos

Esta tesis, enmarcada en el Proyecto de Excelencia de la Junta de Andalucía titulado "Modelo de Adopción de la Tecnología GRID de la Información en la Empresa Andaluza", supone el culmen, el punto de llegada en mi trabajo durante tres años en la Escuela Politécnica Superior de Linares y a la misma vez el punto de partida de ese futuro, que espero, prometedor.

Cuando me pongo a redactar esta página bajo el título de "Agradecimientos" me entra una duda al pensar que mis dedos que surcan el teclado no sepan plasmar lo que mi mente está pensando; que en esas "lagunas" se me olvide alguien que en un momento de ese recorrido me haya dado su ánimo, su punto de vista, su visión distinta a la mía, su saber, en fin algo de su propia vida. Por eso en primer lugar quiero agradecerles a todos aquellos cuyo nombre no aparece aquí y que saben y recuerdan que una vez estuvieron a mi lado.

La realización de esta tesis ha suspuesto para mí un gran objetivo, un gran reto, puesto que no hace mucho tiempo, este punto final me parecía muy lejano, difícil, casi inalcanzable; pues el día a día me hacía ver las cosas un poco más inaccesibles, imprecisas, como si estuviese en una profunda crisis. A día de hoy este reto, este objetivo lo he conseguido gracias al espíritu de superación que caracteriza al ser humano y a la guía y conducción de mis tutores, el Dr. D. Sebastian García y el Dr. D. José Enrique Muñoz y al apoyo tanto moral como personal que me han brindado siempre mis padres, mi hermano y María Luisa.

Durante el tiempo de estudio, investigación y trabajo para la realización de esta tesis doctoral, me he encontrado con muchas personas de las que guardo un grato recuerdo. De ellos he aprendido mucho y desde aquí quiero dejar constancia de su participación e implicación para con este trabajo. A Dr. D. Sebastián Bruque, investigador principal del proyecto por confiar en mí y darme la oportunidad de iniciar mi carrera en el mundo universitario e investigador; a mis compañeros de proyecto de investigación Dr. D. Antonio J. Yuste, Dr. D. Juan M. Maqueira y Dra. D^{a}. Rocío Pérez a los que quiero agradecer sus numerosas colaboraciones, aportaciones y a su apoyo siempre que lo he necesitado.

Durante mi estancia en la EPS de Linares he compartido despacho, pero sobre todo vivencias y sintonía, con un grupo de compañeros excepcionales. Con Julio y Pablo con quienes, al principio, el silencio reinaba en el lugar. Con Fran, David, Rocío y Juan el grupo aumentó y las conversaciones y diálogos fueron enriquecedores. Algún tiempo después este grupo se incrementó con Amparo, José Guadalupe y Pedro; y junto con todos seguí compartiendo mis últimos días

de trabajo.

Finalmente quiero agradecer a todo el personal de mi departamento de “adopción”, Ingeniería de Telecomunicación, su apoyo y asesoramiento en cualquier campo universitario e investigador.

Con todos y a todos, GRACIAS.

Antonio Javier Sánchez Santiago
Martos, Julio 2012

Resumen

Los sistemas Grid surgen como tecnologías innovadoras que permiten agregar y compartir distintos recursos informáticos, tanto de almacenamiento y de procesamiento, geográficamente distribuidos sobre redes de alta velocidad. Esta tecnología Grid posee una serie de características especiales que la hace diferente de los sistemas distribuidos tradicionales. Estos nuevos rasgos diferenciadores son debidos, principalmente, a que un sistema Grid está formado por recursos pertenecientes a distintas organizaciones, por lo tanto, una de las principales características de las tecnologías Grid viene dada por la heterogeneidad de los recursos distribuidos que componen el sistema. Estos recursos son altamente dinámicos y heterogéneos, debido a que pueden estar siendo compartidos en el sistema Grid y a la vez utilizados por sus propietarios, e imprecisos, puesto que pueden unirse o abandonar el sistema Grid en cualquier momento.

Como toda nueva tecnología, su desarrollo produce nuevas oportunidades y retos de investigación. Uno de estos nuevos retos u oportunidades de investigación reside en la planificación de tareas. Debido a la alta heterogeneidad y dinamismo de los recursos que forman estos sistemas los planificadores de los sistemas distribuidos tradicionales no son adecuados en Grid. Por lo tanto, hay una necesidad de desarrollar nuevos planificadores de tareas que sean capaces de administrar las nuevas características de los recursos Grid. Para obtener una planificación de tareas que satisfaga diversas necesidades (tiempo de ejecución mínimo, balance de carga, etc), es imprescindible conocer el estado de los recursos formantes del Grid.

En esta tesis, para gestionar los recursos y nodos formantes de un sistema Grid se proponen dos planificadores que tienen en cuenta el estado y la heterogeneidad de los recursos del Grid para llevar a cabo la ejecución de un conjunto de tareas. En primer lugar, se propone un planificador local balanceado que administra los recursos locales de un Grid asignando a cada recurso una tarea adecuada a su capacidad, consiguiendo así que todos los recursos finalicen sus tareas al mismo tiempo, evitando la espera producida por la ejecución en un recurso con menor capacidad. El segundo planificador propuesto para sistemas Grid es un Meta-Planificador basado en reglas borrosas. Los sistemas basados en reglas borrosas son capaces de gestionar de forma eficiente la incertidumbre y la imprecisión. En esta tesis, las reglas borrosas manejan la información imprecisa inherente al estado de los nodos de un sistema Grid. El Meta-Planificador borroso asigna la siguiente tarea a ejecutar al nodo Grid más adecuado, usando para tal elección un sistema borroso basado en reglas.

Ambos planificadores, han sido verificados en diversos escenarios heterogéneos demostrando que su aplicación en sistemas Grid puede resultar beneficiosa. En concreto, se demuestra que estos planificadores cumplen dos objetivos: minimizan el tiempo de ejecución de un conjunto de tareas y obtienen un balance de carga equilibrado entre todos los nodos y recursos que forman el sistema Grid.

La principal funcionalidad de los sistemas Grid es la ejecución de tareas con grandes necesidades de cómputo. A fin de comprobar, sobre un problema concreto, el funcionamiento de las reglas borrosas en la planificación Grid se desarrolla un Planificador Local Borroso. Este planificador es aplicado, sobre un sistema Desktop Grid, a un problema con grandes necesidades de cómputo, la detección de defectos en tableros de fibra recubiertos de papel melamínico, usando para tal detección análisis de imágenes. Su comportamiento es analizado en base a la reducción en el tiempo de detección de defectos con respecto al tiempo empleado por un operario y al aumento de la tasa de defectos detectados.

Palabras clave: Grid Computing, Sistemas Grid, Desktop Grid, Sistemas Borrosos basados en Reglas, Planificador Local, Meta-Planificador, dinamismo, heterogeneidad, imprecisión, balance de carga, análisis de imágenes, tableros de fibra, tasa de defectos.

Abstract

Grid systems appear as one of the most innovative technologies in current computer science, they allow us to add and share geographically distributed storage and processing computing resources via high speed networks. Grid technology is different from traditional distributed systems, since a Grid system is made up of resources belonging to different organizations. These resources are highly dynamic, heterogeneous and imprecise, they may be shared on the Grid system at the same time they are being used by their owners who can join or leave the Grid system at any time.

The development of new technologies challenges our research capability, one of these new challenges lies in the tasks scheduling. Due to the high heterogeneity and dynamism of the resources making up Grid systems, traditional scheduler for distributed systems is not suitable in Grid technology. Therefore, it is necessary to develop new schedulers which are able to manage the new features arising from Grid resources. To obtain a task scheduling providing a feasible execution time, good load balancing, etc. it is essential to know the status of Grid resources.

In this thesis, two schedulers considering the state and the heterogeneity of Grid resources to carry out the execution of a set of tasks are proposed. The first proposition consists of a local balanced scheduler that manages the local resources of a Grid by assigning each task to a resource according to its capacity. The general idea is to make all the resources complete their tasks at the same time, avoiding the timeout caused by the execution of a task in a resource with lower capacity. The second proposed scheduler is a Meta-Scheduler based on fuzzy rules. Systems based on fuzzy rules are able to manage uncertainty efficiently. Fuzzy rules are reliable to manage imprecise information inherent to the state of nodes in a Grid system. The Meta-Scheduler assigns the next task to execute at the most appropriate Grid node using the fuzzy rule system.

Both schedulers have been verified in several heterogeneous scenarios of interest demonstrating its applicability in Grid. Particularly, these schedulers are found to minimize the execution time of a set of tasks and to obtain a good load balancing between nodes and resources that form the Grid system.

The main functionality of a Grid system lies in the execution of tasks with huge requirements of computing. In order to verify on a specific problem the fuzzy rules in Grid scheduling, a Fuzzy Local Scheduler is developed over a Desktop Grid system. The goal is to detect defects

in fiberboard melamine paper, using for such purpose the image analysis. The efficiency of the Scheduler is analyzed in terms of the time reduction for defect detection with respect to the time taken by an operator as well as the rate of detected defects.

Keywords: Grid Computing, Grid System, Desktop Grid, Fuzzy Rules, Local Scheduler, Meta-Scheduler, dynamism, heterogeneity, imprecise, load balancing, image analysis, fiberborad, rate of defects.

Índice general

Índice de figuras

Índice de tablas

Parte I

Planteamiento de la Investigación

Capítulo 1

Introducción

1.1. Contexto y motivación de la investigación

La popularidad de Internet y la disponibilidad de potentes recursos informáticos interconectados por redes de alta velocidad han cambiado la forma en que son usados los ordenadores a día de hoy. Esta oportunidad tecnológica ha llevado a la posibilidad de utilizar las redes de ordenadores como una unidad de computación. Es posible agrupar una amplia variedad de recursos, incluyendo superordenadores, sistemas de almacenamiento, fuentes de datos, y dispositivos especiales, distribuidos geográficamente, y usarlos como único recurso unificado formando, lo que se conoce como, un sistema Grid.

Un sistema Grid tiene como objetivo agregar múltiples recursos heterogéneos, a gran escala, para proporcionar transparencia, seguridad y acceso coordinado a los diversos recursos informáticos, propiedad de varias instituciones que forman una organización virtual. Por otro lado, un sistema Desktop Grid tiene como objetivo aprovechar la inactividad de una serie de ordenadores de escritorio, propiedad de distintos individuos pertenecientes a una organización. En un sistema Desktop Grid, los voluntarios (es decir, los proveedores de recursos) tienen propiedades heterogéneas tales como CPU, memoria, conexión de red, etc. Además, están expuestos a fallos y pueden participar libremente en el sistema. Por lo tanto, las tres cuestiones principales que caracterizan a ambos sistemas Grids son: heterogeneidad, escalabilidad y adaptabilidad dinámica.

Para gestionar la ejecución de un conjunto de tareas sobre un sistema Grid y cumplir una serie de objetivos, como obtener un tiempo de ejecución mínimo, es necesario conocer el estado de los recursos que forman este sistema. La información sobre el estado de los recursos de un Grid es muy importante para cualquier planificador, especialmente con la naturaleza dinámica y heterogénea de un sistema Grid. En el tratamiento de información sujeta a imprecisiones, es importante destacar el papel de la lógica borrosa. La lógica borrosa tiene la propiedad de poder manejar imprecisión y vaguedad en la información que utiliza. A pesar de la existencia de una gran diversidad de métodos de planificación que se pueden encontrar en la literatura, la gestión eficiente de la incertidumbre y dinamismo inherente a los recursos de los sistemas

Grid, sigue siendo un reto en la planificación de estos Sistemas. En el ámbito científico actual, existen importantes lagunas en sistemas de planificación que utilicen controladores borrosos para la planificación de tareas en sistemas Grid y Desktop Grid.

Para su funcionamiento, ambos sistemas Grid pueden utilizar recursos ociosos, ya existentes, consiguiendo así grandes capacidades de cómputo. Un campo con enormes necesidades de computación es la visión por computador. La visión por computador es un factor clave para la implementación de la calidad total dentro de los diferentes procesos de automatización industrial. La utilización de esta tecnología por parte de las empresas lleva consigo una ventaja competitiva, pudiendo tener un aumento de producción, mejora en la calidad de los productos, como también la reducción de costes de fabricación. Además permite inspeccionar los procesos de producción en forma precisa. Una de las técnicas pertenecientes a visión por computador es el análisis de imágenes. Típicamente las imágenes en dos dimensiones contienen un gran número de píxeles, varios millones; incluso un proceso puntual que simplemente modifica el valor de cada píxel de acuerdo a su contenido anterior requiere dirigir cada píxel. Para un proceso basado en vecindad, cada píxel debe ser abordado muchas veces, por lo tanto la velocidad de procesamiento y de acceso a memoria son requisitos indispensables para este tipo de trabajos.

Una de las principales líneas de investigación de visión por computador, es el proceso de reconocimiento de defectos de manera automática, debido a que la automatización permite establecer políticas y formas de control precisas y objetivas; en cambio los sistemas manuales están afectados por la rutina por parte del operador, causando un control deficiente o inconstante. En esta memoria, se expone el análisis de una serie de imágenes de tableros de fibra recubiertos con papel melamínico para detectar, en tiempo real, defectos en estos tableros. Para afrontar las grandes necesidades de computación de los procesos de análisis de imágenes, serán aprovechadas las capacidades de cómputo ofrecidas por un sistema Desktop Grid, con un planificador local de tareas con sistemas expertos basado en reglas borrosas. Esta fusión de tecnologías aún no ha sido estudiada, por este motivo, es necesario conocer las posibilidades de su aplicación real. Por lo tanto, se abre aquí un campo no explorado como es el análisis de imágenes usando la potencia de cómputo de un sistema Desktop Grid, cuyo planificador de tareas está basado en reglas borrosas.

1.2. Justificación y objetivos de la investigación

La presente investigación persigue los siguientes objetivos generales:

1. *Estudio y desarrollo de un Planificador Local Balanceado.*

   El primer objetivo es la creación de un Planificador Local que posibilite la ejecución de un conjunto de tareas en un ámbito local. Este planificador deberá distribuir, de forma equitativa, la carga de trabajo entre todos los recursos disponibles.

2. *Estudio, desarrollo de un Meta-Planificador basado en reglas borrosas para Sistemas Grid.*

El objetivo debe ser la aplicación de sistemas borrosos para la creación de un Meta-Planificador basado en reglas borrosas que permita asignar a los nodos formantes del Grid una serie de tareas minimizando el tiempo de ejecución y balanceando la utilización de dichos nodos.

3. *Desarrollo de un Planificador Local basado en sistemas borrosos y su verificación.*

   En este objetivo se desarrollará y comprobará el comportamiento de un planificador local basado en reglas borrosas en la ejecución de una serie de algoritmos de análisis de imágenes con grandes necesidades de cómputo. Se analizará la eficiencia de la metodología propuesta en base a: la reducción del tiempo de detección de defectos, con respecto al tiempo empleado por un operario y, al aumento de la tasa de defectos detectados.

Los objetivos específicos, que dan lugar a las respectivas actividades son los siguientes:

1. Identificar, a partir de la literatura científica previa, el desarrollo actual de los sistemas Grid y Desktop Grid, mostrando especial atención a los planificadores de tareas, tanto locales como Meta-Planificadores en estos sistemas.
2. Identificar, a partir de la literatura científica existente, los sistemas borrosos basados en reglas que, serán utilizados en esta investigación para el desarrollo de un Meta-Planificador Borroso.
3. Identificar, a partir de la literatura científica previa, diversas técnicas de análisis de imágenes, que serán utilizadas para la detección de defectos en tableros de fibra recubiertos con papel melamínico.
4. Desarrollar un planificador local balanceado de tareas.
5. Desarrollar un Meta-Planificador con sistemas expertos basado en reglas borrosas.
6. Desarrollar un Planificador Local basado en reglas borrosas.
7. Estudiar y desarrollar un sistema de procesamiento de imágenes capaz de detectar defectos en tableros de fibra recubiertos con papel melamínico.
8. Verificar el comportamiento del planificador local borroso, usando un problema real con grandes necesidades de cómputo.

1.3. Estructura de la tesis

En esta sección se presenta la estructura de la tesis, figura 1.1, que recoge el trabajo de investigación desarrollado. Se estructura en tres bloques temáticos. Por su parte, cada bloque temático está compuesto por una serie de capítulos.

1. Revisión de conocimientos. Este bloque se compone de tres capítulos:

   - **Capítulo 2**: Planificación de Tareas en Sistemas Grid. Este capítulo presenta los conceptos clave de los sistemas Grid. Se expone una taxonomía de las aplicaciones Grid. Además se repasan los tipos de sistemas Grid, dedicando especial interés a los sistemas Desktop Grid. También es detallada la planificación en sistemas Grid y las heurísticas de planificación de tareas.
   - **Capítulo 3**: Sistemas Borrosos. Este capítulo repasa el funcionamiento de un sistema borroso. Así, se revisan las nociones de los conjuntos borrosos y la arquitectura de un sistema experto con reglas borrosas.
   - **Capítulo 4**: Análisis de imágenes. En este capítulo se describen diversas técnicas de análisis de imágenes. Se muestran las técnicas de realce de imagen, segmentación y procesamiento morfológico.

2. Desarrollo y metodología de la investigación. Este bloque de la memoria se divide en dos capítulos:

   - **Capítulo 5**: Planificador de Tareas en Grid: Planificador Local Balanceado y Meta-Planificador de Tareas con Sistemas Expertos basado en Reglas Borrosas. En este capítulo se describen los componentes de un sistema Grid y se desarrolla un Planificador Local Balanceado y un Meta-Planificador basado en Reglas Borrosas.
   - **Capítulo 6**: Detección de defectos en Tableros de Fibra recubiertos con Papel Melamínico. Este capítulo aborda el desarrollo de un sistema de procesamiento y análisis de imágenes capaz de detectar defectos en tableros de fibra y su verificación usando un sistema Grid con un planificador basado en reglas borrosas.

3. Conclusiones, Líneas Futuras y Publicaciones Generadas.

   - **Capítulo 7**: Finalmente, se señalan las conclusiones, las futuras líneas de investigación a seguir y las publicaciones derivadas de la investigación. La memoria concluye con una recopilación bibliográfica de las contribuciones más destacadas de la materia estudiada.

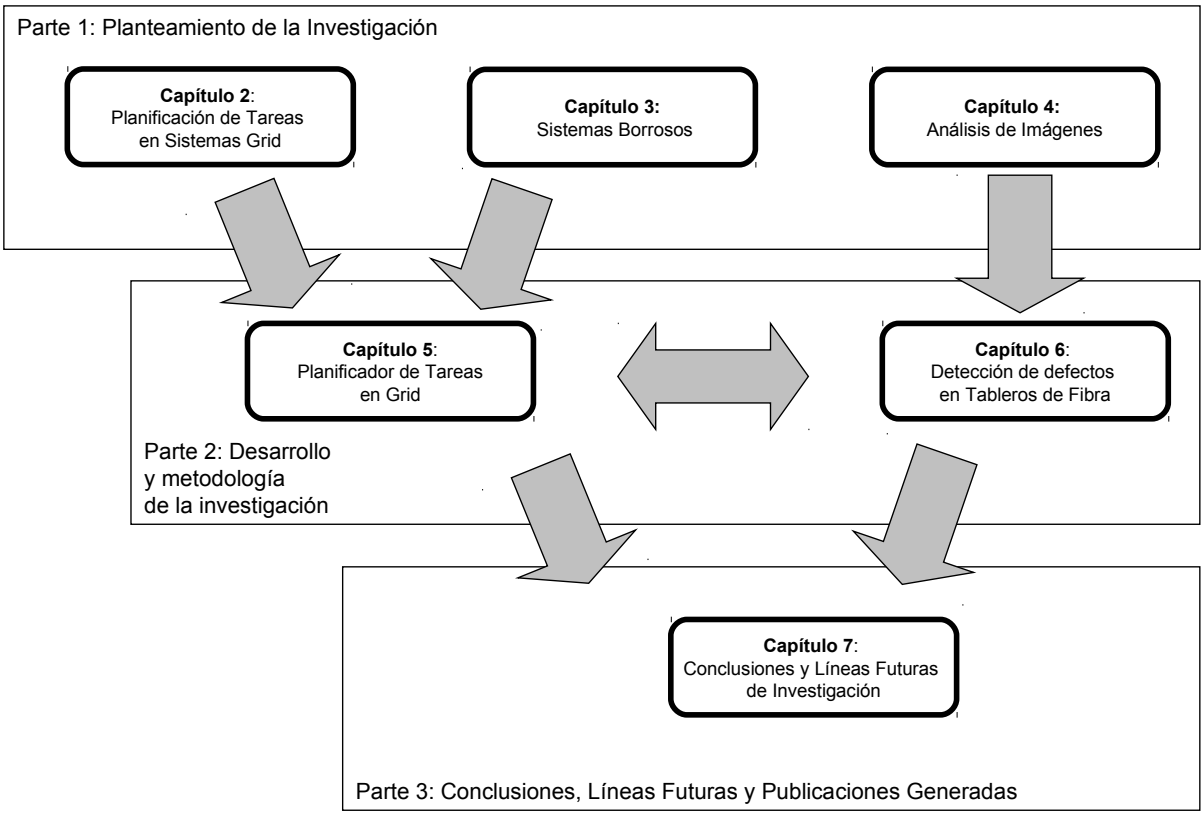


Figura 1.1: Esquema de la Memoria de Tesis

Parte II

Revisión de Conocimientos

Capítulo 2

Planificación de Tareas en Sistemas Grid

2.1. Introducción

El desarrollo de las redes de comunicaciones de alta velocidad, el incremento de la capacidad de cómputo y el bajo coste de los sistemas de almacenamiento han originado un paradigma en computación distribuida: Grid Computing [Foster04a]. Mediante esta tecnología se propone la utilización de recursos computacionales distribuidos para, mediante la agregación y compartición, formar un ordenador virtual con capacidades computacionales superiores a las de los superordenadores actuales [Abbas04].

El principal reto en el campo de Grid Computing radica en resolver, de una forma eficaz y segura, el uso de millones de ordenadores de todo el mundo, que van desde simples dispositivos de mano [Chu04], hasta clusters de ordenadores o de superordenadores conectados a través de redes, para obtener un objetivo común. Este concepto abre un mundo de posibilidades, que puede permitir, por ejemplo, el acceso a recursos computacionales de forma similar a como se accede a determinados servicios como puede ser la energía eléctrica [Maqueira08].

En los últimos años Grid Computing y las tecnologías Grid han tenido un enorme auge en la resolución de problemas científicos y técnicos, usando recursos de computación que se encuentran distribuidos a través de Internet. De esta forma es posible crear superordenadores virtuales que ofrecen una gran cantidad de cómputo, permitiendo resolver problemas complejos en un tiempo menor que el empleado hasta ahora. Así en los últimos años Grid Computing ha sido utilizado en campos tan diversos como meteorología [Tinghuai10], física [Andreeva08] o medicina [Ellisman04]. Algunos ejemplos de estas aplicaciones son usadas a gran escala en: optimización [Nebro07] [Lim07], computación colaborativa [Xhafa10a] y computación intensiva [Brasileiro11].

Para la mayoría de los sistemas Grid, el problema de alojar las tareas, de una forma dinámica y adaptativa en los recursos disponibles en respuesta a la demanda, está todavía sin resolver. En

los casos más simples, planificar la ejecución de tareas, puede ser llevado a cabo de una manera sencilla, asignando tareas entrantes a los recursos de acuerdo a su disponibilidad. Sin embargo, es más rentable usar planificadores más avanzados y sofisticados. Por otra parte, se espera que los planificadores actúen de forma coordinada con la dinámica del sistema Grid, evaluando la carga de trabajo de los recursos o notificando cuando estos recursos se unan o abandonan el sistema.

A continuación, una vez introducidos los conceptos clave en los sistemas Grid se realiza, atendiendo a su naturaleza, una clasificación de los tipos de sistemas Grid existentes en la actualidad. La planificación de tareas en sistemas Grid es repasada, introduciendo las nuevas características de esta planificación, su terminología y sus fases. Para finalizar el capítulo, se presentan las heurísticas tradicionales más importantes de planificación en Grid.

2.2. Computación Distribuida y Paralela

Los sistemas distribuidos pueden ser definidos como sistemas informáticos que contienen múltiples procesadores conectados en una red de comunicación, mediante la cual, los procesadores se comunican entre sí mediante el envío de mensajes. Por el contrario, los sistemas paralelos constan de varios procesadores que se comunican entre sí utilizando memoria compartida. Los sistemas distribuidos están diseñados para que muchos usuarios trabajen de forma conjunta, mientras que los sistemas paralelos están diseñados para lograr la máxima rapidez en un único problema [Gelado10].

Estos sistemas distribuidos son cada vez más asequibles debido a la disminución de los precios de los ordenadores y la disponibilidad de redes de alta velocidad para conectarse. Sin embargo, a pesar de la gran disponibilidad de hardware existente para los sistemas distribuidos, hay pocas aplicaciones de software que se aprovechan de este hardware [Kopetz11]. Una razón importante es que el software distribuido requiere un conjunto de técnicas y herramientas diferentes del software secuencial tradicional. Aunque los algoritmos distribuidos a menudo están compuestos por un pequeño incremento de líneas de código, su comportamiento y sus propiedades pueden ser difíciles de entender. A pesar de esto, el uso de sistemas distribuidos ofrece muchas ventajas con respecto al uso de sistemas paralelos:

- Escalabilidad. Los sistemas distribuidos son inherentemente más escalables que los sistemas paralelos. En los sistemas paralelos la memoria compartida llega a ser un "cuello de botella" cuando es incrementado el número de procesadores.
- Modularidad y heterogeneidad. Un sistema distribuido es más flexible, porque un simple procesador puede ser añadido o eliminado fácilmente. Además este procesador puede ser distinto a los ya existentes.
- Compartir datos. Los sistemas distribuidos, al estar formados por recursos de distintas empresas, proporcionan intercambio de datos; así varias empresas pueden compartir sus datos.

- Compartir recursos. Los sistemas distribuidos contemplan compartir recursos, por ejemplo un procesador de propósito especial y costoso, puede ser compartido por diferentes organizaciones.
- Estructura Geográfica. Las estructura geográfica es una característica básica asociada a la definición de los sistemas distribuidos. Comúnmente, los sistemas paralelos no se encuentran distribuidos geográficamente.
- Fiabilidad. Los sistemas distribuidos son más fiables que los sistemas paralelos, puesto que el fallo de un simple recurso no afecta al resto.
- Bajo coste. El precio de los ordenadores de consumo y la disponibilidad de redes hacen que crear un sistema distribuido sea bastante más barato que un sistema paralelo.

La computación distribuida está en el corazón de muchas aplicaciones. Mientras que la computación paralela se centra principalmente en la eficiencia, la computación distribuida es capaz de manejar la incertidumbre generada por la multiplicidad de flujos de control, la ausencia de memoria compartida y la ocurrencia de fallos.

2.3. Breve taxonomía de aplicaciones Grid

Los sistemas Grid proporcionan grandes capacidades de cómputo que implican capacidades computacionales fiables y generalizadas. Existen varias clases principales de aplicaciones Grid: alto rendimiento de computación, supercomputación distribuida, demanda de capacidades de cómputo, uso intensivo de datos y computación colaborativa [Foster04a].

- Las aplicaciones de alto rendimiento de computación utilizan el sistema Grid para ejecutar un gran número de tareas débilmente acopladas o independientes con el objetivo de usar el mayor número de ciclos de procesador posibles (a menudo estos ciclos provienen desde recursos ociosos). El resultado puede ser, como en supercomputación distribuida, la focalización de los recursos disponibles en un sólo problema. La naturaleza de las tareas conduce a diferentes tipos de problemas que implican métodos de resolución muy variados. Un sistema de este tipo es *Condor* [Thain05] de la Universidad de Wisconsin, que se utiliza para gestionar cientos de puestos de trabajo en universidades de todo el mundo.
- Las aplicaciones distribuidas de supercomputación usan Grid para agregar recursos informáticos con el fin de hacer frente a problemas que no pueden ser resueltos en un sólo recurso. Dependiendo del Grid en el que se trabaje, estos recursos agregados pueden incluir la mayoría de las supercomputadoras en un país o, simplemente, todas las estaciones de trabajo dentro de una empresa. Algunos ejemplos de aplicaciones de este tipo son simulaciones interactivas o simulaciones de procesos físicos complejos. Las simulaciones interactivas distribuidas son una técnica utilizada para el entrenamiento y la planificación en el ejército.

Escenarios realistas pueden incluir cientos de miles de entidades, cada una con complejos patrones de comportamiento. También han sido utilizadas, con éxito, supercomputadoras aparejadas para aplicaciones de modelización del clima, cosmología y aplicaciones de química computacional.

- Las aplicaciones "en demanda" necesitan de capacidades Grid para satisfacer a corto plazo necesidades de recursos (software, repositorios de datos, sensores, etc.) que no son rentables poseer. En contraste con las aplicaciones distribuidas de supercomputación, estas aplicaciones son, a menudo, impulsadas por preocupaciones de rentabilidad en lugar de rendimiento. Estas aplicaciones permiten usar recursos de computación de forma similar a como se accede a determinados servicios como puede ser la energía eléctrica.

- Las aplicaciones intensivas de datos están centradas en la síntesis de información de los datos que se encuentran geográficamente distribuidos en distintos repositorios, bibliotecas digitales y bases de datos. El proceso de síntesis es, habitualmente, intensivo desde el punto de vista computacional. Los experimentos de física de alta energía generan terabytes de datos al día, alrededor de un petabyte al año. La comunidad científica que tendrá acceso a estos datos se encuentra geográficamente distribuida, por lo tanto los sistemas de información en los que los datos se guardan, probablemente se distribuyan también de manera distribuida. Las aplicaciones de sistemas meteorológicos de previsión hacen un uso extensivo de datos para incorporar las observaciones a distancia vía satélite.

- Las aplicaciones de computación colaborativa, están referidas principalmente a la mejora de las interacciones humanas y a compartir recursos. Este tipo de aplicaciones se estructuran, a menudo, en un espacio virtual compartido. Muchas aplicaciones de colaboración tratan de hacer posible el uso compartido de recursos computacionales, tales como archivos de datos y simulaciones. Compartir no es simplemente el intercambio de documentos sino, que implica directamente el acceso a software de control remoto, computadoras, datos, sensores y otros recursos. Por ejemplo, los miembros de un consorcio pueden proporcionar acceso a software especializado y a datos, y/o compartir sus recursos computacionales. De manera más abstracta, lo que estos dominios de aplicaciones de colaboración tienen en común es la necesidad de distribución de los recursos y resolver problemas en organizaciones dinámicas virtuales y multi-institucionales. El intercambio que un sistema Grid propone, está referido a un acceso directo a las computadoras, software, datos y otros recursos, como requerido para la resolución de una amplia gama de problemas. Este intercambio está muy controlado, con la definición clara por parte de proveedores y consumidores de lo que se comparte, quién está autorizado a compartir, y las condiciones en que se produce el intercambio.

  Un conjunto de individuos y/o instituciones definidas por las reglas de distribución anteriores forman una Organización Virtual (VO), un concepto que se está convirtiendo en fundamental en el mundo de la computación moderna. VOs permiten que diferentes grupos

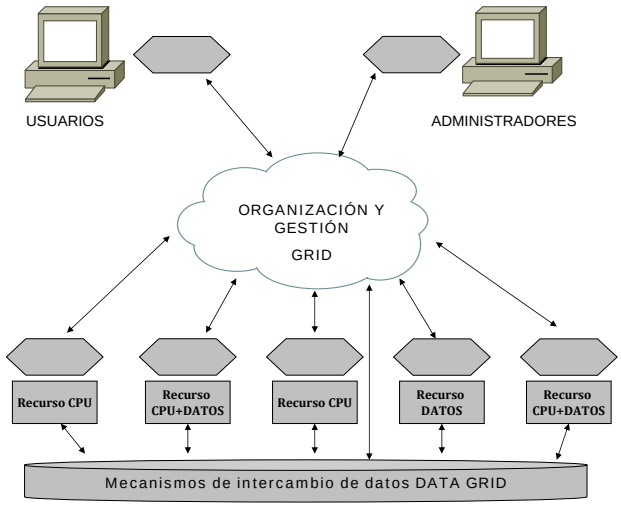


Figura 2.1: InfraGrid o Grid Sencillo. Fuente: Maqueira 2008

de organizaciones y/o personas puedan compartir los recursos de una manera controlada, por lo que que los miembros pueden colaborar para lograr un objetivo común. Las VOs pueden variar mucho en su propósito, alcance, tamaño, duración, estructura, comunidad, y la sociología [Foster04a]. Sin embargo, son identificados entre todas las organizaciones, un amplio conjunto de preocupaciones comunes y grandes necesidades de recursos computacionales.

2.4. Tipos de Sistemas Grid

Durante los últimos años los tipos de sistemas Grid han evolucionado de forma ininterrumpida. Debido a la diversa terminología encontrada en la literatura sobre sistemas Grid es necesario aclarar y clasificar, atendiendo a diversos criterios, los distintos tipos de sistemas Grid existentes. En esta sección se realiza un repaso de los tipos de sistemas Grid existentes siguiendo tres clasificaciones: según el ámbito organizativo, según el ámbito de los recursos y según su finalidad. A continuación se exponen estas clasificaciones.

2.4.1. Tipología según el ámbito organizativo

La clasificación que [Berstis02] realiza distingue entre tres tipos de sistemas Grid:

- InfraGrid o Grid Sencillo. Un sistema Grid formado por pocas máquinas de similar arquitectura hardware y conectadas en una red de área extensa forman un InfraGrid (figura 2.1). Este sistema suele ser usado por varios departamentos, aunque no contiene políticas

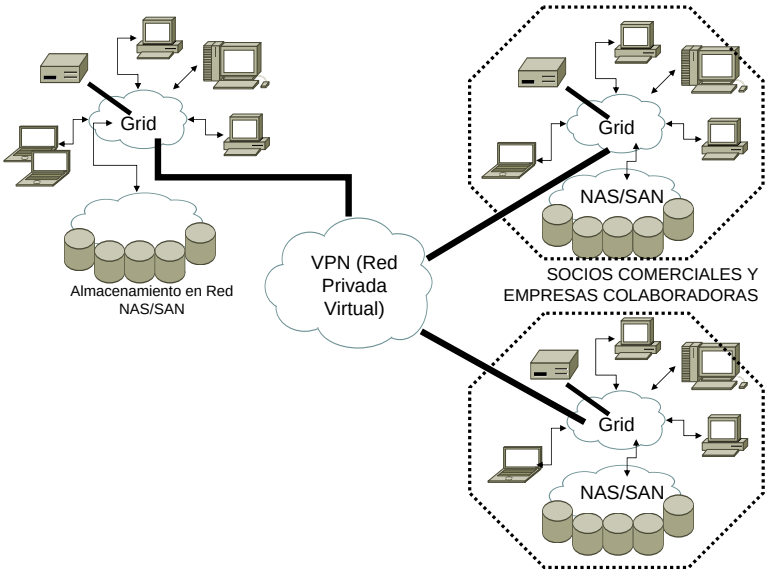


Figura 2.2: ExtraGrid. Fuente: Maqueira 2008

especiales de seguridad. Su uso principal está asociado a los experimentos con software Grid. Estos sistemas tienen muchas semejanzas con los clusters interdepartamentales, que son una evolución de los cluster departamentales que sólo están formados por recursos homogéneos. También reciben la denominación de Cluster Grid [Gentzch04], ya que no se limitan a un único domino en la red local, como los cluster interdepartamentales, sino que tratan elementos distribuidos geográficamente.

- IntraGrid. En esta configuración están disponibles muchos tipos de recursos con una mayor heterogeneidad y compartidos por medio de una red de alta velocidad. Los recursos participantes pertenecen a múltiples departamentos pero siempre de una misma organización. Estos sistemas pueden abarcar organizaciones con sedes distribuidas geográficamente. En algunos casos se utilizarán conexiones dedicadas entre las distintas delegaciones y en otros muchos casos se cambia a tecnologías basadas en Internet aumentando la importancia de la seguridad y eliminando el control centralizado.

- ExtraGrid. Estos sistemas están formados por diversas organizaciones colaboradoras que comparten recursos, creando organizaciones virtuales, figura 2.2. Una denominación equivalente a ExtraGrid es Partners Grid: desarrollos grids entre organizaciones que colaboran en proyectos comunes, siendo necesario optimizar los recursos existentes y compartirlos en busca de una meta común [Abbas04].

- InterGrid. Estos sistemas tienen unos requisitos de seguridad muy altos, ya que en un sistema InterGrid las organizaciones colaboran en proyectos y objetivos comunes sin existir

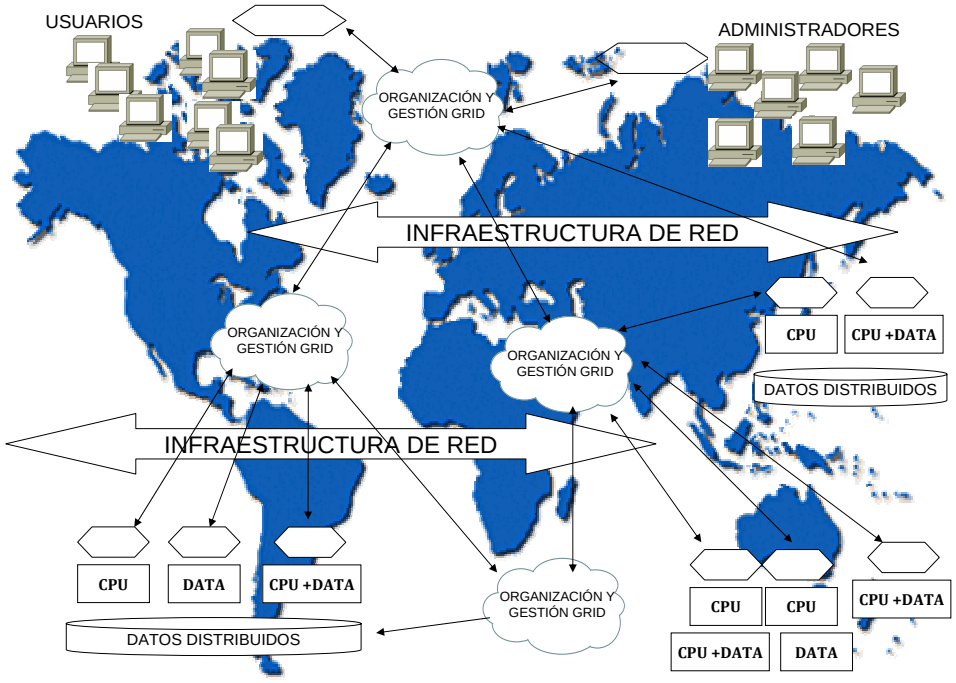


Figura 2.3: InterGrid. Fuente: Maqueira 2008

necesariamente lazos o vínculos entre ellas. Con estos sistemas se introducen posibilidades de venta y uso de recursos para un amplio grupo de posibles consumidores que utilizan los recursos como un servicio. El objetivo puede ser una gran red Grid de nivel mundial. La figura 2.3 recoge, de forma gráfica, la idea de este tipo de Grid.

2.4.2. Tipología según el ámbito de los recursos participantes

En esta clasificación la diferencia viene dada por la procedencia de los recursos que forman el sistema Grid. Si las organizaciones que aportan recursos al Grid los comparten en un ámbito restringido o privado, o si son aportados por usuarios anónimos que ofrecen sus recursos cuando no los están utilizando. Según este criterio es posible distinguir entre Grid Público (Public Grid), Grid Privado (Private Grid) [Hwang07] [Taylor07] y Grid Híbrido.

- Grid Público (Public Grid). Este tipo de sistema contiene la computación voluntaria, ya que en este sistema los recursos provienen de diversas fuentes independientes entre sí, siendo lo habitual que procedan de usuarios particulares que aportan los recursos de sus ordenadores personales cuando no los están utilizando. En este Grid se identifican tres tipos de individuos [Taylor07]: clientes, operadores y participantes. Los clientes son aquellos que utilizan este sistema para realizar una determinada tarea, ya que tienen grandes necesida-

des de cómputo. Los participantes son individuos que ofrecen y aceptan que sus recursos sean utilizados para llevar acabo las tareas de los clientes. El operador es el encargado de la gestión del sistema, básicamente, realiza el trabajo necesario para conectar clientes y participantes. En los últimos años han surgido una gran cantidad de proyectos de este tipo y, en algunos casos, una misma infraestructura Grid puede ser utilizada para varios proyectos. Algunos autores [Plaszczak05] hacen referencia a un Grid abierto (Open Grid) en contraposición a los Grid de ámbito empresarial (Enterprise Grid, Partners Grid). Un Open Grid se caracteriza por utilizar estándares abiertos y por utilizar recursos libres, es decir los aportados por voluntarios. Los Grid públicos pueden ser considerados como un caso especial de un Open Grid, donde existe un gran desarrollo Grid en el que individuos y organizaciones pueden participar de forma libre y aportando recursos no utilizados.

- Grid Privado (Private Grid). A diferencia del tipo anterior, este tipo de Grid es cerrado y se centra en el ámbito de una sóla organización o grupo de organizaciones que forman una Organización Virtual. En este tipo de Grid, los recursos son aportados por la organización u organizaciones formantes entre las que existen confianza o algún tipo de relación empresarial. Este grupo engloba a InfraGrid, IntraGrid y ExtraGrid.

- Grid híbrido. Este sistema está formado por recursos aportados por la organización u organizaciones participantes y, además, por recursos aportados por usuarios independientes de éstas, que con su aportación agregan una capacidad mayor. También se pueden agregar recursos de un Open Grid creando un mercado de recursos como el expuesto en [Altmann08]. En este tipo de estructuras, organizaciones con Grid Privados, pueden acudir a comprar recursos adicionales en los momentos que los necesiten para satisfacer altas demandas. De la misma manera, pueden existir usuarios y organizaciones dispuestos a donar o vender sus recursos ociosos a determinadas organizaciones cuyos objetivos les resulten interesantes.

2.4.3. Tipología según su finalidad

Según [Xhafa10b], atendiendo a la finalidad del sistema, esta tipología tiene en cuenta las diferentes arquitecturas que dan solución a diferentes problemas existentes, distinguiendo entre Grid Computacional (Computational Grid), Grid de Barrido (Scavenging Grids), Grid científicos (eScience Grids), Grid de Datos (Data Grid) y Grid Empresarial (Business Grid). Dentro de los sistemas Business Grid se encuentran los sistemas Desktop Grid, que serán expuestos con detalle, ya que son los sistemas Grid más cercanos al ámbito de esta tesis.

- Computational Grids

  Una de las primeras cuestiones que fue planteada durante los comienzos de los sistemas Grid fue la necesidad de disponer grid computacionales. Por un lado, las propuestas computacionales han mostrado un gran éxito en cualquier campo relacionado con la actividad humana. Guiado por el incremento de la complejidad de los problemas en la vida real, y ayudado por

el incremento de la capacidad de las tecnologías, la actividad humana (meteorología, ingeniería, negocios, personal, etc.) está altamente basada en computación. Los ordenadores son usados para modelar y simular problemas complejos [Deelman05], para diagnósticos médicos [Shah04], bioinformática [Sun07], pronóstico de tiempo [Keith08], etc. Aún así, existen muchos problemas que superan nuestra habilidad para resolverlos, normalmente porque requieren procesar gran cantidad de operaciones o datos. A pesar de que la capacidad de los ordenadores sigue en aumento, los recursos computacionales no responden a la continua demanda de más poder computacional.

Por otro lado, estudios estadísticos demuestran que los ordenadores son normalmente infrautilizados [Buyya02]. La mayor parte de los ordenadores de las empresas, administraciones, etc. pasan la mayoría de su tiempo ociosos o usados para tareas básicas que no requieren el total de la potencia computacional de la que disponen. Es señalado en [Buyya05] que una cantidad considerable de dinero es gastada en la adquisición de estos recursos. Uno de los principales objetivos de la tecnología Grid es, por tanto, beneficiarse de la existencia de muchos recursos de computación para utilizarlos de forma compartida con sus propietarios.

- Scavenging Grids

  Las políticas de scavenging, o de barrido, implican que cada vez que un recurso permanece inactivo, este debe informar de su estado al nodo responsable de la gestión y planificación de recursos [Kane10]. Entonces, este nodo principal asigna al nuevo recurso ocioso la siguiente tarea pendiente que puede ser ejecutada en esta máquina. Normalmente esta forma de computación Grid dificulta la ejecución de la aplicación, ya que en el caso de que el recurso ocioso cambie su estado a ocupado, ésta será suspendida o retrasada. Dicha situación puede hacer que los tiempos de finalización de las tareas no sean predecibles [Sonmez09]. Con el objetivo de tener un comportamiento predecible, existen recursos que son dedicados en exclusiva al grid. Además, esto permite que las herramientas asociadas a los planificadores calculen, de forma aproximada, el tiempo de finalización para un conjunto de tareas, cuando sus características son conocidas a priori. Un ejemplo de scavenging grids es el proyecto Seti@home.

- eScience Grids

  Bajo el nombre de eScience Grid se encuentran varios tipos de Grids, que están principalmente dedicados a la solución de problemas de ciencias e ingeniería. Dichos Grids dan soporte a una infraestructura computacional (acceso a recursos computacionales y de datos) necesaria para resolver muchos problemas complejos derivados de áreas de la ciencia e ingeniería. Ejemplos representativos son UK eScience Grid [Hey02], German D-Grid [Gentzsch06], french Grid'5000 [Bolze06] y China's eScience [Zhuge05] por nombrar unos pocos.

- Data Grids

Data Grids son sistemas de Grid que, principalmente, tratan con repositorios de datos para el intercambio, acceso y administración de gran cantidad de datos distribuidos. Muchas aplicaciones científicas y de ingeniería requieren acceder a gran cantidad de datos distribuidos, sin embargo, estos datos pueden tener su propio formato. Una aplicación necesita acceder a datos de diferentes fuentes de manera segura y transparente. En tales sistemas Grids, muchos tipos de algoritmos, como los de replicación [Tang06], son importantes para incrementar el rendimiento de las aplicaciones Grid en el uso de gran cantidad de datos. También, el movimiento de datos es un problema para obtener un gran rendimiento.

- Business Grids

  Aunque las tecnologías Grid han sido (y son) usadas para muchos proyectos científicos, hoy en día Grid Computing ha llegado a ser un importante componente de los entornos de negocios. En realidad, actualmente los negocios deben ser capaces de responder a la incipiente demanda de los consumidores y ajustarse dinámicamente y de manera eficiente a los cambios de mercado y a las nuevas peticiones de los consumidores. Los sistemas de Business Grids [Mietzner09] hacen posible ejecutar varios proyectos de muchos departamentos compartiendo recursos (datos o capacidad de cómputo) de una forma transparente. En estos sistemas Grid, la seguridad y las políticas de administración de recursos no son las primeras preocupaciones. En realidad Grid Computing es un importante factor que permite el incremento de la productividad en los negocios de todo el mundo. El Grid ofrece un gran potencial para resolver problemas de negocio facilitando el acceso global a servicios de computación y de datos.

2.4.3.1. Desktop Grids

Una forma de Business Grid ha emergido en los últimos años, Desktop Grid, la cual usa los ciclos ociosos de PC's de escritorio [Kacsuk07]. Pequeñas empresas e instituciones están equipadas por cientos o miles de PC's de escritorio usados para tareas de oficina. En Desktop Grid, los recursos comunes son utilizados para lograr un alto rendimiento en la ejecución de un conjunto de tareas. Esta cantidad de PC's es un buen recurso para establecer un sistema Grid para la organización. En este caso, la particularidad del Grid es su único dominio, lo que hace más fácil administrar la baja heterogeneidad y la volatilidad de los recursos. Por supuesto, Desktop Grid permite usar diversos dominios, en cuyo caso la heterogeneidad y la volatilidad de los recursos será similar a la de un sistema Grid.

Los Desktop Grids (DGs) evolucionan en dos direcciones principales: *institucional o local* y *computación voluntaria*. La primera, normalmente llamada *Computación de escritorio*, es referida a una infraestructura Grid que se limita, dentro de la frontera institucional donde la capacidad de procesamiento es utilizada, para apoyar la ejecución de la aplicaciones de empresa. La participación de los usuarios en Grid de este tipo no suele ser voluntario y se rige por la política de empresa. Aplicaciones como CONDOR [Thain05], Platform LSF [Lumb04], GridMP [Hanada05] o Virtual EZ Grid [Belgacem12] son ejemplos de este tipo.

En la computación voluntaria existe un acuerdo por el que los voluntarios ofrecen recursos informáticos a los proyectos que utilizan estos recursos para la computación distribuida y/o almacenamiento. Los voluntarios suelen ser miembros del público en general que tienen PC's propios conectados a Internet. Organizaciones, como Universidades y empresas pueden también compartir de forma voluntaria sus ordenadores. Los proyectos son típicamente académicos (universitarios) y de investigación científica. Varios de los aspectos en la relación entre proyectos y voluntarios son:

- Los voluntarios son anónimos, aunque se les puede pedir registrarse, suministrar la dirección de correo electrónico o cualquier otra información, no hay forma de vincular un proyecto a una identidad del mundo real.
- Debido a su anonimato, los voluntarios no son responsables de los proyectos. Si un voluntario realiza algo mal, de alguna manera (por ejemplo, intencionalmente devolver incorrectamente resultados computacionales) el proyecto no puede procesar o disciplinar al voluntario.
- Los voluntarios deben confiar en los proyectos en varias razones: 1) el voluntario confía en el proyecto para recibir aplicaciones que no dañan su ordenador o invadan su privacidad, 2) el voluntario confía en que el proyecto es veraz acerca de sobre lo que está trabajando, y cómo será utilizada la propiedad intelectual resultante, 3) el voluntario confía en el proyecto para seguir las prácticas de seguridad adecuadas, por lo que los hackers no pueden utilizar el proyecto como un vehículo para actividades maliciosas.

El primer proyecto de computación voluntaria fue GIMPS [Woltman07]. Otros proyectos iniciales incluyen SETI@home y Folding@home. Desktop Grid se diferencia de computación voluntaria en varios puntos:

- Se puede confiar en los recursos de computación; por ejemplo, se puede suponer que los recursos no devuelven resultados incorrectos, ya sea intencionalmente o debido a mal funcionamiento del hardware. De ahí que, por lo general, no es necesario el cálculo redundante.
- El cálculo es completamente invisible y se encuentra fuera del control de los usuarios.
- El despliegue de clientes suele ser automático; por lo tanto, no es necesario realizar una búsqueda de usuarios voluntarios.

Desktop Grids Vs Grid

Los sistemas de Desktop Grid han incrementado recientemente su atracción para la ejecución de aplicaciones de alto rendimiento debido a que la capacidad cómputo, el almacenamiento y las redes han mejorado y han llegado a ser más asequibles desde el punto de vista económico. Como se observa en la tabla 2.1, Desktop Grid es diferente de Grid en términos de tipos y características de los recursos y las formas de compartirlos. Principalmente los recursos de Desktop Grid

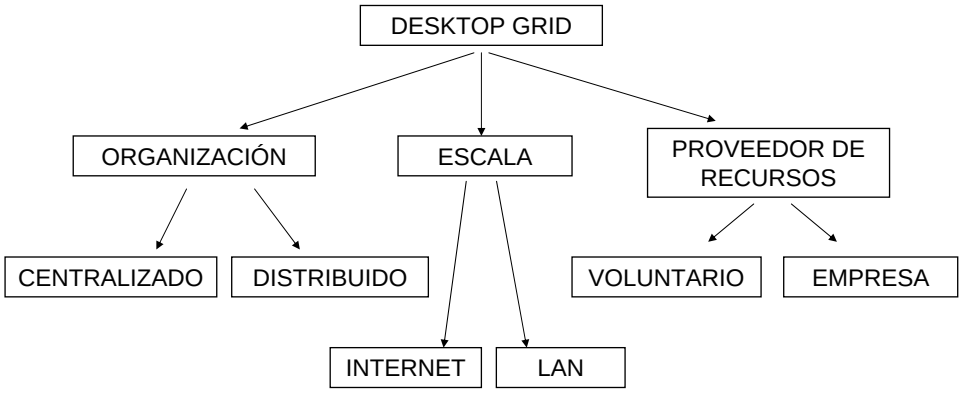


Figura 2.4: Tipología Desktop Grid

son ordenadores personales, esto es, ordenadores de escritorio, mientras que los recursos Grid incluyen superordenadores, cluster, instrumentos científicos, bases de datos, etc. Los recursos de Desktop Grid son muy volátiles, no dedicados y altamente heterogéneos. También en un sistema Desktop Grid voluntario puede existir recursos maliciosos, poco fiables y deficientes. En un Desktop Grid cada recurso es administrado por un usuario individual, mientras que los recursos Grid son gestionados por administradores profesionales. Las aplicaciones de Desktop Grid no tienen ninguna dependencia entre tareas. Sin embargo, Grid incluye aplicaciones dependientes. Un Desktop Grid trata de lograr un alto rendimiento del sistema, es decir, la cantidad de trabajo que los ordenadores de escritorio pueden hacer en un determinado período de tiempo, mientras que Grid se centra, principalmente, en el alto rendimiento, es decir, la velocidad de ejecución de un conjunto de tareas.

Tipología de Desktop Grids

Los sistemas Desktop Grid se pueden clasificar, figura 2.4, según su organización, escala y proveedor de recursos.

- **Organización**. Según la organización de sus componentes los sistemas Desktop Grid se clasifican en sistemas centralizados y distribuidos:

  Desktop Grid centralizado. Los sistemas DG centralizados se componen de tres elementos: clientes, voluntarios y servidor. El modelo de ejecución de un DG centralizado contiene siete fases: registro, envío de trabajos, asignación de tareas, ejecución de tareas, retorno de resultados de tareas, certificación de resultados y retorno de resultados del trabajo.

  1. Fase de registro. Los recursos voluntarios registran su información en el servidor.
  2. Fase de envío de trabajos. Un cliente envía un trabajo al servidor.

	Desktop Grid (DG)		Grid
	Basado en Internet (DG Voluntario)	Basado en LAN (DG Empresa)	
Recurso	De escritorio. Voluntario anónimo	De escritorio. Dentro de una organización, universidad, etc	Superordenadores, cluster, instrumentación científica, bases de datos, almacenamiento. Organizaciones Virtuales
Conexión	No dedicada. Pobre ancho de banda. Considerar Firewalls, NAT, direcciones dinámicas, etc.	No dedicada. Conexión más constante que en computación voluntaria	Dedicada. Alta velocidad y ancho de banda
Heterogeneidad	Alta	Media. Menos heterogeneidad que en DG voluntario	Baja
Dedicación	No dedicado. Muy vólatil.Necesarios mecanismos de incentivo.	No dedicado. Volatilidad Baja	Dedicado. Capaz de ser reservado.
Confianza	Voluntarios maliciosos. Es necesario certificar resultados.	Baja confianza en proveedor de recursos	Alta confianza en proveedor de recursos
Fallos	Poco fiables	Poco fiable. Más fiable que en DG voluntario.	Más fiable que DG
Administración	Administración individual. Difícil de administrar.	Administración individual. Más fácil de administrar que DG voluntario	Administración basada en dominio. Administrador profesional.
Aplicación (Trabajo)	Independiente. Computación intensiva. De alto rendimiento.	Independiente. Computación intensiva. Datos intensivos (posible). De alto rendimiento.	Independiente o Dependiente. Computación o datos intensiva. De alto rendimiento.

Tabla 2.1: Comparación Desktop Grids y Grid

3. Fase de asignación de tareas. El servidor distribuye las tareas a los voluntarios registrados por medio de un mecanismo de planificación.
4. Fase de ejecución. Cada voluntario ejecuta su tarea.
5. Fase de retorno de resultados. Cada voluntario envía los resultados de su tarea al servidor.
6. Fase de certificación de resultados. El servidor comprueba la exactitud de los resultados recibidos con el fin de soportar los voluntarios maliciosos [Choi05] o para hacer frente a las variaciones en el procesamiento numérico debido a la variedad de hardware y software [Taufer05].
7. Fase de retorno de resultados de trabajo. El servidor envía los resultados finales al cliente.

Ejemplos típicos son BOINC [Anderson04], XtremWeb [Cappello05] o Entropia [Chien03].

Desktop Grid distribuido. Los sistemas DG distribuidos se componen de clientes y voluntarios. En contraste con los sistemas Desktop Grid centralizados, no existe servidor por lo que los voluntarios tienen información de otros voluntarios. Los voluntarios son responsables de construir la red de cómputo (CON) [Subramanian05]. Está red está formada por un conjunto de voluntarios que ejecutan tareas. La planificación es llevada a cabo en cada voluntario de manera distribuida, según CON, es decir, los voluntarios distribuyen las tareas a otros voluntarios de manera diferente de acuerdo a una característica o topología de la red CON (por ejemplo, árbol, tabla hash distribuida, etc).

El modelo de ejecución de un Desktop Grid distribuido se compone de siete fases: registro, envío de trabajos, construcción de CON, asignación de tareas, ejecución de tareas, retorno de resultados de tareas y retorno de resultados del trabajo.

1. Fase de registro. Los voluntarios intercambian información entre otros voluntarios.
2. Fase de envío de trabajos. Un cliente confecciona un trabajo y lo envía a sus vecinos voluntarios.
3. Fase de construcción de CON. Los voluntarios auto-organizan su CON, según capacidad, tiempo de registro, zona horaria, o al azar de forma distribuida.
4. Fase de asignación de tareas. Los voluntarios envían tareas a sus vecinos o los voluntarios apropiados por medio de un mecanismo de planificación distribuido.
5. Fase de ejecución. Cada voluntario ejecuta su tarea.
6. Fase de retorno de resultados. Cada voluntario envía los resultados de su tarea al voluntario correspondiente.
7. Fase de retorno de resultados de trabajo. Los voluntarios envían los resultados finales al cliente.

Ejemplos típicos de sistemas Desktop Grid distribuidos son Javelin [Neary05] y Computer Power Market (CPM) [TienPing04].

- **Escala**. Según su escala, un Desktop Grid está basado en Internet o en una red LAN. Internet Desktop Grid está basado en voluntarios anónimos. Para ello se debe tener en cuenta firewall, NAT, dirección dinámica, ancho de banda pobre y conexión fiable. Por otro lado, Desktop Grid basado en LAN está formado por recursos en una empresa, universidad o institución. Este caso, existe una conectividad mayor y constante que el sistema basado en Internet.

- **Proveedor de Recursos**. De acuerdo al proveedor de recursos, un sistema Desktop Grid está constituido por recursos ofrecidos de forma voluntaria por sus participantes (DG voluntario) o recursos no voluntarios pertenecientes a una empresa o institución (DG empresarial). DG voluntarios están basados, sobre todo, en sistemas Desktop Grid en Internet. Los sistemas Desktop Grid empresariales están construidos sobre redes LAN. Los sistemas DG voluntarios son más volátiles y deficientes que un sistema Desktop Grid empresarial, que es más controlable porque los recursos se encuentran en el mismo dominio administrativo. Ejemplos típicos de DG voluntarios son BOINC, XtremWeb y Javelin. Sistemas DG empresa son Entropia y Condor.

Requisitos para sistemas Desktop Grids

Los sistemas Desktop Grids agregan un gran número de máquinas y recursos. Estos sistemas permiten que los dispositivos de escritorio se incorporen en una red amplia sin grandes esfuerzos. Los sistemas DG contienen una colección de recursos de computación heterogéneos, distribuidos a lo largo una red corporativa y sujetos a la gestión de varios usos y regímenes de utilización y agregados en un único recurso fácilmente manejable y utilizable. Además, un sistema Desktop Grid debe hacer esto de una manera que asegure que el impacto sea indetectable sobre el uso de los recursos informáticos con otros propósitos. Para los usuarios finales de computación distribuida y sistemas Grid, los recursos agregados deben ser usados de una manera simple. Los requerimientos clave según [Foster04a], [Domingues07], [Cao12], se muestran en la tabla 2.2:

2.5. Planificación en Sistemas Grid

La planificación de tareas en un sistema Grid puede ser vista como una familia completa de problemas. Esto es debido a la gran cantidad de parámetros que intervienen en la planificación. A continuación son mostrados varios conceptos básicos de la planificación en sistemas Grid e identificados los tipos de planificadores más comunes. Desde el punto de vista computacional,

Requisito	**Breve Descripción**
Eficiencia	Un DG debe cosechar prácticamente todos los recursos ociosos.
Robustez	Las tareas deben completarse con el rendimiento previsible, enmascarando los fallos subyacentes de los recursos. Los sistemas DG deben tolerar fallos en trabajos, máquinas y en la red e incluir una variedad de mecanismos para asegurar la finalización oportuna de un trabajo con la presencia de tales fallos.
Seguridad	El sistema debe proteger la alteración o divulgación de datos. Además, el DG debe proteger la integridad de los equipos de escritorio, prevenir el acceso y la modificación de datos y aplicaciones de los recursos de escritorio.
Escalable	Un DG debe escalar miles de PCs de escritorio desplegados en las redes de la empresa. Los sistemas deben escalar en varias direcciones, obteniendo buenos resultados con un esfuerzo razonable en una variedad de escalas.
Manejable	Con miles de recursos de computación, la gestión y esfuerzo de la administración en un DG no se puede ampliar con el número de recursos. Estos sistemas deben lograr una capacidad de gestión que no requiera incrementar el esfuerzo humano de los clientes que se agregan al sistema.
Discreto	Los sistemas DGs comparten recursos de computación, almacenamiento o redes con otros usos en la empresa. El uso de estos recursos debe hacerse de una forma discreta, para no interferir con el uso principal de sus propietarios.
Facilidad para integrar aplicaciones	Un sistema Desktop Grid es una plataforma que soporta aplicaciones que a su vez proporcionan valor a los usuarios finales. Los sistemas de computación distribuida deben ser compatibles con las aplicaciones desarrolladas en varios lenguajes de programación, modelos y herramientas todo con un mínimo de esfuerzo de desarrollo.
Participación en múltiples proyectos	La razón fundamental para la participación en varios proyectos reside en el hecho de que muchos proyectos tienen tiempo de inactividad (por mantenimiento, reparación de la infraestructura, etc), o escasez de tareas.

Tabla 2.2: Requisitos para Desktop Grids

la planificación de tareas en sus diferentes formas es muy costosa. El problema de encontrar la planificación óptima en sistemas heterogéneos es en general un problema de tipo NP [Garey90].

2.5.1. Conceptos básicos y terminología

Muchos tipos de recursos pueden ser compartidos y usados en un sistema Grid. Normalmente éstos son accedidos a través de una aplicación que se ejecuta en el Grid. Una aplicación es usada para definir el trabajo de más alto nivel en el Grid. Un escenario típico es el siguiente: una aplicación genera varios trabajos, que pueden estar compuestos de subtareas, con el fin de ser resueltas. El sistema grid es responsable de enviar cada subtarea a un recurso para ser ejecutada en este. En un escenario grid simple es el usuario el que selecciona la máquina más adecuada para ejecutar su subtarea. Sin embargo, en general, los sistemas Grid disponen de un planificador, que automáticamente y de forma eficiente, encuentra las máquinas más apropiadas para ejecutar varios trabajos.

2.5.1.1. Nuevas características en la planificación Grid

El problema de planificación de tareas no es un problema aparecido con los sistemas Grid. De hecho, es uno de los problemas más estudiados en investigación. Sin embargo, en el escenario grid hay varias características que hacen este problema diferente de la versión tradicional de sistemas distribuidos. Algunas de esas características son las siguientes:

1. La estructura dinámica del sistema Grid. A diferencia de la computación distribuida tradicional, como los clusters, los recursos en el Grid pueden unirse o dejar el sistema de una manera impredecible [Foster04b]. Puede ser debido a una simple pérdida de conexión al sistema o porque sus propietarios apagan la máquina o cambian el sistema operativo, etc. Dado que los recursos pertenecen a diferentes dominios, no es posible ejercer algún control sobre los ellos.

2. La alta heterogeneidad de los recursos. Los sistemas Grid actúan como un gran superordenador virtual, por lo que, los recursos que lo forman pueden ser muy dispares [Yu08], que van desde laptops, Pc's de sobremesa, clusters, superordenadores o pequeños dispositivos con limitados recursos computacionales.

3. La alta heterogeneidad de los trabajos. Los trabajos llegan a cualquier sistema Grid de muy diversas y heterogéneas formas en términos de necesidades de cómputo. Por ejemplo, pueden ser de computación intensiva o de manejo de datos; algunos trabajos pueden ser aplicaciones con un gran rango de especificaciones, otras pueden ser solo una tarea atómica. En gran medida, un sistema Grid no puede ser consciente del tipo de tareas, trabajos o aplicaciones que van a llegar al sistema.

4. La alta heterogeneidad de las redes de interconexión. Los recursos de un Grid pueden estar conectados a través de Internet usando diferentes redes de conexión. Los costes de

transmisión pueden ser muy importantes en el rendimiento del sistema y, por lo tanto, son necesarias formas inteligentes de hacer frente a la heterogeneidad de las conexiones.

5. La existencia de planificadores locales. Los sistemas Grid son construidos gracias a la participación de diversas instituciones, universidades, empresas e individuos. La mayoría de estos recursos pueden ser usados en redes locales con sus propios planificadores locales. En estos casos un posible requisito es usar el planificador local del dominio en vez de uno externo.

6. La existencia de políticas locales en los recursos. No es posible asumir un control total sobre los recursos debido a los diversos propietarios de estos. Las empresas pueden tener inesperadas necesidades computacionales y pueden decidir reducir su contribución al Grid. Otras políticas, como derechos de acceso, capacidad de almacenamiento disponible, pago por uso, etc. también deben ser tenidas en cuenta.

7. Gran escala del sistema Grid. Se espera que estos sistemas sean enormes uniendo cientos o miles de nodos a través del mundo. Además, los trabajos, tareas o aplicaciones enviadas al Grid pueden ser grandes en número debido a que diferentes usuarios y/o aplicaciones pueden enviar sus trabajos al Grid sin conocer previamente el estado de carga del sistema. Por lo tanto, una administración eficiente de los recursos requiere el uso de diferentes tipos de planificadores (meta-planificadores, descentralizados, locales, etc.) y sus posibles combinaciones jerárquicas.

8. Seguridad. Esta característica, inexistente en la planificación clásica, es un factor importante en Grid. La seguridad tiene dos objetivos: de una parte, una tarea, trabajo o aplicación debe ser alojada en un nodo seguro, con unos determinados requisitos de seguridad, esto es, el nodo no debe "ver" o acceder al proceso o a los datos usados por la tarea, trabajo o aplicación. Por otra parte, el nodo debe tener requisitos de seguridad, esto es, la tarea, trabajo o aplicación que está siendo ejecutada en el recurso no puede "ver" o acceder a otros datos existentes en el nodo.

2.5.1.2. Definiciones y terminología

Una definición precisa de un planificador Grid depende en gran medida de cómo se encuentra organizado el planificador y las características del entorno. En una configuración genérica, un planificador Grid debe estar en permanente ejecución: recibiendo nuevos trabajos entrantes, comprobando recursos disponibles, seleccionando los recursos apropiados según los requisitos de los trabajos y criterios de comportamiento, y produciendo planes de trabajo para los recursos existentes.

La siguiente terminología es usada, normalmente, en el ámbito de los planificadores Grid:

Tarea: Representa una unidad computacional (un programa y unos datos asociados) que se ejecuta en un nodo Grid. Aunque en la literatura no hay una definición exacta, habitualmente, una tarea es considerada la unidad planificada mínima e indivisible.

Trabajo: Un trabajo (Job) es una actividad computacional formada por varias tareas que pueden requerir diferentes capacidades de cómputo y tener distintos requisitos en cuanto a recursos (CPU, número de nodos, memoria, librerías software,etc.). En el caso más simple un trabajo puede ser solo una tarea.

Aplicación: Una aplicación es un software encargado de resolver un problema en una infraestructura computacional.

Recurso: Un recurso es una entidad básica de cómputo donde las tareas, trabajos y aplicaciones son planificadas, alojadas y procesadas correctamente. Los recursos tienen sus propias características como CPU, memoria, software, etc. Otros parámetros asociados a los recursos y que cambian constantemente son la capacidad de cómputo y la carga de trabajo.

Planificadores: Componentes software encargados de alojar tareas, trabajos o aplicaciones en recursos del Grid bajo múltiples criterios y configuraciones. Existen diferentes niveles de planificadores: meta y local. Como componente principal de un sistema Grid, el planificador interactúa con el resto de elementos del sistema. ,

2.5.2. Fases en Planificación Grid

Con el fin de realizar el proceso de planificación, el planificador Grid debe seguir una serie de pasos clasificados en cuatro bloques:

1. Preparación y acopio de información: El planificador debe tener acceso a la información de los recursos disponibles, tareas y trabajos o aplicaciones. Esta información es crucial para que el planificador lleve a cabo su trabajo.

2. Selección del recurso: No todos los recursos pueden ser candidatos a alojar tareas, trabajos o aplicaciones, por lo tanto, el proceso de selección está basado en los requisitos de las tareas y las características de los recursos. Este proceso de selección depende del modo de planificación. También forma parte de este bloque la reserva del recurso.

3. Asignación de tareas: En esta fase, la planificación se hace efectiva: las tareas o trabajos son asignados al recurso seleccionado.

4. Ejecución y monitorización de la tarea: Los recursos comienzan la ejecución de las tareas asignadas. La monitorización informa del progreso en la ejecución, así como los posibles fallos.

2.5.3. Tipos de planificadores

2.5.3.1. Clasificación según su funcionamiento

Mencionado anteriormente el problema de planificación en entornos Grid, introduce nuevos retos debido a las características de estos sistemas. En este apartado se realiza una clasificación, según su diseño, de los diferentes tipos de algoritmos de planificación que pueden ser usados en entornos Grid.

Estáticos - Dinámicos

En caso de algoritmos estáticos la información acerca de los recursos y de las tareas es asumida "a priori". Por el contrario, en el caso de algoritmos dinámicos, la idea reside en estudiar el estado del sistema Grid para planificar y alojar una tarea en el mejor recurso disponible.

- Algoritmos estáticos. En modo estático la asignación de una tarea a un nodo o recurso es estática, por lo tanto, se puede realizar una estimación previa del coste de computación de la ejecución actual. Uno de los mayores beneficios de los algoritmos estáticos es su facilidad de implementación. Esta facilidad contrasta con la dificultad para adaptarse a situaciones no esperadas, como pueden ser que un recurso o nodo falle, que se quede aislado en la red y no sea accesible, etc. Desafortunadamente, estas situaciones son bastante posibles, por lo que, para mejorar el funcionamiento de estos algoritmos se han introducido mecanismos de re-planificación [Cooper04].

- Algoritmos dinámicos. Los algoritmos dinámicos son los ideales para planificar una serie de tareas que llegan escalonadamente al planificador. Estos algoritmos antes de asignar una tarea tienen en cuenta el estado de cada uno de los recursos, en el caso de un planificador local o el estado de cada nodo en el caso de un Meta-Planificador. El estado de cada nodo o recurso implica diferentes parámetros como capacidad de cómputo, carga actual, tamaño de cola de tareas, estado de la red, etc. La ventaja, de estos algoritmos frente a los estáticos reside en que se realiza una planificación "inteligente" dependiendo del estado de cada nodo o recurso, mientras que en los estáticos se realiza una planificación "a ciegas" ya que no se tiene en cuenta el estado de cada nodo o recurso a la hora de asignar tareas.

Local - Global

La función principal de un algoritmo de planificación Global (Meta-Planificador) es obtener la información del estado general de un nodo Grid proporcionada por los planificadores locales de cada nodo y decidir de entre estos nodos cuál es el óptimo para que ejecute una tarea. En un algoritmo de planificación local para tomar la decisión de a qué recurso enviar una tarea solo se tiene en cuenta los recursos locales del nodo grid.

On line - On batch

En modo on-line, o modo inmediato, una tarea es asignada a un recurso en cuanto es recibida por el planificador, por lo que no tiene que esperar para ser ejecutada. En este modo una tarea es considerada solo una vez para su asignación. Este modo funciona bien cuando la tasa de llegada de tareas es baja.

En modo on batch las tareas no son asignadas a los recursos inmediatamente, sino que, son recogidas en una bolsa de tareas y son asignadas cada cierto tiempo. Esto permite tomar mejores decisiones obteniendo completa información del estado del sistema, permitiendo retrasar o adelantar el siguiente evento de envío de tareas. Si el tiempo entre llegadas de tareas es alto este modo funciona mejor que el anterior.

Criterio Simple - Criterio Múltiple

Planificar tareas en un sistema Grid es un problema complejo de optimización que requiere considerar varios criterios. Sin embargo, en general, no es posible obtener una optimización para estos criterios puesto que estos suelen ser totalmente opuestos. Algunos de estos criterios de optimización son el tiempo de ejecución de un conjunto de tareas, distribución de la utilización de recursos, criterios económicos, etc.

Lo usual es desarrollar algoritmos que optimicen estos criterios de forma individual o de forma conjunta, normalmente usando dos de estos criterios, algoritmos bi-criterio [Wieczorek08]. Existen otras soluciones que proponen optimizar una combinación lineal de múltiples criterios con diferentes pesos asignados a cada uno de ellos. Estas aproximaciones que intentan optimizar una combinación lineal de múltiples criterios asumen que el usuario es capaz de especificar sus requerimientos en este modelo, cosa que no siempre se cumple.

De entre los algoritmos que consideran más de un criterio, los que eligen optimizar el tiempo de ejecución y el coste económico [Sakellariou07] [Yu06a] son los más usuales; mientras que existen otros que optimizan el tiempo de ejecución y fiabilidad [Dabrowski09]. También existen planificadores que optimizan la utilización de recursos sin interferir en el tiempo de ejecución [Chang09] [Li09].

Distribuido - Centralizado

En escenarios dinámicos de planificación la responsabilidad de planificar tareas puede estar centralizada en un planificador o bien compartida de forma distribuida entre múltiples planificadores. En un entorno Grid puede haber muchas aplicaciones que requieren ser ejecutadas simultáneamente o re-planificadas. La estrategia centralizada tiene la ventaja de la facilidad de implementación, pero tiene un mayor sufrimiento con la escalabilidad, tolerancia a fallos, pudiéndose convertir en un "cuello de botella". La gran mayoría de los Meta-Planificadores existentes son centralizados [Christodoulopoulos09].

Un planificador completamente descentralizado y dinámico fue propuesto en [Huang08]. Una propiedad de este algoritmo es que usa una estrategia de búsqueda para encontrar recursos a los que una tarea puede migrar.

Cooperativo - No cooperativo

Si es elegido un planificador distribuido, el próximo paso es considerar si los nodos de cada planificador van a trabajar de forma cooperativa o independientes (no cooperativa). En el caso de no cooperar los planificadores actúan como entidades autónomas y toman las decisiones siguiendo sus propios objetivos independientemente de los efectos de esta decisión en el resto del sistema.

En el caso de planificadores cooperativos [Subrata10], cada uno tiene la responsabilidad de planificar su propia porción de tareas, pero todos los planificadores trabajan de forma conjunta hacia un objetivo global y común. Cada planificador tiene su propia política local, pero las decisiones tomadas son consensuadas con el resto de planificadores para alcanzar este objetivo global en vez de tomar decisiones que solo afectan al comportamiento local de una tarea en particular.

2.5.3.2. Clasificación según funciones objetivo

Un sistema Grid está formado por dos partes principales, por un lado los consumidores de recursos, que ejecutan sus tareas en los recursos de un Grid y, por otra, los proveedores de estos recursos. Normalmente estas dos partes tienen distintas motivaciones cuando se unen al Grid. Estas motivaciones están representadas en una función objetivo. La mayor parte de los objetivos son heredados de los sistemas distribuidos tradicionales. Los usuarios Grid están básicamente centrados en el rendimiento de sus aplicaciones, por ejemplo, el coste total de ejecutar una aplicación en particular, mientras que los proveedores de recursos están más centrados en la utilización de un determinado recurso durante un período. Así, estas funciones objetivo pueden clasificarse en dos categorías: centradas en la aplicación y centradas en los recursos.

Centradas en la aplicación

Los algoritmos de planificación centrados en la aplicación tienen como objetivo optimizar el comportamiento de cada aplicación individual perteneciente a un usuario. La mayoría de las aplicaciones de Grid se ocupan del tiempo, por ejemplo el makespan (diferencia de tiempo entre el inicio de un trabajo y su finalización), que es el tiempo que pasa desde el inicio de la primera tarea de un trabajo hasta la finalización de la última tarea de ese trabajo. Makespan es una de las medidas más populares para minimizar en los algoritmos de planificación y de la que existen muchos ejemplos en la literatura [Chang09] [Tseng09].

Si las características de Grid Computing son inspeccionadas y comparadas con otros sistemas se puede observar que el Grid tiene una alta similaridad con un mercado. El mercado es también un sistema descentralizado, competitivo y dinámico donde los clientes y los proveedores de productos tienen sus propios objetivos. Basado en esta observación se introduce en Grid un nuevo modelo económico con el fin de optimizar la administración de recursos y los problemas de planificación [Kumar09]. Los componentes básicos en un mercado son los productores, consumidores y los productos, análogos a los propietarios de los recursos, los usuarios de los recursos y recursos de computación en el sistema Grid. Las teorías económicas están establecidas según

el estudio del comportamiento del mercado. Precio y calidad de los productos son parámetros que permiten tomar decisiones a los consumidores y proveedores. Por ejemplo, un consumidor normalmente quiere obtener los mejores servicios (un menor makespan) con un coste menor, si es posible, mientras que un proveedor normalmente quiere obtener una mayor utilización de sus productos (recursos) para aumentar sus beneficios. El uso de métodos económicos en Grid implica un proceso de interacción entre los proveedores de recursos y los usuarios similar al comportamiento de varios mercados como la negociación, la oferta o la subasta. En [Buyya02] varios modelos económicos que pueden ser aplicados al mundo Grid son discutidos.

Al introducirse estos modelos económicos en Grid surgen nuevas oportunidades de investigación. El coste económico y el beneficio son considerados por proveedores de recursos, nuevas funciones objetivo, por lo que, se proponen nuevos algoritmos de planificación. Estos algoritmos implementan distintas estrategias. Por ejemplo, garantizan el plazo de ejecución y minimizan el coste o garantizan un presupuesto y minimizan el tiempo de ejecución. Las dificultades para optimizar estos dos parámetros en un algoritmo residen en que las unidades de coste y de tiempo son diferentes, por lo que estos dos objetivos suelen tener conflictos [Venugopal05].

Estas ideas básicas permiten construir nuevos métodos de planificación con objetivos tradicionales. El objetivo por parte del consumidor es minimizar el makespan, sin embargo el objetivo del proveedor es maximizar la utilización de recursos [Mehta10]. Estos modelos económicos para problemas de planificación son muy interesantes debido a su similitud con la vida diaria.

Además de estas funciones simples, muchas aplicaciones usan objetivos compuestos, por ejemplo, un usuario quiere ejecutar sus tareas en un tiempo bajo y con un coste económico bajo. La primera dificultad es adaptar estos objetivos con diferentes medidas como son el tiempo y el coste. Es necesario que los algoritmos de planificación sean lo suficientemente adaptativos para cumplir su misión. El desarrollo de la infraestructura Grid ha mostrado una orientación hacia los servicios diferenciados, por lo que la calidad de servicio (QoS) puede ser un gran objetivo para muchas aplicaciones Grid. Estas situaciones hacen que los algoritmos de planificación en Grid sean mucho más complicados.

Centradas en los recursos

Estos algoritmos están diseñados con el ánimo de incrementar al máximo el rendimiento de los recursos. Estos objetivos están relacionados con la utilización de recursos, como por ejemplo, el rendimiento, que es la capacidad de un recurso de procesar un cierto número de tareas en un período dado; la utilización, que es el porcentaje de tiempo que un recurso está ocupado. Baja utilización significa que un recurso está ocioso e inutilizado. Para un multiprocesador, las diferencias de utilización a través de los procesadores describen la carga del sistema y el grado de rendimiento. En los entornos de computación Grid, debido a la autonomía de los usuarios y de los proveedores de recursos, los objetivos de ambos tipos están en contradicción.

Centradas en el flujo de trabajo

Los flujos de trabajo o jobs pueden variar con respecto a su nivel de complejidad, la semántica

de sus componentes y la dinamicidad de su modelo de ejecución. Todos estos aspectos deben ser tomados en cuenta cuando se planifican flujos de trabajo, ya que, la ejecución de un flujo de trabajo está basado en un modelo específico puede requerir una planificación específica. Por ejemplo, existen algoritmos de planificación dedicados a ejecutar grafos acíclicos dirigidos.

En la literatura un flujo de trabajo está normalmente descrito, [Kiepuszewski03], mediante cuatro perspectivas: perspectiva de control de flujo, perspectiva de datos, perspectiva de recursos y perspectiva de operaciones. La perspectiva de control de flujo describe las tareas y su orden de ejecución a través de diferentes parámetros como secuencia, elección, paralelismo y sincronización. La perspectiva de datos define el flujo de datos entre tareas. La perspectiva de recursos define la responsabilidad de los recursos para ejecutar tareas. Finalmente la perspectiva operacional describe las acciones elementales ejecutadas por las tareas.

Estos flujos de trabajo pueden tener diferentes estructuras y son diseñados para dominios específicos. El modelo de flujo de trabajo más común es el de un Grafo Acíclico Dirigido (DAG), donde existen dependencias entre los nodos que forman el flujo de trabajo que pueden estar organizados en cualquier forma pero sin crear ciclos en el grafo. Existen otros flujos que extienden este modelo permitiendo ciclos y otras construcciones como bucles paralelos o condicionales. Otros modelos están basados en una simple estructura formada por tareas independientes. Por lo tanto existen clases de flujos de trabajos:

- DAG. El flujo de trabajo es un DAG.
- Grafo extendido. La estructura del flujo de trabajo extiende un DAG pero añadiendo estructuras adicionales como bucles y condicionales.
- DAG simplificado.
  - Paralelos. El flujo de trabajo está formado por un conjunto de tareas independientes que son distribuidas a través de los diferentes recursos.
  - Secuencia. El flujo de trabajo es una aplicación secuencial.
  - Árbol. El flujo de trabajo es un grafo tipo árbol.
  - Otros. Algunos flujos no se incluyen en los apartados anteriores como puede ser, la Transformada de Fourier o el algoritmo de Strassen.

En [Subhlok00] se considera un flujo de trabajo basado en una secuencia de tareas paralelas. En [Gounaris06] el flujo está basado en una estructura de árbol desarrollado según las restricciones de una serie de reglas. El flujo considerado en [Deelman05] tiene una estructura que permite introducir la idea de dividir un flujo de tareas en una secuencia de varios subflujos.

Un flujo de trabajo está compuesto por diferentes nodos. Estos nodos (elementos atómicos de un flujo de trabajo) pueden ser añadidos o eliminados del flujo o, pueden ser agrupados formando un nuevo nodo atómico con el ánimo de incrementar el beneficio del usuario o del Grid.. Se pueden distinguir dos clases: nodos fijos, que son aquellos que forman un flujo de trabajo que no puede

ser cambiado durante el proceso de planificación o; nodos tuneables, que son aquellos que pueden ser añadidos, modificados o eliminados durante el proceso de planificación.

Las tareas de un flujo de trabajo pueden considerar una o varias entradas de datos asociadas. En caso de que el flujo de trabajo se ejecute una sola vez para un entrada simple se hablará de Entrada Simple; en el caso de que el flujo de trabajo se ejecute varias veces, cambiando los datos de entrada, se esta hablando de un flujo de trabajo tipo Pipelined.

2.5.4. Criterios de optimización en planificadores Grid

Numerosos criterios de comportamiento y optimización pueden ser considerados en planificación Grid. La planificación en sistemas Grid es un problema multi-objetivo en su formulación general. Tradicionalmente, se pueden distinguir entre criterios de comportamiento y de optimización a pesar de que es la consideración conjunta de ambos criterios la que permite la caracterización del comportamiento general del sistema Grid. Los criterios de rendimiento de un sistema Grid incluyen el uso de CPU de los recursos, balanceo de carga, el uso del sistema, tiempo de espera en cola, rendimiento, tiempo de respuesta y tiempos de espera. Además, otros criterios para mostrar el comportamiento del sistema Grid son, plazos, plazos fallidos, prioridades de usuarios, fallos en recursos, etc. Por otra parte, los criterios de optimización incluyen makespan, tiempo de flujo, tiempo de respuesta promedio ponderado, uso de recursos, disminución de velocidad, la tardanza, etc. Entre las definiciones formales más relevantes de indicadores de comportamiento se señalan los siguientes:

- Makespan [Xhafa10b]

$$makespan = \max_{j \in J} T_j \tag{2.1}$$

  donde T_j indica el tiempo de ejecución de la tarea j.

- Tiempo de flujo (Flowtime) [Xhafa10b]

$$flowtime = \sum_{j \in J} T_j \tag{2.2}$$

  o la suma de tiempos de ejecución de todas las tareas.

- Tiempo de respuesta promedio ponderado (Average Weighted Response Time AWRT) [Franke07]

$$AWRT = \frac{\sum j \in J \omega_j (T_j - R_j)}{\sum j \in J \omega_j} \tag{2.3}$$

  donde ω_j y R_j indican el peso y el tiempo de envío asociado a la tarea j, respectivamente.

- Uso del sistema (System Usage)

$$SystemUsage = \frac{N_{ac}}{\min(N_{av}, N_r)} \tag{2.4}$$

donde N_{ac} y N_{av} describen el número de CPUs activas y disponibles, respectivamente y N_r representa el numero de CPU demandadas.

- Tardanza (Tardiness) [Klusácek08]

$$Tardiness = \text{máx}\{T_j - d_j, 0\} \tag{2.5}$$

donde d_j indica el tiempo de finalización para la tarea j.

- Disminución de velocidad (Slowdown) [Ernemann04]

$$Slowdown = \frac{T_j - R_j}{p_j} \tag{2.6}$$

donde p_j indica el tiempo de procesamiento de la tarea j.

- Desaceleración promedia ponderada (Average Weighted Slowdown AWSD) [Ernemann04]

$$AWSD = \frac{\sum_{j \in \xi(t)} p_{jj} \cdot (T_j - R_j)}{\sum_{j \in \xi(t)} p_{jj}^2} \tag{2.7}$$

donde $\xi(t)$ representa el conjunto de tareas finalizadas en tiempo t y m_j denota el numero de maquinas usadas para la ejecución de la tarea j.

- Tiempo de espera (Wait Time) [Ernemann04]

$$WaitTime_j = S_j - R_j \tag{2.8}$$

donde S_j representa el tiempo de inicio para la tarea j.

Uno de los criterios de optimización más estudiados en sistemas Grid es *makespan*. El *makespan* es un indicador de la productividad total de un sistema Grid. Por lo tanto, valores pequeños para *makespan* indican que el planificador está proporcionando una buena y eficiente planificación de tareas en los recursos. Por otra parte, la minimización del *makespan* no supone la optimización de otros criterios, de hecho, la optimización de *makespan* puede ir en detrimento de otros criterios de optimización.

2.6. Heurísticas de Planificación de Tareas en Grid

La planificación de tareas es uno de los trabajos más difíciles en un entorno de computación Grid y es la clave para el funcionamiento correcto de estos sistemas [Huang09].

En esta sección se expone una taxonomía con las heurísticas tradicionales de planificación de tareas en sistemas Grid. Esta taxonomía se expone en la figura 2.5. En primer lugar, la

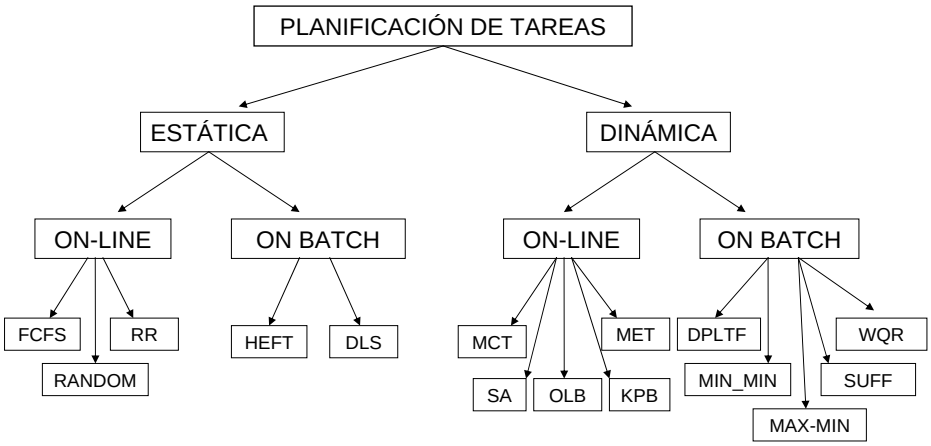


Figura 2.5: Clasificación de heurísticas

planificación de tareas está dividida en dos enfoques: estática y dinámica. Los algoritmos estáticos no tienen en cuenta toda la información relacionada con las características de los recursos y las decisiones de planificación son tomadas antes de la ejecución de la aplicación, mientras que, los algoritmos dinámicos estudian el estado del Grid antes de asignar una tarea y las decisiones de planificación son realizadas durante la ejecución de la aplicación [Xhafa10b].

A partir de esta clasificación, es posible tener en cuenta dos tipos de problemas de planificación [Chauhan10]: modo on-line y modo batch.

- En el modo on-line, las tareas son asignadas a los recursos de forma inmediata a su llegada [Aziz08]. Cuando la tasa de llegada de tareas es baja, este modo obtiene un mejor comportamiento porque las tareas no necesitan esperar al siguiente evento [Christodoulopoulos09].

- En el modo batch, cuando las tareas llegan al planificador, éstas no son enviadas a un recurso, sino que son recogidas en una Bolsa de Tareas (BoT) [Iosup11] donde se mantienen a la espera hasta que se produce el siguiente evento de envío. Estos eventos comienzan cada un determinado tiempo o cuando la BoT está llena. Este modo obtiene mejores resultados cuando la tasa de llegada de tareas es alta porque existe un gran número de tareas en ejecución en el sistema que mantienen a los recursos ocupados.

Mientras que el modo on-line considera la asignación de una tarea una sola vez, el modo batch considera cada tarea en cada evento hasta que la tarea comienza su ejecución. En el modo on-line, no hay retraso entre los eventos, las tareas son enviadas según llegan, pero su comportamiento no es tan bueno como en el modo batch que ofrece mejores resultados cuando la llegada de tareas es muy intensa.

De acuerdo con la literatura previa, existen muchos algoritmos de planificación para ambos modos. En el modo on-line existen cinco heurísticas tradicionales:

- Opportunistic Load Balancing (OLB). La heuristica OLB recoge una tarea arbitraria y asigna esta al próximo recurso que se espera que esté disponible [Kousalya08]. No tiene en cuenta el tiempo previsto de ejecucicón de la tarea en ese recurso. Las ventajas de esta heurística son: su simplicidad y la intención de mantener todos los recursos tan ocupados como sea posible. Esta heurística también es conocida como First Come First Served (FCFS).

- Minimun Execution Time (MET). Esta heurística asigna cada tarea a un recurso elegido que obtenga el menor tiempo de ejecución previsto para esa tarea sin tener el cuenta el tiempo transcurrido hasta que el recurso está disponible [Liu07]. Con esta heurística se produce un grave desequilibrio de la carga en los recursos, aunque MET asigna cada tarea al mejor recurso.

- Minimun Completion Time (MCT). La heurística MCT asigna cada tarea, en orden arbitrario, al recurso con el menor tiempo de finalización previsto [Yu08]. MCT combina los beneficios de OLB y MET y trata de evitar las circunstancias en que las heurísticas OLB y MET realizan un mal comportamiento.

- Switching Algorithm (SA). Esta heurística usa MCT y MET de forma cíclica dependiendo de la carga de los recursos [Sahu11].

- K-Percent Best (KPB). La heurística KPB considera, para cada tarea, un subconjunto de recursos. Este subconjunto está formado por los mejores recursos según el menor tiempo de finalización previsto [Tseng09].

En el modo batch existen un grupo de tres heurísticas dinámicas que están centradas en minimizar el tiempo de finalización de la BoT [Weng05]. Estos tres algoritmos operan de una forma similar y todos están basados en el cálculo del MCT. Estas heurísticas usan una BoT para su funcionamiento. Para cada tarea de la bolsa calculan el MCT y, según el algoritmo empleado, seleccionan la tarea a enviar al recurso para su ejecución. La tarea recién seleccionada es eliminada de la bolsa y el proceso es repetido hasta que todas las tareas de la bolsa son asignadas a un recurso.

- Min-Min. La heurística Min-Min [YuMing08] comienza con un conjunto T de tareas no asignadas. Entonces se busca un conjunto M que represente el mínimo tiempo de finalización de cada tarea en T. A continuación, la tarea con el menor tiempo total de finalización de M es seleccionada y asignada al recurso correspondiente (de ahí el nombre de Min-min). Para finalizar, la tarea recién asignada es eliminada de T, y el proceso, se repite hasta que todas las tareas se asignan a un recurso (es decir, el conjunto T está vacío). Min-min está basado en el Minimun Completion Time. Sin embargo, Min-Min considera todas las tareas

no asignadas en cada decisión de asignación y MCT solo considera una tarea una vez. Min-Min asigna tareas con el fin de que los cambios en el estado de disponibilidad del recurso sean la menor cantidad de cualquier trabajo posible. Sea t_i la primera tarea asignada por Min-Min en un sistema vacío. El recurso que finaliza t_i primero, m_j, es también el recurso que ejecuta t_i más rápido. Para cada tarea asignada después de t_i, la heurística Min-Min cambia el estado de disponibilidad de m_j por la menor cantidad posible para cada tarea. Por lo tanto, el porcentaje de tareas asignadas a su primera opción (sobre la base de tiempo de ejecución) es probable que sea mayor en Min-min que para Max-min (ver más abajo). Un menor makespan puede ser obtenido si más tareas son asignadas a los recursos que las completan primero y también las ejecutan más rápido. Min-Min da prioridad a las tareas que pueden ser completadas antes, por lo que, las tareas con menor MCT son ejecutadas primero. El porcentaje de tareas asignadas en la primera opción es mayor que usando el algoritmo Max-Min.

- Max-Min. La heurística Max-Min es muy similar a la heurística Min-Min. El algoritmo Max-Min [Lee06] selecciona la tarea con el mayor MCT de la bolsa de tareas y es asignada al recurso correspondiente. Esta heurística, también comienza con un conjunto T de tareas no asignadas. El conjunto M que representa el mínimo tiempo de finalización es buscado. En primer lugar, la tarea con el máximo valor de M es seleccionada y asignada al recurso correspondiente; a continuación, la tarea recien asignada es eliminada del T y el proceso se repite hasta que todas las tareas son asignadas a un recurso, es decir, hasta que el conjunto T se encuentra vacío. Max-Min minimiza los efectos de ejecución de las tareas con mayor tiempo de ejecución. Asignando las tareas con mayor tiempo de ejecución a los mejores recursos en primer lugar, permite que esta tarea sea ejecutada de forma conjunta con el resto de tareas, que tienen un menor tiempo de ejecución. Este algoritmo produce un mejor mapeo de tareas que Min-Min, donde todas las tareas cortas son ejecutadas en primer lugar y las tareas largas quedan para el final, que serán ejecutadas en pocos recursos mientras que el resto de recursos permanecen ociosos. Por lo tanto, el algoritmo Max-Min obtiene un mejor balance de carga a través de los nodos y un mejor tiempo de finalización.

  Los algoritmos Min-Min y Max-Min son simples y pueden ser fácilmente modificados para adaptarse a diferentes escenarios. Por ejemplo, en [He03], la heurística Min-Min es modificada para que se puedan garantizar requisitos de QoS de determinadas tareas y minimizar el "makespan" al mismo tiempo. En [Min-You00] es implementada una modificación de Min-Min denominada Segmented Min-Min. En esta heurística, las tareas son primero ordenadas por tiempo de finalización previsto (el método de ordenación puede ser: máximo ECT, mínimo ECT o ECT promedio de todos los recursos); entonces, la secuencia ordenada es segmentada, y, finalmente, Min-Min se aplica a todos estos segmentos. Esta heurística mejora el funcionamiento típico de Min-Min cuando la longitud de las tareas es muy variada, dando la oportunidad a más tareas para que se ejecuten antes que en el algoritmo tradicional de Min-min.

- Sufferage. En la heurística Sufferage [YuMing08], las tareas con alto valor de "sufrimiento" (sufferage) tienen prioridad. Para cada tarea, el valor de sufrimiento es definido como la diferencia entre el mejor MCT y el segundo mejor MCT para esa tarea. Una variación del algoritmo de sufrimiento es el XSufferage [Falzon10], en este caso el valor de sufrimiento no es calculado a nivel de recurso, sino que es calculado al nivel de nodo, es decir, calculando el mínimo de los MCT en todos los recursos de cada nodo.

Como ha sido indicado anteriormente, la planificación de tareas es conocida por ser un problema NP completo. Por lo tanto las heurísticas de Soft Computing hacen frente a esta dificultad. Existen trabajos que usan heurísticas simple: Tabu Search (TS) es usado en [Fayad07] [Xhafa09] para encontrar soluciones optimas al problema de asignación de tareas a recursos; con Simulated Annealing (SA) [Paletta09] es obtenido un alto rendimiento de planificación. Particle Swarn Optimization [Tseng08] [Hongbo10] y algoritmos genéticos [Lim07] [Priya07] también son usados para planificar tareas en Grid. Alguna de estos enfoques usan heurísticas de Soft Computing para ordenar la bolsa de tareas [Chang09] [Franke08].

2.7. Resumen y Conclusiones

En este capítulo los antecedentes generales de planificación en computación Grid han sido presentados. Por un lado, se han introducido las mayores dificultades que hacen que la planificación en computación Grid sea diferente de la planificación en otros sistemas de computación distribuida. Han sido repasados los conceptos esenciales en la planificación de Sistemas Grid. También han sido realizadas diversas clasificaciones de Sistemas Grid según: el grado de complejidad, ámbito de los recursos que participa y su finalidad. Por otro lado, con el fin de situar el contexto de esta tesis, se ha realizado una revisión de las heurísticas existentes para la planificación de tareas en Sistemas Grid.

La reciente aparición de redes de alta velocidad y la creciente necesidad de nuevas infraestructuras a gran escala que hagan frente a los problemas informáticos han llevado a los sistemas Grid a ser una plataforma prometedora. Un Grid se compone de un conjunto heterogéneo de recursos y capacidades, distribuidas geográficamente, con el fin de lograr un objetivo común. Los recursos pueden pertenecer a diferentes dominios, considerando sus propias políticas de acceso y restricciones de seguridad; por lo tanto, la coordinación y la cooperación se consideran claves para su posible utilización. En este sentido los sistemas Grid ofrecen nuevas oportunidades de investigación, de entre las que destacan los algoritmos de planificación de tareas.

Por último, hay que señalar que esta tesis se centra en los sistemas de planificación Grid. Un aspecto clave del proceso de planificación está relacionado con la gestión de incertidumbre, ya que la información disponible de los componentes de un Grid es en su mayoría imprecisa y cambiante debido al dinamismo del sistema, por ello usando planificadores Grid con sistemas expertos basados en reglas borrosas, es posible gestionar esta imprecisión y lograr resultados eficientes.

Capítulo 3

Sistemas Borrosos

3.1. Introducción

Los Sistemas Borrosos proporcionan un marco de procesamiento de la información basados en conocimiento, formulándola de una manera sistemática que intenta emular las propiedades del razonamiento humano. Esta técnica combina la Teoría de los Conjuntos Borrosos y la Lógica Borrosa [Zadeh65] y consigue reproducir comportamientos complejos, enunciados en forma de relaciones cualitativas e imprecisas, propios del lenguaje natural, y expresarlos en términos numéricos, compatibles con los métodos usados en ingeniería [Zadeh73].

Un sistema borroso, en sentido amplio, es un sistema basado en lógica borrosa, donde la lógica borrosa puede utilizarse como base para la representación de diferentes formas de conocimiento, o para modelar las interacciones existentes entre las variables de un sistema [Herrera97].

El controlador borroso [Mamdani74] [Mamdani75] fue el primer tipo de sistema basado en reglas borrosas [Cordon01]. Mamdani incrementó la formulación inicial de Zadeh de forma que permitía a los sistemas borrosos trabajar en sistemas de control.

En este capítulo se realiza un repaso del funcionamiento de un sistema experto basado en reglas borrosas. Es importante señalar que las descripciones que se realizan se plantean bajo un punto de vista introductorio, con el único objetivo de situar el problema con el que se ha trabajado durante el desarrollo de esta tesis.

3.2. Sistemas Expertos Basados en Reglas Borrosas

3.2.1. Introducción

El desarrollo de sistemas basados en reglas borrosas se considera como una de las más prósperas y fructíferas aplicaciones de la teoría de conjuntos borrosos. Ya en los trabajos iniciales [Zadeh73] [Zadeh75], Zadeh introdujo la idea de formular el problema de control mediante el uso de reglas expresadas con representaciones lingüísticas. Las experiencias diarias de la vida real ofrecen muchos ejemplos donde se confirma cómo el entendimiento, el pensamiento y la habilidad

humana pueden resolver eficientemente el problema de gestión para una gran variedad de sistemas sin hacer uso de los sofisticados algoritmos de la teoría de control. Los primeros trabajos en los que se aplicó la lógica borrosa al control de procesos fueron desarrollados por Mamdani, particularmente, en un sistema de control para una máquina de vapor. Desde entonces, el número de aplicaciones de los sistemas basados en reglas borrosasha ido en aumento en todos los campos de la ingeniería.

Estos sistemas tienen potencialmente un gran número de ventajas cuando las especificaciones del problema requieren robustez, adaptabilidad y flexibilidad debido a perturbaciones del entorno o a efectos no modelables de la dinámica del sistema. Los sistemas borrosos son básicamente controladores basados en reglas que discretizan el espacio de operación continuo, no lineal y multidimensional en clases discretas. Utilizando lógica multivaluada, las variables borrosas de las reglas preservan la información cuantitativa en los correspondientes valores de clase y el valor de pertenencia a esa clase. El comportamiento del sistema borroso es optimizado mediante una máquina de estados finitos en términos de las clases discretas y finalmente usa la información de la función de pertenencia para hacer una interpolación suave entre los puntos vecinos del espacio de operación continuo.

No es fácil responder a la pregunta de cuándo aplicar, o no, un sistema basado en reglas borrosas, por tanto, con el objetivo de responder a esta pregunta, se exponen los principales beneficios y desventajas que exhibe esta tecnología.

3.2.2. Beneficios y desventajas de los Sistemas Borrosos

Se pueden mencionar cuatro aspectos que hacen que los sistemas borrosos sean atractivos, presentando ventajas sobre los sistemas o controladores clásicos.

- **Robustez.** De manera contraria a los controladores tradicionales que son muy sensibles a variaciones de parámetros, un sistema borroso puede tratar con mucha más seguridad un problema cuyos parámetros son variantes en el tiempo.
- **Flexibilidad.** Un sistema borroso puede ser diseñado con muy poco conocimiento del proceso que se supone controlará. Consecuentemente, el mismo sistema, mediante el ajuste de sus parámetros de operación, puede ser usado en diferentes tipos de procesos.
- **Adquisición del Conocimiento Experto.** Una de las mayores cualidades de los sistemas basados en reglas borrosas es su gran capacidad para capturar de una forma natural el conocimiento que un operador experto tiene de un proceso o sistema. Mediante el uso de variables lingüísticas como “alto”, “normal”, “muy alto”, etc., un experto puede expresar su conocimiento mediante reglas de la forma:

  Si el nivel es alto, **entonces** disminuir flujo de agua.

  Si la presión es baja, **entonces** aumentar flujo de agua.

  Si el consumo es nulo, **entonces** flujo de agua nulo.

- **Alto grado de automatización.** En la industria existen aplicaciones, en donde la instalación de un sistema borroso en el nivel supervisor o de control de calidad, de todo el sistema de producción consigue un alto grado de automatización, permitiendo obtener sustanciales reducciones tanto de la variación de la calidad del producto, como del consumo de energía y materia prima.

Algunas de las desventajas y de los límites actuales de su aplicación son indicadas a continuación.

- **Sistematización.** No existen procedimientos que permitan sistematizar el diseño de sistemas borrosos. Siempre son diseñados heurísticamente y basándose en el conocimiento de un operador experto.

- **Adquisición de conocimiento.** Si bien, como se ha mencionado anteriormente, una de las principales cualidades de los sistemas borrosos es su facilidad para expresar de manera fácil el conocimiento de un operador experto, cuando el conocimiento a priori y relevante de la operación del proceso no está disponible, es pobre, inadecuado, difícilmente representable bajo el paradigma basado en reglas, o no resulta muy consistente, sus benignas cualidades pueden tornarse en serias desventajas y debe de reconsiderarse la idoneidad de la tecnología borrosa para estos casos.

3.2.3. Conjuntos Borrosos

La noción de conjunto refleja la tendencia a organizar, generalizar y clasificar el conocimiento sobre los objetos del mundo real. El encapsulamiento de los objetos es una colección cuyos miembros comparten una serie de características o propiedades que implican la noción de conjunto. Los conjuntos introducen una noción de dicotomía, que en esencia es una clasificación binaria: o se acepta o se rechaza la pertenencia de un objeto a una categoría determinada. Habitualmente la decisión de aceptar se denota como 1 y la de rechazar como 0. Esta decisión de aceptar o rechazar se expresa mediante una función característica, según las propiedades que posean los objetos del conjunto.

La Lógica Borrosa se fundamenta en el concepto de conjunto borroso [Zadeh65] que suaviza el requerimiento anterior y admite valores intermedios en la función característica, que se denomina función de pertenencia. Esto permite una interpretación más realista de la información, puesto que la mayoría de las categorías que describen los objetos del mundo real, no tienen unos límites claros y bien definidos.

Para cualquier conjunto clásico C definido en el universo de discurso U es posible definir una *función característica* $Memb_C: U \rightarrow \{0, 1\}$ como:

$$Memb_C(u) = \begin{cases} 1 & si \quad u \in C \\ 0 & si \quad u \notin C \end{cases} \tag{3.1}$$

En teoría de conjuntos borrosos, esta función característica es generalizada como una función de pertenencia que asigna a cada elemento $u \in U$ un valor en el intervalo unitario [0,1]. El conjunto F obtenido en base a tal función de pertenencia es definido como *Conjunto Borroso.*

Un **conjunto borroso** en un *universo de discurso* U, se define como:

$$F = Memb_F(u_1)/u_1 + ... + Memb_F(u_n)/u_n = \sum_{i=1}^{n} Memb_F(u_i)/u_i \tag{3.2}$$

Intuitivamente, la idea de conjunto borroso hace referencia al grado de pertenencia de los elementos al conjunto, de forma que se asignarán grados de pertenencia comprendidos entre los valores 0 (exclusión total del elemento) y 1 (pertenencia completa del elemento). Los valores intermedios indican una pertenencia parcial de los elementos al conjunto. De esta forma, los conjuntos borrosos constituyen un método natural para representar la imprecisión y subjetividad propias de la actividad humana.

La **función de pertenencia** $Memb_F$ del conjunto borroso F es:

$$Memb_F \colon U \to [0, 1] \tag{3.3}$$

donde, para cada elemento $u \in U$, se asigna el grado de pertenencia $Memb_F(u) \in [0, 1]$. De esta manera, F queda completamente determinado por el conjunto de pares:

$$F = (u, Memb_F(u))|u \in U \tag{3.4}$$

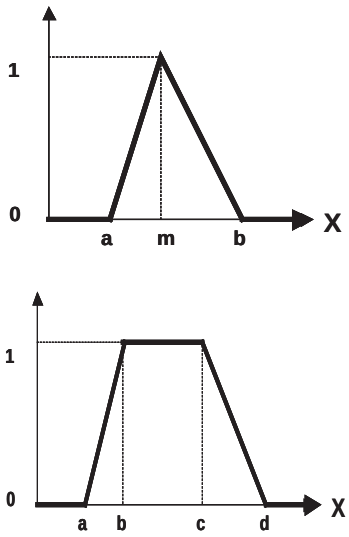
Formalmente, dado un universo U, un subconjunto borroso A se define por medio de su función de pertenencia, $\mu(x)$ definida sobre U, con valores en el intervalo [0,1]. A continuación se presentan algunas de las funciones de pertenencia más utilizadas:

- Triangular

$$\mu(x) = \begin{cases} 0, & si \quad x \leq a \\ \frac{(x-a)}{(m-a)}, & si \quad x \in (a, m] \\ \frac{(b-x)}{(b-m)}, & si \quad x \in (m, b] \\ 0, & si \quad x \geq b \end{cases} \tag{3.5}$$

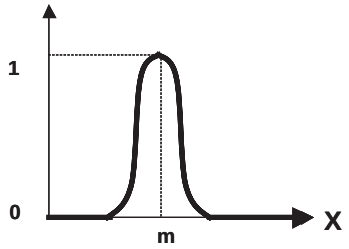
- Trapezoidal

$$\mu(x) = \begin{cases} 0, & si \quad x \leq a, \qquad x \geq d \\ \frac{(x-a)}{(b-a)}, & si \quad x \in (a, b] \\ 1, & si \quad x \in (b, c) \\ \frac{(d-x)}{(d-c)}, & si \quad x \in (c, d] \end{cases} \tag{3.6}$$



- Gaussiana

$$\mu(x) = e^{-k(x-m)^2} \tag{3.7}$$



Una **variable lingüística** es una descripción simbólica constante utilizada para representar, en general, una cantidad variando en el tiempo dentro del intervalo definido por un universo de discurso U_i. Los diferentes valores que esta variable puede adquirir son conocidos como *valores lingüísticos.*

3.2.4. Arquitectura de Sistemas Basados en Reglas Borrosas

Un sistema de inferencia borroso está compuesto por cuatro bloques principales [Cordon01], [Magdalena96], [Magdalena97]. Estos bloques son mostrados en la figura 3.1.

Los elementos que constituyen los sistemas borrosos basados en reglas, son los siguientes:

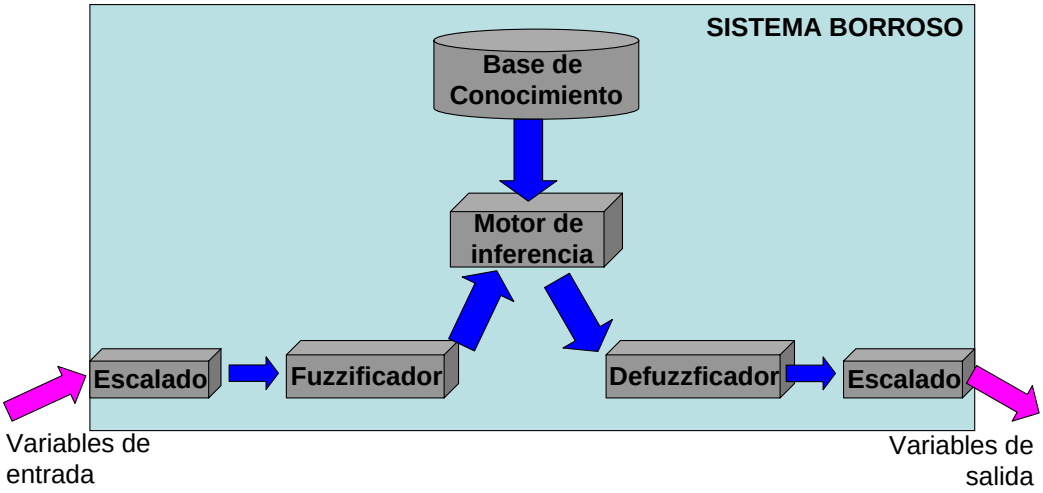


Figura 3.1: Estructura de un sistema de control borroso.

- Fuzzificador
- Base de conocimiento
- Motor de inferencia
- Defuzzificador

Además de los sistemas constituyentes de un sistema borroso, son necesarios, dos sistemas adicionales, uno a la entrada y otro a la salida que permitan escalar las magnitudes de las variables con las que trabaja el sistema, de manera que lo haga de forma normalizada, es decir, con valores comprendidos en el intervalo [0,1] o en el intervalo [-1,1].

A continuación se describirán detalladamente las funciones de cada uno los elementos que forman un sistema borroso. Es importante tener en consideración que aunque el diseño de un sistema basado en reglas borrosas implica la definición de cada uno de los cuatro módulos, se asume que en algún sentido, antes de comenzar la especificación de cada uno de ellos en detalle, ya se conoce la forma que tendrá. Estos es debido a que existe una estrecha interdependencia entre todos ellos; por un lado, a medida que se van definiendo los parámetros de un módulo y se avanza hacia la especificación de los otros, es requerido un proceso de iteración; por otro lado, el proceso puede empezar a partir de cualquiera de ellos y avanzar en el diseño en la determinación de los demás. Por ejemplo, no es posible establecer la forma en la que la información medida será convertida en borrosa sin conocer previamente la forma en la que se obtendrán las reglas. Por ejemplo, si las reglas son obtenidas a partir del conocimiento heurístico de un operador experto, es muy posible que se decida la selección del número de clases o valores lingüísticos de cada variable

borrosa en función del conocimiento expresado por el experto. Por el contrario, cuando no se cuenta con información heurística para describir los criterios de operación, se fijarán un número aproximado de clases y la definición de las reglas deberá de limitarse al número predefinido. Las ideas de interdependencia y de un proceso de prueba y error deben ser mantenidas en mente durante la descripción de la arquitectura general de los sistemas borrosos que se da a continuación. El orden de exposición no es el orden de diseño; más bien, es el orden que sigue el flujo de información a medida que las señales de entrada son procesadas para encontrar una respuesta.

3.2.4.1. Fuzzificador

Naturalmente, cuando se realizan medidas en cualquier proceso, para posteriormente tomar una decisión, éstas no vienen en forma de conjunto borroso. Por lo que es necesario un módulo que transforme esos valores de la entrada del sistema en conjuntos borrosos que pueden ser tratados por el motor de inferencia. Hay varias funciones que pueden estar incluidas en el proceso llevado acabo por la interfaz de fuzzificación:

1. Obtener la medida de las variables de entrada. En las aplicaciones borrosas los valores obtenidos como entrada suelen ser valores escalares.

2. Normalización. Si la gama de valores asignados a cada variable lingüística en los antecedentes de la base de reglas está definida en un dominio normalizado, al valor de entrada debe aplicársele el factor de escala correspondiente, que relaciona el universo de discurso de la variable de entrada con el espacio normalizado de la variable lingüística.

En definitiva la función de un sistema fuzzificador es convertir un valor no borroso, “crisp”, en un valor borroso.

3.2.4.2. Base de conocimiento

La primera tarea a llevar a cabo cuando se diseña un sistema borroso es identificar cuáles serán las variables de entrada y salida, así como los parámetros que las definirán. La determinación de qué variables son utilizadas para diseñar el sistema quedará, a partir de este momento, implícita en la configuración del sistema y en la Base de Conocimiento.

La Base de Conocimiento está formada a su vez por dos componentes. El primero es una Base de Datos que mantiene los parámetros y las características de cada una de las variables lingüísticas, el segundo es la Base de Reglas que almacena el conocimiento respecto a la causalidad entre las variables de entrada y las variables de control. Desde un punto de vista, se puede decir que la base de datos da los elementos que permiten la expresión de la base de reglas, pero visto de manera inversa, también puede decirse que es la base de reglas la que, una vez expresada, se formaliza mediante los parámetros registrados en la base de datos.

Base de Datos

La base de datos proporciona información a la interfaz de fuzzificación, al mecanismo de inferencia y a la interfaz de defuzzificación. Así mismo, proporciona los elementos del lenguaje para construir las reglas. Su contenido está estructurado con respecto a cada variable lingüística, proporcionando para cada una:

- El número de valores lingüísticos. Una vez determinadas las variables de entrada y de salida del sistema borroso, debe ser definido el número de valores lingüísticos de cada variable lingüística. La determinación de este número es también conocido como *clustering, cuantificación, partición o discretización* del universo de discurso y es una de las operaciones que involucran mayor heurística en el diseño de un sistema borroso. Existen algunos métodos para encontrar la partición óptima del dominio de una variable en función de un conjunto representativo de datos [Krishnapuram93] [Cheeseman90].

- Los parámetros de la función de pertenencia correspondientes a cada valor lingüístico. Cada valor lingüístico es definido por su correspondiente función de pertenencia. En el desarrollo de sistemas basados en reglas borrosas han sido utilizados diferentes tipos de funciones de pertenencia. Debido a la facilidad de su descripción paramétrica y funcional, las funciones más comúnmente usadas son: triángulares, trapezoidales y gaussiana.

  Cuando se ha determinado el tipo de función de pertenencia para cada valor lingüístico $F_{i,j}$, en el universo de discurso U_i con $Memb_F : U \rightarrow [0,1]$, los parámetros que la describen deben de ser fijados. Los más importantes son:

  - Valor Máximo

    $$m_{i,j} \qquad \text{tal que} \qquad Memb_{X_{i,j}}(m_{i,j}) = 1.0$$

    Si la función es triangular o gaussiana el valor máximo es único, si la función es trapezoidal el valor máximo es un intervalo.

  - Soporte

    $$X_{i,j} = \{x_i\} \qquad \text{tal que} \qquad Memb_{X_{i,j}}\ (x_i) > 0$$

    El intervalo de soporte de una función borrosa es dividido en dos regiones por el valor máximo, estableciéndose la región izquierda y derecha. Si ambas regiones son iguales, se dice que la función es simétrica.

  - Punto de cruce

    $$c_{i,j} \qquad \text{tal que} \qquad Memb_{X_{i,j}}(c_{i,j}) = \alpha$$

La determinación de este parámetro es muy importante ya que representa el grado de intersección entre dos (o más) valores lingüísticos contiguos.

- El factor de normalización y desnormalización aplicado al universo de discurso. Cuando el diseño del sistema incluye las operaciones de normalización y desnormalización, la base de datos mantiene el registro de los diferentes factores de escala aplicados a cada una de las señales de entrada y de salida permitiendo mapear los valores físicos del sistema real en el dominio normalizado en el que opera el controlador.

Base de Reglas

La base de reglas es utilizada por el motor de inferencia. De forma que la base de datos contiene información acerca de la forma de los conjuntos borrosos, mientras que la base de reglas contiene proposiciones borrosas, utilizando para ello la información recogida en la base de datos.

Las reglas borrosas combinan uno o más conjuntos borrosos de entrada (antecedentes) y las asocian uno de salida (consecuente). Existen dos tipos de formato de reglas:

- **Mamdani**: asocian la salida a un conjunto borroso de otro universo de discurso.
- **Sugeno**: la función de salida es una combinación lineal de las variables de entrada.

Las reglas se definen por sus antecedentes y consecuentes [Velasco95], los cuales están asociados a conceptos borrosos. La estructura más común de las reglas en los controladores borrosos implica el uso de variables lingüísticas [Zadeh75] asociadas a los conjuntos borrosos. Este tipo de reglas, cuando utilizan múltiples entradas y una única salida, tienen la siguiente forma:

$$\begin{array}{lll} \text{[TIPO 1]:} & R_1: & \textbf{if } x_1 \text{ is } A_{11} \textbf{ and } \ldots \textbf{and } x_m \text{ is } A_{m1} \textbf{ then } y \text{ is } B_1 \\ & \ldots & \\ & R_n: & \textbf{if } x_1 \text{ is } A_{1n} \textbf{ and } \ldots \textbf{and } x_m \text{ is } A_{mn} \textbf{ then } y \text{ is } B_n \end{array}$$

Aunque ocasionalmente los consecuentes pueden ser funciones analíticas de las variables de entrada:

$$\begin{array}{lll} \text{[TIPO 2]:} & R_1: & \textbf{if } x_1 \text{ is } A_{11} \textbf{ and } \ldots \textbf{and } x_m \text{ is } A_{m1} \textbf{ then } y = f_1(x_1, \ldots x_m) \\ & \ldots & \\ & R_n: & \textbf{if } x_1 \text{ is } A_{1n} \textbf{ and } \ldots \textbf{and } x_m \text{ is } A_{mn} \textbf{ then } y = f_n(x_1, \ldots x_m) \end{array}$$

En cualquiera de los dos posibles casos, x_i son las variables de entrada, A_{ij} son conjuntos borrosos relativos a las variables de entrada, y es la variable de salida, B_k, son los conjuntos borrosos relativos a la variable de salida, y fl son funciones de las variables de entrada, normalmente de la forma:

$$fl(x_1, \ldots, x_m) = a_{0l} + a_{1l}x_1 + a_{2l}x_2 + \ldots + a_{ml}x_m$$

La conectiva and, entre conceptos borrosos, es normalmente implementada a través de cualquier T-norma (podría implementarse con los operadores producto o mínimo) [Cordon01].

Los sistemas que utilizan el primer tipo de reglas son llamados normalmente controladores de tipo Mamdani [Mamdani74][Mamdani75], mientras que los utilizan el segundo tipo de reglas, son normalmente denominados controladores de tipo TSK (Takagi, Sugeno, Kang) [Takagi85].

El comportamiento completo de un sistema borroso puede ser caracterizado por una relación borrosa que es una combinación de todas las relaciones borrosas por cada elemento del conjunto de reglas. Esta combinación puede representarse a través de la conectiva *also* [Cordon01]:

$$R = also(R_1, \ldots, R_n)$$

Esta conectiva, podría implementarse con cualquier T-conorma (normalmente implementada con el operador máximo) [Cordon01], y se utiliza para generar la salida borrosa. Esta salida borrosa consiste en un subconjunto borroso de y. Al conjunto de todas las reglas utilizadas para describir el comportamiento del sistema junto con los datos necesarios para poder definir los conjuntos borrosos, es lo que se conoce con el nombre de base de conocimiento [Cordon01].

3.2.4.3. Motor de inferencia

Este módulo representa al sistema que infiere acciones utilizando implicaciones borrosas, y las reglas de inferencia de lógica borrosa. Es decir, el motor de inferencia obtiene un conjunto borroso para la salida a partir de los conjuntos borrosos para la entradas conforme a una relación definida a partir del conjunto de reglas borrosas. La mayoría de los sistemas expertos almacenan su conocimiento en forma de reglas de inferencia. Entre las principales reglas tenemos:

- Modus Ponens: es quizás la regla de inferencia mas comúnmente utilizada. Se utiliza para obtener conclusiones simples. En ella, se examina la premisa de la regla, y si es cierta, la conclusión pasa a formar parte del conocimiento. Por ejemplo, se supone que tenemos la regla, "Si A es cierto, entonces B es cierto" y que se sabe además que A es cierto. La regla Modus Ponens concluye que B es cierto. Esta regla de inferencia, que parece trivial, debido a su familiaridad, es la base de un gran número de sistemas expertos.

- La regla de inferencia Modus Tollens se utiliza también para obtener conclusiones simples. En este caso se examina la conclusión y si es falsa, se concluye que la premisa también es falsa. Por ejemplo, de nuevo se tiene la regla, "Si A es cierto, entonces B es cierto" pero se sabe que B es falso. Entonces, utilizando la regla Modus Ponens no se puede obtener ninguna conclusión pero la regla Modus Tollens concluye que A es falso.

El rendimiento del motor de inferencia depende del conjunto de reglas en su base de conocimiento. Hay situaciones en las que el motor de inferencia puede concluir utilizando un conjunto de reglas, pero no puede, utilizando otro (aunque éstos sean lógicamente equivalentes).

3.2.4.4. Defuzzficador

El proceso de defuzzificación es el proceso inverso de la fuzzificación, es decir, es el proceso de convertir un valor borroso (o una conclusión borrosa) en información concreta expresada mediante un escalar. Ningún dispositivo es capaz de interpretar una señal borrosa como entrada. Necesariamente las conclusiones borrosas que el mecanismo de inferencia produce deben ser recodificadas a un valor escalar. Zadeh fue el primero en percatarse de este problema y sugirió algunas posibilidades para resolverlo. Actualmente existe un buen número de métodos de defuzzificación, pero como ocurre con otros elementos del diseño de sistemas basados en reglas borrosas, no existe aún un procedimiento sistemático para seleccionar el más adecuado, dependiendo del caso de aplicación particular.

Existen varias formas para implementar la defuzzificación, cada una de ellas con sus ventajas e inconvenientes, pero las más utilizadas son el método del centro de gravedad y el método de la media de los máximos. Las expresiones correspondientes para obtener las salidas no borrosas son las siguientes:

$$y_o = \frac{\int\limits_Y \mu_o(y) y dy}{\int\limits_Y \mu_o(y) dy} \tag{3.8}$$

$$y_o = \frac{\int\limits_{Y^*} y dy}{\int\limits_{Y^*} dy} \tag{3.9}$$

donde Y es el universo de discurso de la variable de salida, $\mu_o(y)$ es la función de pertenencia del conjunto borroso de salida, e Y^* es el $\alpha - cut$ de la salida borrosa con α igual al máximo de $\mu_o(y)$.

Cuando se utilizan consecuentes que son funciones de las variables de entrada, la conectiva also se implementa como la suma ponderada de las salidas de las reglas [Takagi85].

$$y_o = \frac{\sum\limits_n \mu_i y_i}{\sum\limits_n \mu_i} \tag{3.10}$$

Donde y_i es la salida de la regla i, y μ_i es el grado de verdad del antecedente de la regla i calculado como el mínimo de los grados de pertenencia de las variables de entrada a los conjuntos borrosos indicados en el antecedente.

3.3. Resumen y Conclusiones

En este capítulo se ha presentado un repaso y estudio de las principales características de los sistemas basados en reglas borrosas. Se ha mostrado como los sistemas borrosos utilizan la lógica

borrosa como base para la representación de conocimiento. Han sido repasados los beneficios y desventajas de usar sistemas borrosos. También se ha expuesto la arquitectura de un sistema borroso, mostrando cada uno de sus bloques principales: fuzzificador, base de conocimiento, motor de inferencia y defuzzficador.

Los sistemas basados en reglas borrosas incorporan conocimiento experto en su base de conocimiento y tienen facilidad para trabajar en entornos sujetos a incertidumbre e imprecisión.

Aunque existen una gran diversidad de métodos de planificación de tareas que pueden ser encontrados en la literatura, la gestión eficiente de la incertidumbre y dinamismo inherente a los recursos y aplicaciones de los sistemas Grid sigue siendo un reto en la planificación de estos sistemas. En la planificación de tareas, el tratamiento de la incertidumbre asociada a los recursos de un Sistema Grid resulta fundamental, por lo tanto, en esta tesis, se utilizan sistemas expertos basados en reglas borrosas para la planificación de tareas en computación Grid. La funcionalidad de este sistema borroso se comprobará mediante la ejecución de una serie de algoritmos de análisis de imágenes con altas necesidades de computación.

Capítulo 4

Análisis de Imágenes

4.1. Introducción

Análisis de imágenes hace referencia a un conjunto de técnicas destinadas a extraer información relevante de un sistema, objeto de estudio, a partir de imágenes de dicho sistema. Este conjunto de técnicas comprenden operaciones cuyo origen es una imagen y cuyo resultado final es una nueva imagen de salida, que incrementa la cantidad de información que puede ser interpretada. El valor del píxel en la imagen de salida puede ser función del valor que tenía en la imagen de entrada, de los valores de sus vecinos o del valor de todos los puntos de la imagen de entrada. El objetivo de estas técnicas es procesar una imagen inicial de tal modo que la imagen resultante sea más adecuada que la imagen original para una aplicación específica, permitiendo extraer información para tomar una decisión.

En este capítulo se describen diversas técnicas de análisis de imágenes, que serán utilizadas en el desarrollo de esta tesis para la detección de defectos en tableros de fibra recubiertos con papel melamínico. En la primera sección se exponen los métodos de realce de imagen, detallando la técnica de detección de bordes, dentro de la cual se describen el filtro de Roberts, Prewitt, Sobel, Laplaciano y los métodos de Canny y Zerocross. A continuación se describe el proceso de segmetación por umbralización. Finalmente, se presenta la utilización del procesamiento morfológico.

4.2. Realce de Imagen

Los métodos de realce (o mejora) de una imagen posibilitan aumentar la visibilidad de una parte, aspecto o elemento de una imagen, aunque por lo general a costa de los demás aspectos o elementos, cuya visibilidad resulta disminuida. En este sentido, el procesamiento de imágenes permite reordenar los elementos para hacer un producto más agradable o interpretable, sin cambiar la cantidad total de datos. En el caso de las imágenes, esto generalmente significa que el número de bytes (o píxeles) no es reducido. Un ejemplo sería, el de identificar y contar las

características de una imagen, reduciendo la cantidad de datos de un millón de bytes a unas pocas docenas o incluso a un simple "sí" o "no" para algún control de calidad, sistema médico o aplicaciones forenses.

El procesamiento de imágenes con fines de realce o mejora se puede realizar en el dominio espacial (la matriz de píxeles que comprenden nuestro punto de vista convencional de la imagen) o en otros dominios, como el de Fourier. En el dominio espacial, los valores de píxeles pueden ser modificados de acuerdo a reglas que dependen del valor de los píxeles originales. Alternativamente, los valores de píxeles pueden ser combinados o comparados con otros, en su vecindad inmediata, en una gran variedad de maneras.

4.2.1. Manipulación de contraste

En la mayoría de los sistemas, aumentar el contraste de una imagen poco clara, se puede hacer de forma casi instantánea escribiendo una tabla de valores en el hardware de la pantalla. Esta tabla de búsqueda (Look Up Table, LUT) sustituye un valor de brillo de la pantalla para cada valor almacenado y por lo tanto no requiere tener que modificar cualquiera de los valores almacenados en la memoria pertenecientes a la imagen. Ampliar el rango de contraste asignando el valor píxel más oscuro a negro, el valor más brillante al color blanco, y cada uno de los otros tonos de gris usando interpolación lineal, hace mejor uso de las pantallas y mejora la visibilidad de los rasgos de la imagen.

La misma tabla de búsqueda puede ser usada con colores, asignado un triplete de valores rojo, verde y azul a cada valor de la escala de grises almacenada. Este pseudo-color también aumenta la diferencia visible entre píxeles similares, y a veces es una ayuda para el usuario que desea ver pequeños o graduales cambios de la luminosidad de la imagen.

Una pantalla de ordenador típico puede mostrar 2^8 ó 256 tonos distintos de gris, y muchas pueden producir colores con los mismos 2^8 valores de brillo para cada uno de los componentes rojo, verde y azul para producir un total de 2^{24} ó 16 millones de colores diferentes. Esto es descrito como "color verdadero", ya que la gama de colores que puede mostrar es la adecuada para reproducir escenas de forma más natural. Esto no implica, por supuesto, que los colores mostrados sean precisos o idénticos a la escena original mostrada. De hecho, ese tipo de precisión es muy difícil y requiere un hardware especial y calibración.

Los 16 millones de colores diferentes que tal sistema es capaz de mostrar, e incluso los 256 tonos de gris, son mucho más que los que el ojo humano puede distinguir. Bajo condiciones de buena visibilidad, por lo general se puede ver sólo unas pocas decenas de niveles de gris diferentes y distinguir cientos de colores. Eso significa que el hardware de la pantalla de un sistema de procesamiento de imágenes no se está utilizando muy bien para comunicar la información de la imagen al usuario. Si muchos de los píxeles de la imagen son muy brillantes, por ejemplo, no pueden ser distinguidos. Si también hay presentes algunos píxeles oscuros, no se puede simplemente ampliar el constraste.

En general, es posible describir la manipulación de la escala de grises en términos de una

función de transferencia sobre el valor de brillo almacenado en cada píxel a un valor mostrado. Si esta relación es uno a uno, para cada valor almacenado se muestra un valor único. En algunos casos, es conveniente utilizar las funciones de transferencia que no son de uno a uno: varios valores almacenados se muestran con el mismo valor de brillo, de modo que los demás valores almacenados se pueden transmitir más separados para aumentar su diferencia visual.

Invertir toda la gama de contraste produce el equivalente de un negativo fotográfico, que a veces mejora la visibilidad de los detalles (en especial en las regiones oscuras en la imagen original). Invirtiendo sólo una parte de la gama de luminosidad produce un efecto visual, que también se puede utilizar para mostrar los detalles en áreas sombreadas o saturadas.

El aumento de la pendiente de la función de transferencia produce una imagen en la que diferentes valores de brillo almacenados pueden tener el mismo brillo en la pantalla. Si la organización general de la imagen es familiar para el visor o espectador, esta modificación puede no ser muy perjudicial, y permite aumentar la visibilidad de las pequeñas diferencias. Sin embargo, este tipo de tratamiento es fácilmente exagerado y puede confundir en lugar de mejorar la mayoría de las imágenes.

De forma experimental modificando la función de transferencia hasta que la imagen "se ve bien", y muestra mejor las características de mayor interés, en última instancia, proporciona una herramienta flexible. En la mayoría de los casos, es conveniente disponer de funciones de transferencia que se pueden aplicar a una serie de imágenes, de manera que sea posible una comparación adecuada.

Las principales funciones de transferencia son: inversa, logarítmica, raíz cuadrada, exponencial, identidad, cuadrado o inversa del logaritmo. Cualquiera de estas funciones pueden ser utilizadas como añadidura de la expansión de contraste, que extiende la escala original a toda la gama de la pantalla.

4.2.2. Ecualización del histograma

Además de las tablas pre-calculadas y almacenadas, a veces es ventajoso construir una función transferencia de una imagen específica. A diferencia de las funciones arbitrarias es posible obtener un algoritmo específico que da resultados óptimos y reproducibles. El método más popular se llama ecualización del histograma.

La figura 4.1 muestra un ejemplo de una imagen con su histograma. El histograma de una imagen representa la frecuencia relativa de los niveles de gris de la imagen. La gráfica muestra el número de píxeles de la imagen para cada uno de los 256 posibles valores de brillo. Los picos en el histograma se corresponden con los valores de brillo más comunes, que suelen identificar estructuras particulares que se presentan. Los valles entre los picos indican los valores de brillo que son menos comunes en la imagen. Para una imagen en color, es posible mostrar tres histogramas correspondientes a cada uno de los tres ejes de color.

Por lo general, las imágenes tienen un único histograma. Incluso las imágenes de diferentes áreas de la misma muestra o escena, en el que las diferentes estructuras presentes tienen niveles

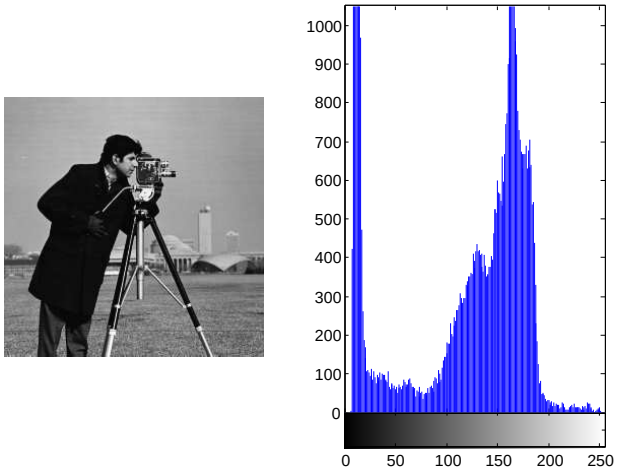


Figura 4.1: Imagen y su histograma

de brillo constante donde quiera que ocurran, tendrán histogramas diferentes. Un cambio de la iluminación general se trasladará a los picos en el histograma. Además, la mayoría de las imágenes reales presentan alguna variación de brillo dentro de las características o en regiones diferentes.

Desde el punto de vista del uso eficiente de los niveles de gris disponibles en la pantalla, algunos valores de la escala de grises están infrautilizados. Podría ser mejor extender el nivel de gris mostrado en las áreas de picos de forma selectiva, comprimiendo estos en valles, de modo que el mismo número de píxeles de la pantalla indiquen los niveles de brillo posibles. Esto se llama ecualización del histograma. La función de transferencia es simplemente el histograma de brillo original de la imagen, representado como un diagrama acumulativo.

La ecualización del histograma, figura 4.2, es una forma de manipulación de histograma que reduce automáticamente el contraste en las áreas muy claras o muy oscuras de una imagen. También expande los niveles de gris a lo largo de todo intervalo. Consiste en una transformación no lineal que considera la distribución acumulativa de la imagen original, para generar una imagen resultante cuyo histograma será aproximadamente uniforme. La opción de modificación, parte del principio que dice que el contraste de una imagen sería optimizado si todos los 256 niveles de intensidad posibles fueran igualmente utilizados o, en otras palabras, todas las barras verticales que componen el histograma fueran de la misma altura. Obviamente esto no es posible debido a la naturaleza discreta de los datos digitales de una imagen. Sin embargo, se consigue una aproximación al dispersar los picos del histograma de la imagen, dejando intactas las partes más bajas. Este proceso se obtiene a través de una función de transferencia que tiene una alta

Figura 4.2: Imagen original y ecualizada

inclinación siempre que el histograma original presenta un pico y una baja inclinación en el resto del histograma.

Una imagen con unas pocas regiones con valores de brillo muy similares presenta un histograma con picos. Los tamaños de estos picos darán el área relativa de las diferentes regiones y son útiles para el análisis de imágenes. Realizar una ecualización del histograma de la imagen extiende hacia afuera los picos, mientras que comprime otras partes del histograma mediante la asignación de valores de brillo iguales o muy cercanos a los píxeles que son muy pocos en número y tienen brillo intermedio. Esta ecualización permite ver pequeñas variaciones dentro de las regiones que parecían casi uniformes en la imagen original.

El proceso es bastante simple. Para cada nivel de brillo j de la imagen original (y su histograma), el nuevo valor asignado k se calcula como:

$$k = \sum_{i=0}^{j} \frac{N_i}{T} \tag{4.1}$$

donde la suma cuenta el número de píxeles de la imagen (mediante la integración del histograma), con un brillo igual o menor que j, y T es el número total de píxeles (o la superficie total del histograma).

Con la ecualización, algunos píxeles que originalmente tenían diferentes valores se asignan el mismo valor, lo que representa una pérdida de información, mientras que los valores que alguna vez fueron muy próximos entre sí se han extendido hacia fuera, dejando huecos en el histograma.

Este proceso de ecualización no es necesario que sea realizado en una imagen completa. La mejora de una porción de la imagen original, en lugar de toda la zona, también es útil en muchas situaciones. Esto es particularmente cierto cuando grandes regiones de la imagen corresponden a diferentes tipos de estructuras o escenas que en general son más brillantes o más oscuras que el resto de la imagen. Cuando partes o regiones de una imagen pueden ser seleccionadas, ya sea manualmente o mediante algún algoritmo basado en la variación del histograma, puede ser

utilizada una ecualización selectiva para resaltar detalles locales.

La ecualización del histograma de las regiones dentro de una imagen puede mejorar dramáticamente la visibilidad de los detalles locales, pero normalmente altera la relación entre el brillo y estructura. En la mayoría de los casos, es deseable que el nivel de brillo de los píxeles asociados a una característica particular de la imagen sea el mismo. Esto permitiría una rápida clasificación de las características para contarlas o medirlas. La modificación local de la escala de grises anula este supuesto, haciendo que el brillo sea dependiente de otras características que se encuentran cercanas o en la región seleccionada.

El proceso de ecualización local mejora el contraste cerca de los bordes, revelando detalles en las regiones de luz y en las regiones oscuras. Sin embargo, en el centro de las grandes regiones, normalmente uniformes, esto puede producir variaciones artificiales de contraste que introducen artefactos en la imagen. La solución es aumentar el tamaño de la región de manera que sea más grande que cualquier área uniforme en la imagen. Sin embargo, esto ralentiza el proceso y pierde la capacidad de aumentar el contraste local en los bordes.

4.2.3. Optimización espacial

Los procedimientos para la optimización espacial modifican el valor de un píxel basándose en los valores de los píxeles que son sus vecinos inmediatos. Estos procedimientos incluyen aplicar filtros espaciales para:

- Corregir y restaurar imágenes afectadas por el mal funcionamiento del sistema. Reducir o eliminar patrones de ruido.
- Mejorar las imágenes para su interpretación visual, por ejemplo, mejorar los detalles de bordes en imágenes.
- Extraer rasgos particulares.

Como todos los procedimientos de procesamiento de imágenes el objetivo es crear nuevas imágenes a partir de los datos de la imagen original, de tal manera que se incremente la cantidad de información que pueda ser interpretada.

Los filtros espaciales pueden clasificarse basándose en su linealidad en filtros lineales y en filtros no lineales. A su vez los filtros lineales pueden ser clasificados según las frecuencias que dejen pasar: los filtros paso bajo atenúan o eliminan las componentes de alta frecuencia a la vez que dejan inalteradas las bajas frecuencias; los filtros paso alto atenúan o eliminan las componentes de baja frecuencia con lo que agudizan las componentes de alta frecuencia. A continuación se describe el uso de estos filtros:

- Filtros paso bajo (Low pass filters). Estos filtros son utilizados en la reducción de ruido; suavizan y aplanan un poco las imágenes y como consecuencia se reduce o se pierde la nitidez. La aplicación de un filtro de paso bajo tiene el efecto de eliminar frecuencias altas

y medias dando como resultado una imagen que tiene un menor contraste, una apariencia más suave. Es por esto que este proceso es también denominado "suavización de imágenes" y al filtro de frecuencia baja se le llama filtro de suavizado o de homogeneización. Es muy fácil suavizar una imagen, el problema básico reside en que al hacerlo no se pierdan rasgos de interés. Por esta razón, el mayor énfasis a tener en cuenta en la aplicación de "filtros de baja frecuencia" es la preservación de los bordes.

- Filtros paso alto (High pass filters). En algunas ocasiones se observan cambios abruptos, en una imagen, desde un área con valores uniformes hacia otra con valores diferentes. Los límites de este tipo son conocidos como aristas o bordes (edges). Estos bordes ocupan un área pequeña y son por lo tanto rasgos de alta frecuencia. Los filtros de paso alto están diseñados para resaltar frecuencias altas y para suprimir frecuencias bajas. Estos filtros son utilizados para detectar cambios de luminosidad. Son usados para la detección de patrones como bordes o para resaltar detalles finos de una imagen. Es por esto que los filtros de alta frecuencia son también llamados filtros de optimización de bordes.

  Dentro de los filtros de paso alto se distinguen dos clases de filtros: filtros de gradiente o direccionales y filtros laplacianos o no-direccionales. Los filtros de gradiente son filtros direccionales y se emplean para mejoramientos específicos de tendencias lineales. Se diseñan de tal manera que resaltan los objetos lineales o bordes orientados en una cierta dirección (horizontal, vertical o diagonal). En su forma más simple, los filtros calculan la diferencia que existe entre un píxel y la de sus vecinos. Matemáticamente pueden verse como el resultado de tomar la primera derivada. Los filtros Laplacianos no son direccionales porque resaltan rasgos lineales sin importar la dirección que tengan en la imagen. Estos filtros no consideran el propio gradiente, sino los cambios del gradiente. En su forma más simple, pueden verse como el resultado de tomar la segunda derivada.

4.2.3.1. Detección de bordes

La detección de bordes es una área importante en el campo de Visión por Computador. Los bordes se definen como la variación significante en los niveles de gris en alguna región de una imagen [Efford00]. Con ello se pueden encontrar sombras en una imagen u otro cambio distinto en la intensidad de una imagen. Los detectores de bordes son fundamentales en el procesamiento a bajo nivel de imagen y los bordes son necesarios para el procesamiento de alto nivel [Ahmad99].

La detección de bordes es la aproximación mas común para la detección de discontinuidades significativas en niveles de gris [Gose96]. Los algoritmos de detección de bordes localizan y acentúan bordes. El principal propósito de detectar bordes es la segmentación de escenas para la identificación de objetos en una imagen. La calidad de la detección de bordes es muy dependiente de las condiciones de iluminación, presencia de objetos de intensidades similares, la densidad de los bordes de la escena, y el ruido. Debido a que diferentes detectores de bordes funcionan mejor en diferentes condiciones, lo ideal sería tener un algoritmo que hace uso de detectores de bordes múltiples, y aplicar cada uno de ellos cuando las condiciones de la escena son las más idóneas

para su método de detección. La delineación de bordes es usada en medicina, vigilancia, análisis de muestras, y, en general para extraer las principales componentes geométricas presentes en la escena.

Un detector de bordes acepta imágenes como entrada y produce un mapa de bordes como salida. El mapa de bordes de algunos detectores incluye información sobre la posición, intensidad y orientación de los bordes.

Desde el punto de vista de la integración de un detector de bordes en un sistema de visión por computador existen dos clases de detectores. El primero incluye detectores que no usan conocimiento *a priori* sobre la escena y el borde a detectar. Esta clase de detectores no está influenciada por otros componentes del sistema de visión, ni por información contextual. Estos detectores son sensibles en el sentido de que no están limitados a imágenes específicas. La segunda clase de detectores son contextuales, son guiados por los resultados de otros componentes del sistema o por conocimiento *a priori* sobre el borde o la estructura de la escena.

La mayoría de esquemas propuestos para detección de bordes (de ambos tipos) incluyen tres operaciones: suavizado, localización de variaciones de gris y etiquetado:

- Suavizado de la imagen. El suavizado sobre una imagen tiene un efecto positivo, reduciendo el ruido y asegurando una detección de bordes robusta. Como efecto negativo produce perdida de información. Es evidente que existe un compromiso entre perdida de información y reducción de ruido. El objetivo final es encontrar detectores óptimos que aseguren un compromiso favorable entre la reducción de ruido y la conservación de los bordes.

- Localización de variaciones de gris. El propósito de la detección de bordes es localizar las variaciones del nivel de gris de la imagen e identificar los fenómenos físicos que los producen. La operación mas habitual para localizar estas variaciones es la diferenciación, que es el cálculo de las derivadas necesarias para localizar los bordes. El operador de diferenciación se caracteriza por su orden, su invariancia a la rotación y su linealidad.

- Etiquetado de Bordes. El etiquetado de bordes es la ultima operación en realizarse. Consiste en la localización de los bordes y el aumento de la relación señal-ruido mediante la supresión de bordes falsos. El procedimiento de localización depende del operador de diferenciación utilizado.

La especificación de un detector de bordes en término de estas tres operaciones está incompleta. De hecho, un detector de bordes no incluye el contexto preciso en el que puede ser utilizado con éxito. Por lo tanto, es necesario definir una metodología de detección de bordes para seleccionar un detector de bordes adecuado para una aplicación.

Algoritmos de detección de bordes

En el área de procesamiento de imágenes, la detección de los bordes de una imagen es de suma importancia y utilidad, pues facilita muchas tareas, entre ellas, el reconocimiento de objetos, la

segmentación de regiones, entre otras. Se han desarrollado variedad de algoritmos que ayudan a solucionar este inconveniente. En esta sección se muestran algunos de los algoritmos de detección de bordes más utilizados en el área de procesamiento de imágenes. En el último epígrafe de esta sección se exponen los operadores más usados para detectar la dirección del borde.

1. **Método Canny**

   El detector de bordes *Canny* es ampliamente considerado como un estándar en la detección de bordes. Fue desarrollado por John Canny en 1986 [Canny86], y todavía obtiene mejores resultados que los nuevos algoritmos que han sido desarrollados. Uno de los métodos relacionados con la detección de bordes es el uso de la primera derivada, la que es usada porque toma el valor de cero en todas las regiones donde no varía la intensidad y tiene un valor constante en toda la transición de intensidad. Por tanto un cambio de intensidad se manifiesta como un cambio brusco en la primera derivada [Gonzalo01], característica que es usada para detectar un borde, y en la que se basa el algoritmo de Canny. El algoritmo de Canny consiste en tres grandes pasos:

   a) Obtención del gradiente. En este paso se calcula la magnitud y orientación del vector gradiente en cada píxel. Para la obtención del gradiente, lo primero que se realiza es la aplicación de un filtro gaussiano a la imagen original con el objetivo de suavizar la imagen y tratar de eliminar el posible ruido existente. Sin embargo, se debe de tener cuidado de no realizar un suavizado excesivo, pues se podrían perder detalles de la imagen y provocar un pésimo resultado final. Este suavizado se obtiene promediando los valores de intensidad de los píxels en el entorno de vecindad con una mascara de convolución de media cero y desviación estándar σ. Una vez que se suaviza la imagen, para cada píxel se obtiene la magnitud y módulo (orientación) del gradiente, obteniendo así dos imágenes.

   b) Supresión no máxima. En este paso se logra el adelgazamiento del ancho de los bordes, obtenidos con el gradiente, hasta lograr bordes de un píxel de ancho. Las dos imágenes generadas en el paso anterior sirven de entrada para generar una imagen con los bordes adelgazados. El procedimiento es el siguiente: se consideran cuatro direcciones identificadas por las orientaciones de 0^o, 45^o, 90^o y 135^o con respecto al eje horizontal. Para cada píxel se encuentra la dirección que mejor se aproxime a la dirección del ángulo de gradiente. Posteriormente se observa si el valor de la magnitud de gradiente es más pequeño que al menos uno de sus dos vecinos en la dirección del ángulo obtenida en el paso anterior. De ser así se asigna el valor 0 a dicho píxel, en caso contrario se asigna el valor que tenga la magnitud del gradiente. La salida de este segundo paso es una imagen con los bordes adelgazados.

   c) Histéresis de umbral. En este paso se aplica una función de histéresis basada en dos umbrales; con este proceso se pretende reducir la posibilidad de aparición de contornos falsos. La imagen obtenida en el paso anterior suele contener máximos locales creados

por el ruido. Una solución para eliminar dicho ruido es la histéresis del umbral. El proceso consiste en tomar la imagen obtenida del paso anterior, tomar la orientación de los puntos de borde de la imagen y tomar dos umbrales, el primero más pequeño que el segundo. Para cada punto de la imagen se debe localizar el siguiente punto de borde no explorado que sea mayor al segundo umbral. A partir de dicho punto seguir las cadenas de máximos locales conectados en ambas direcciones perpendiculares a la normal del borde siempre que sean mayores al primer umbral. Así se marcan todos los puntos explorados y se almacena la lista de todos los puntos en el contorno conectado. Es así como en este paso se logra eliminar las uniones en forma de Y de los segmentos que confluyan en un punto.

Al aplicar el algoritmo de Canny es muy común realizar un cuarto paso. Este paso consiste en cerrar los contornos que pudiesen haber quedado abiertos por problemas de ruido. Un método muy utilizado es el algoritmo de Deriche y Cocquerez [Shams08]. Este algoritmo utiliza como entrada una imagen binarizada de contornos de un píxel de ancho. El algoritmo busca los extremos de los contornos abiertos y sigue la dirección del máximo gradiente hasta cerrarlos con otro extremo abierto.

2. **Método Zero-cross**

   La técnica de detección de bordes *zero-cross* está basada en el cálculo de la segunda derivada [Marr91]. Explora el hecho de que un borde corresponde a un cambio abrupto en la imagen. La primera derivada de la imagen debe tener un valor extremo en la posición correspondiente a un borde de la imagen, por lo que la segunda derivada debe ser cero en la misma posición; sin embargo, es mucho más fácil y más preciso encontrar una posición de *zerocrossing* que un extremo.

   La cuestión crucial es cómo calcular la segunda derivada con robustez. Una posibilidad es aplicar un suavizado a la imagen (para reducir el ruido) y a continuación calcular las segundas derivadas. Al elegir un filtro de suavizado, hay dos criterios que deben cumplirse. En primer lugar, el filtro debe ser suave y la banda limitada al dominio de la frecuencia para reducir el posible número de frecuencias en las que cambia la función. En segundo lugar, la restricción de localización espacial requiere la aplicación del filtro a puntos cercanos entre sí. Estos dos criterios son contradictorios, pero se pueden optimizar simultáneamente usando una distribución Gaussiana.

3. **Método Susan**

   Algoritmo de detección de bordes cuyas siglas responden al nombre *Smallest Univalue Segment Assimilating Nucleus* [Smith97]. La identificación de bordes se realiza mediante el filtrado de la imagen original a través de una máscara que calcula un nuevo valor del píxel tomado como centro dependiendo de su semejanza o no con sus píxeles vecinos. Posteriormente el algoritmo se encarga de etiquetar los píxeles en dos categorías (como borde o

como fondo), dependiendo de si superan o no cierto umbral. Los píxeles con valores por encima del umbral (cuyo nivel en la imagen original difiere del de sus vecinos) son considerados como borde, mientras que los que no superan el umbral (cuyo nivel en la imagen original es semejante al de sus vecinos) serían etiquetados como fondo. Posteriormente, y si es necesario, se aplica un filtrado para eliminar niveles intermedios de gris para finalizar con el adelgazamiento (identificación de un borde de grosor unidad) de la imagen en blanco y negro resultante.

4. **Operadores para la detección de bordes**

Los operadores de detección de bordes permiten, mediante el cálculo de primeras (gradiente) y segundas derivadas (laplaciana), determinar puntos de principal importancia para determinar objetos de interés. En el primer caso se buscarán grandes picos y en el segundo, pasos de positiva a negativa o viceversa (cruces por cero) [Escalera01].

Los operadores en el dominio espacial operan directamente sobre los pixels de la imagen. Operan en la vecindad de los pixels, generalmente mediante una máscara cuadrada o rectangular. Una máscara de convolución h es una aproximación numérica de la operación diferenciación. Las dimensiones de ésta máscara son normalmente impares e iguales, de forma que se pueda determinar un centro de una matriz cuadrada. La posición del valor central se corresponde con la posición del píxel de salida. Las máscaras de convolución usadas en detección de bordes tienen coeficientes positivos y negativos y suman un total de 0. A continuación se describen diversos operadores del dominio espacial y sus máscaras de convolución:

- Filtros basados en el gradiente o direccionables.
  - Operador Roberts. Es uno de los operadores más antiguos. Es muy fácil de calcular ya que usa sólo una matriz de 2 x 2 para cada píxel. Sus máscaras de convolución son:

$$h1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \qquad h2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{4.2}$$

    La principal desventaja de este operador es que es muy sensible al ruido, porque muy pocos píxels son usados para aproximar el gradiente.
  - Operador Sobel.
    Operador matricial basado en el operador gradiente [Kittler83], [Lee83]. Se realiza un filtrado de la imagen original mediante el que se determinará si el píxel es o no de borde, tras un proceso de umbralización. Es usado como detector simple de bordes horizontales y verticales, en cuyo caso sólo son usadas las máscaras h_1 y

h_3. Pone especial énfasis en píxeles cercanos al centro de la máscara.

$$h1 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, \quad h2 = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}, \quad h3 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, ... \tag{4.3}$$

- Operador Prewitt.
  Este operador es similar al de Sobel y a otros operadores que aproximan a la primera derivada. Identifica por separado los bordes verticales y horizontales al filtrar la imagen con dos máscaras que la resaltan en las direcciones anteriormente indicadas. El gradiente es estimado en ocho posibles direcciones, y el resultado de convolución de mayor magnitud indica la dirección del gradiente. Este operador no otorga una importancia especial a píxeles cercanos al centro de la máscara. Los operadores de aproximación de la primera derivada de una función de la imagen a veces se llaman compass operators (operadores brújula) debido a su capacidad para determinar la dirección del gradiente. Similar a este operador se encuentran los dos operadores siguientes, Robinson y Kirsch. De estos operadores, se presentan las tres primeras máscaras de 3 x 3, el resto pueden ser creadas por rotación.

$$h1 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}, \quad h2 = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}, \quad h3 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} ... \tag{4.4}$$

- Operador Robinson.

$$h1 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix}, \quad h2 = \begin{bmatrix} 1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & -1 & 1 \end{bmatrix}, \quad h3 = \begin{bmatrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{bmatrix} ... \tag{4.5}$$

- Operador Kirsch.

$$h1 = \begin{bmatrix} 3 & 3 & 3 \\ 3 & 0 & 3 \\ -5 & -5 & -5 \end{bmatrix}, \quad h2 = \begin{bmatrix} 3 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & -5 & 3 \end{bmatrix}, \quad h3 = \begin{bmatrix} -5 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & 3 & 3 \end{bmatrix} ... \tag{4.6}$$

- Filtros Laplacianos o no direccionales.

  - Operador de Laplace. El operador de Laplace es un operador muy popular aproximando la segunda derivada, la cual ofrece la magnitud del gradiente. A menudo

es usada una máscara h de 3 x 3; para 4 y 8 vecinos es definida como:

$$h = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \qquad h = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{4.7}$$

El operador laplaciano es utilizado, a menudo, con el píxel central o en sus alrededores enfatizado. En esta aproximación se pierde la invariancia a la rotación. Con este operador existe la desventaja de que algunos bordes son detectados de forma repetida.

$$h = \begin{bmatrix} 2 & -1 & 2 \\ -1 & -4 & -1 \\ 2 & -1 & 2 \end{bmatrix}, \qquad h = \begin{bmatrix} -1 & 2 & -1 \\ 2 & -4 & 2 \\ -1 & 2 & -1 \end{bmatrix} \tag{4.8}$$

4.3. Segmentación

Uno de los pasos más utilizados en el proceso de reducción de imágenes a información es la segmentación. La segmentación es un proceso que consiste en dividir una imagen en regiones homogéneas con respecto a una o más características con el fin de facilitar un posterior análisis, distinguir objetos de interés o reconocimiento automático. La segmentación se describe, a menudo, por analogía a los procesos visuales como la separación frente/fondo, lo que implica que el procedimiento de selección se concentra en un sólo tipo de función y descarta el resto.

La segmentación debe verse como un proceso que a partir de una imagen, produce otra en la que cada píxel tiene asociada una etiqueta distintiva del objeto al que pertenece. Así, una vez segmentada una imagen, se podría formar una lista de objetos consistentes en las agrupaciones de los píxeles que tengan la misma etiqueta. Se consideran dos tipos de segmentaciones: segmentación completa de la escena o imagen y segmentación parcial. La segmentación completa obtiene regiones disjuntas que corresponden, únicamente, con objetos de la imagen de entrada. Suele necesitar cooperación con procesos de alto nivel que usen conocimiento específico del dominio. En la segmentación parcial, las regiones que se obtienen no corresponden directamente con objetos de la imagen. La imagen se divide en regiones que son homogéneas respecto a una determinada propiedad como color, reflectancia, textura, etc. En imágenes complejas puede ser necesario obtener regiones homogéneas con solapamiento. La segmentación final se realiza con la ayuda de información de alto nivel.

Dentro de una misma imagen pueden realizarse diferentes segmentaciones. En general, el proceso de la segmentación suele resultar complejo debido, por un lado, a que no se tiene una información adecuada de los objetos a extraer y, por otro, a que en la escena

a segmentar aparece normalmente ruido. Es por esto que el uso de conocimiento sobre el tipo de imagen a segmentar o alguna otra información de alto nivel puede resultar muy útil para conseguir la segmentación de la imagen.

A continuación, se expone uno de los enfoques para realizar el proceso de segmentación: la segmentación basada en umbralización, que clasifica las distintas partes de la imagen en puntos de objeto y puntos de fondo, mediante el cálculo de un valor límite que realizará dicha separación o binarización.

4.3.1. Segmentación basada en umbralización

Muchos objetos o regiones de una imagen están caracterizadas por su reflexión, absorción de la luz en su superficie o brillo. Estas características son usadas para obtener el umbral de una imagen que permita diferenciar objetos del fondo. La segmentación basada en umbralización, es usada para identificar y separar objetos del fondo, en base a la distribución de los niveles de gris o la textura de los objetos en las imágenes.

La segmentación basada en umbralización es una técnica de segmentación rápida, que tiene un coste computacional bajo y que puede ser realizada en tiempo real durante la captura de la imagen. Por otro lado, muchas técnicas de segmentación se basan en la estadística del histograma de los niveles de gris o en la matriz de co-ocurrencia de una imagen [Liao01]. El histograma de una imagen no tiene en cuenta la información espacial sino solamente la distribución de grises en la imagen. Por ello, dos imágenes muy diferentes pueden tener el mismo histograma. Esto hace que, los métodos de segmentación basados en umbralización, como único medio de segmentación, resulten limitados en muchos problemas reales. Aunque sí se usan con frecuencia como complemento de otros métodos.

4.3.1.1. Umbralización fija

El caso más sencillo, conocido como umbralización fija, se puede usar en aquellas imágenes en las que existe suficiente contraste entre los diferentes objetos que se desea separar. Consiste en establecer un valor fijo sobre el histograma que marque el umbral de separación. Para obtener dicho umbral se debe disponer de información sobre los niveles de intensidad de los objetos a segmentar y del fondo de la imagen. De esta forma, la imagen binaria resultante $B(i,j)$ se define a partir de la imagen original $I(i,j)$ en función de un valor U que corresponde al umbral de separación seleccionado según la siguiente ecuación:

$$B(i,j) = \begin{cases} 1, si I(i,j) \geq U \\ 0, si I(i,j) < U \end{cases} \tag{4.9}$$

La elección de un valor de umbral correcto resulta decisivo para llevar acabo la segmentación de una imagen de manera satisfactoria. La obtención del umbral suele basarse en el

histograma de la imagen. Cuando no hay razón para creer que el fondo y el objeto ocupan áreas similares en la imagen un buen valor inicial de U es el nivel promedio de gris de la imagen. Si en el histograma se aprecian uno o más lóbulos, éstos suelen corresponder con una o varias zonas de la imagen, que comparten niveles de intensidad similares. Estos objetos pueden ser directamente los objetos a segmentar o corresponder a partes homogéneas de objetos más complejos. Lógicamente, la transición de un lóbulo a otro se corresponde con un mínimo del histograma, correspondiendo estos mínimos a los puntos que fijan el valor umbral. La búsqueda de dichos mínimos (basada por ejemplo en el cálculo de derivadas) se encuentra dificultada por la naturaleza ruidosa del histograma. Para atenuar este problema puede aplicarse un filtro paso bajo sobre el histograma de la imagen.

4.3.1.2. Umbralización generalizada

En general, la obtención de un único valor de umbral fijo no es útil en imágenes complejas. Por ejemplo, sobre imágenes de documentos con fondos complejos o sobre escenas con iluminación no uniforme, el estudio del histograma de la imagen puede revelar la inexistencia de un único valor umbral que permita separar los objetos del fondo. Esto lleva a considerar otros tipos de umbralización que resultan de generalizar la idea de umbral. A continuación, se definen la umbralización de banda, la multiumbralización, la semiumbralización y la umbralización adaptativa.

- **Umbralización de banda**

  La umbralización de banda permite segmentar una imagen en la que los objetos (regiones de píxeles) contienen niveles de gris dentro de un rango de valores y el fondo tiene píxeles con valores en otro rango disjunto. Así

$$B(i,j) = \begin{cases} 1, si I(i,j) \in R \\ 0, en otro caso \end{cases} \tag{4.10}$$

  donde R representa un rango de valores correspondientes a niveles de gris que definen a los elementos a extraer de la imagen digital

- **Multiumbralización**

  La multiumbralización, como su nombre indica, consiste en la elección de múltiples valores de umbral dentro del proceso, permitiendo separar a diferentes objetos dentro de una escena cuyos niveles de gris difieran. El resultado no será ahora una imagen binaria sino que los diferentes objetos (regiones) tendrán etiquetas diferentes.

$$I_s(i,j)\begin{cases} = 1, si I(i,j) \in R_1 \\ = 2, si I(i,j) \in R_2 \\ ..... \\ = n, si I(i,j) \in R_n \\ = 0, en otro caso \end{cases} \tag{4.11}$$

donde $I(i,j)$ es la imagen original, $I_s(i,j)$ es la imagen segmentada y R_1, R_2,...,R_n representan los n diferentes rangos de niveles de gris usados para umbralizar.

- **Semiumbralización**

  La semiumbralización persigue obtener una imagen resultado en niveles de gris, y para ello pone a cero el fondo de la imagen conservando los niveles de gris de los objetos a segmentar que aparecen en la imagen inicial. Su fórmula es:

$$I_s(i,j) = \begin{cases} I(i,j) si I(i,j) \geq U \\ 0 en otro caso \end{cases} \tag{4.12}$$

- **Umbralización adaptativa**

  En las umbralizaciones anteriores, los rangos de umbralización se consideran fijos con independencia de las características locales de la imagen considerada. En muchas imágenes, donde la iluminación no es uniforme, puede ocurrir que píxeles del mismo objeto a segmentar tengan niveles de gris muy diferentes. Ello conlleva que no sea posible elegir un único umbral que, sobre toda la imagen, distinga los píxeles de un objeto de los de otro. La umbralización adaptativa o variable permite resolver este problema haciendo que el valor del umbral varíe según una función que depende de las características locales del entorno del punto que se evalúa.

Métodos de determinación de umbrales

La selección del umbral, para una imagen, puede ser realizada mediante procedimientos parámetricos o no paramétricos. En los procedimientos paramétricos, la distribución de los niveles de gris de un objeto permite obtener el umbral. Estudios como los descritos en [Tsai95] y [Wang84], son un ejemplo de este tipo de procedimiento.

En los procedimientos no parámetricos, los umbrales se obtienen de una forma óptima, de acuerdo a algún criterio. Otsu [Otsu79] presentó un método no paramétrico y no supervisado, para la selección automática del umbral para segmentación de imágenes, propuesto para imágenes en escala de gris. El algoritmo de Otsu representa a los algoritmos basados en la obtención de una función criterio a partir de estadísticos extraídos del histograma, y la búsqueda del umbral óptimo que haga máxima o mínima, según el caso, dicha función criterio. El umbral óptimo maximiza la varianza entre clases, se selecciona mediante el método de discriminantes que maximiza la separabilidad de las clases resultantes en niveles

de gris. Usa los momentos acumulativos de orden cero y uno del histograma de niveles de grises.

El método de Otsu fue extendido en [Cheriet98], proponiendo un método recursivo para la obtención de umbrales. Este enfoque segmenta en cada ciclo el objeto homogéneo más brillante de una imagen dada, dejando sólo los objetos homogéneos más oscuros después de la última recursión. Este método fue posteriormente usado como base para el análisis de imágenes en color. Este método ha sido usado [Nimbarte10] como base, comparando el método de Otsu con el MCVT (Minimum Class Variance Thresholding), obteniendo como conclusión que Otsu sigue siendo más potente para una amplia gama de imágenes.

La selección del umbral basada en la entropía, fue propuesta en [Pun80] y [Pun81] y luego desarrollada en varios artículos [Luthon04], [Chang06]. Este método maximiza a priori una función de evaluación, la cual es una entropía, determinada a posteriori. El criterio de entropía y entropía relativa (también conocida como distancia de información de Kullback-Leibler) fue ampliamente desarrollado por varios autores [Chang06]. La entropía mide la incertidumbre de una fuente de información con un umbral óptimo, que se obtiene maximizando la entropía de Shannon; mientras que la última, mide la discrepancia de información entre dos fuentes con un umbral obtenido minimizando la entropía relativa. Ambos métodos (Otsu y entropía de Shannon) son considerados métodos clásicos de testeo de imágenes y también son métodos base para desarrollo de otros.

4.4. Procesamiento morfológico

4.4.1. Introducción

Las tareas de segmentación no suelen dar un resultado exacto de la delimitación de los objetos o regiones de interés. Aparecen píxeles mal clasificados, bordes imprecisos de los objetos o regiones que están solapadas. Para corregir estos problemas se puede usar el procesamiento morfológico, que es una técnica de procesamiento no lineal de la señal, caracterizada en realzar la geometría y forma de los objetos. Su fundamento matemático se basa en la teoría de conjuntos.

Los operadores morfológicos son muy útiles ya que permiten realizar funciones complejas a partir de la combinación de sus operadores. De este modo, se pueden realizar diferentes análisis en imágenes binarias, los cuales son utilizados comúnmente para la reducción del ruido, realce de la imagen, detección de bordes, etc. Las dos operaciones básicas en la matemática morfológica son la erosión y la dilatación. Estos operadores toman dos entradas, la imagen a erosionar o a dilatar y su estructura (elemento estructurante). Es importante notar que la erosión y la dilatación no son inversas, de hecho dilatar y luego erosionar raramente se transforma en la misma imagen. A partir de ellas podemos componer las operaciones de apertura y cierre.

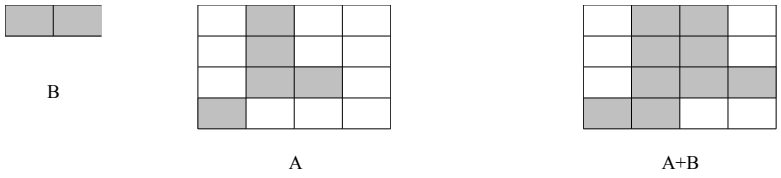


Figura 4.3: Ejemplo de dilatación

4.4.2. Operaciones morfológicas

La clase más amplia de las operaciones de procesamiento de imágenes binarias a veces se describen colectivamente como operaciones morfológicas. Estas incluyen la erosión y la dilatación, y las modificaciones y combinaciones de estas operaciones. Dado que los valores de píxeles en las imágenes binarias están restringidos a 0 ó 1, las operaciones son más sencillas. Las operaciones se pueden describir simplemente en términos de adición o eliminación de los píxeles de la imagen binaria de acuerdo a ciertas reglas, que dependen del patrón de los píxeles vecinos. Cada operación se realiza en cada píxel de la imagen original utilizando el modelo original de píxeles, es decir, ninguno de los nuevos valores de los píxeles se utilizan para evaluar el patrón de píxeles vecinos.

4.4.2.1. Dilatación

Este operador es comúnmente conocido como "relleno", "expansión" o "crecimiento". Puede ser usado para rellenar "huecos" de tamaño igual o menor que el elemento estructurante con el que se opera la dilatación. Usado con imágenes binarias, donde cada píxel es 0 ó 1, la dilatación es similar a la convolución. Sobre cada píxel de la imagen se superpone el origen del elemento estructurante. Si el píxel de la imagen no es cero, cada píxel que cae en la estructura es añadido al resultado aplicando el operador 'or', como se observa en la figura 4.3. Usado con imágenes en escala de grises, la dilatación se efectúa tomando el máximo de una serie de sumas. Puede ser usado para implementar el operador de "máxima vecindad" con la forma de la vecindad dada en el elemento estructurante. El efecto básico del operador en una imagen binaria es ampliar gradualmente los límites de las regiones de píxeles de primer plano. Así, las áreas de píxeles en primer plano crecen en tamaño, mientras que los agujeros dentro de esas regiones se hacen más pequeños, como se observa figura 4.4.

Formalmente la dilatación de $A \subseteq X$ y $B \subseteq X$, se nota por $A \oplus B$ y se define mediante:

$$A \oplus B = \{a + b | a \in A \wedge b \in B\} \tag{4.13}$$

Figura 4.4: Dilatación

En la práctica los conjuntos A y B no son simétricos. El primer elemento, A, está asociado con la imagen que se está procesando y el segundo es el elemento estructural, que es la forma que actúa sobre A en la dilatación para producir $A \oplus B$. Esta operación representa un crecimiento progresivo del conjunto A. Al pasar el elemento estructurante dentro del conjunto, éste no se modificará. Sin embargo, en la frontera del conjunto A, al desplazar a B, el conjunto resultado se expansionará, como se observa en la figura 4.4.

4.4.2.2. Erosión

La erosión, junto con la dilatación, es una de las operaciones básicas en el área de procesamiento morfológico. El efecto básico del operador en una imagen binaria es erosionar las fronteras de las regiones de píxeles de primer plano. Así, las áreas de píxeles en primer plano disminuyen de tamaño, y los agujeros dentro de esas áreas se hacen más grandes. Los efectos son de "encogimiento", "contracción", o "reducción". Puede ser utilizado para eliminar islas menores en tamaño que el elemento estructurante. Sobre cada píxel de la imagen se superpone el origen del elemento estructurante. Si cada elemento no cero de dicho elemento está contenido en la imagen, entonces el píxel de salida es puesto a 1, figura 4.5. Al igual que sucede en la dilatación, el tamaño y la forma finales del conjunto erosionado dependerá fuertemente del tamaño y forma del elemento estructurante.

Como se observa en la figura 4.6, con el operador erosión, las regiones claras, rodeadas de regiones oscuras se encogen, por el contrario, las regiones oscuras rodeadas de regiones claras crecen en tamaño. Los píxeles claros rodeados de zonas oscuras desaparecen, esto permite eliminar la mayoría del ruido anexado en la imagen, en cambio los píxeles oscuros rodeados de zonas claras, crecen.

Formalmente la erosión de $A \subseteq X$ y $B \subseteq X$, se nota por $A \ominus B$ y se define mediante:

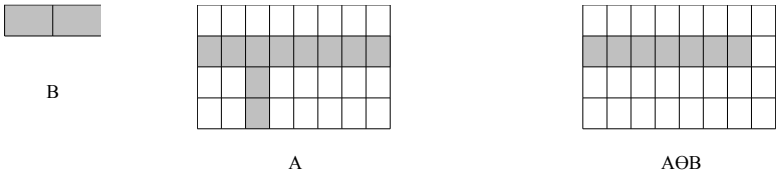


Figura 4.5: Ejemplo de erosión

Figura 4.6: Erosión

$$A \ominus B = \{x \in X | x + b \in A, \forall b \in B\} \tag{4.14}$$

4.4.2.3. Dualidad erosión-dilatación

La dilatación y la erosión exhiben la propiedad de dualidad, con la particularidad de que entran en juego el complemento del conjunto operando y el conjunto reflejado del elemento estructurante. Sean A y B subconjuntos de X, entonces se cumple que:

$$(A \ominus B)^c = A^c \oplus B^-$$

La acción de erosionar un conjunto A por un elemento estructurante B dado es en cierto sentido equivalente a dilatar el complemento del conjunto A con el reflejado del elemento de estructura B. Recíprocamente, dilatar un conjunto A por un elemento de estructura B es equivalente a erosionar su complemento (A^c) con el reflejado del elemento estructurante B.

4.4.2.4. Apertura y Cierre

A pesar de que la dilatación y la erosión son operaciones independientes, dentro de la morfología, en un gran porcentaje de procesos de análisis de imágenes, se usan por parejas alternadas; es decir, normalmente se realizan procesos del tipo: erosión seguida de una dilatación y una dilatación seguida de una erosión. En un principio podría pensarse que en ambos procesos se obtendrá el mismo resultado, pero no es así, la erosión y la dilatación no son operaciones inversas, es por ello que a estos procesos se les conoce con el nombre de apertura y cierre, respectivamente, y de hecho se consideran como operaciones fundamentales. La definición de apertura y cierre se expresa en términos de dilatación y erosión.

La apertura de un conjunto $A \subseteq X$ por el elemento de estructura $B \subseteq X$ se denota como $A \circ B$ y se define así:

$$A \circ B = (A \ominus B) \oplus B$$

Como se observa en la figura 4.7, los efectos de la apertura en una región de interés se pueden resumir en tres aspectos:

a) Se eliminan islas de tamaño menor al elemento estructurante.

b) Se eliminan picos o cabos más delgados que el elemento estructurante.

c) Se rompen istmos cuya anchura sea menor al diámetro de elemento estructurante.

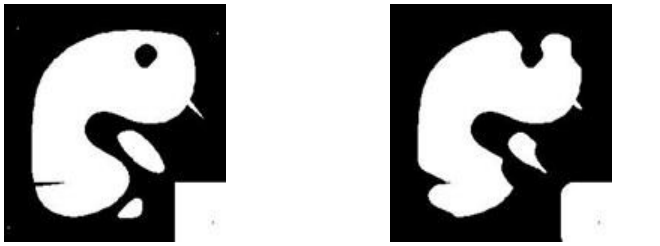


Imagen Original Imagen tras operación de Apertura

Figura 4.7: Apertura

Imagen Original Imagen tras operación de Cierre

Figura 4.8: Cierre

El cierre de un conjunto $A \subseteq X$ por el elemento de estructura $B \subseteq X$ se denota como $A \bullet B$ y se define así:

$$A \bullet B = (A \oplus B) \ominus B$$

El cierre produce efectos diferentes a los que produce la apertura. Según la figura 4.8 y la siguiente lista, el cierre produce los siguientes efectos:

a) Se rellenan los lagos o huecos de tamaño menor al elemento de estructura.

b) Se rellenan rajaduras o golfos más delgados que el elemento estructurante.

c) Se funden estrechos cuya anchura sea menor al diámetro del elemento estructura.

Hay muchas aplicaciones para las operaciones de erosión, dilatación y apertura y cierre. Estas cuatro operaciones definen prácticamente todos los filtros y procesos de la morfología matemática.

4.5. Resumen y Conclusiones

En este capítulo se ha realizado un repaso de distintas técnicas de análisis de imágenes. El realce de imágenes, permite aumentar la visibilidad de elementos de una imagen, permitiendo interpretar los diferentes objetos de la imagen. Con la manipulación del contraste, es posible mostrar mejor las características de mayor interés. La ecualización del histograma permite observar pequeñas variaciones en la imagen. La detección de bordes, una de las técnicas más empleadas en Visión por Computador, se basa en detectar discontinuidades significativas en niveles de gris para mostrar diferentes objetos en una imagen. La segmentación divide una imagen en regiones homogéneas con respecto a una o varias caracteristicas. Por último, el procesamiento morfológico permiten dar un resultado exacto en la delimitación de objetos o regiones de interés.

Estas técnicas pueden aportar soluciones al problema de procesamiento de imágenes planteado en esta tesis, permitiendo distinguir e identificar objetos. La segmentación basada en umbralización y el procesamiento morfológico, concretamente sus dos principales operaciones, dilatación y erosión, permiten pre-procesar imágenes para reducción de ruido o su realce. Las técnicas de detección de bordes localizan y remarcan cambios en la intensidad de las imágenes. Cada una de estas técnicas anteriormente descritas, serán utilizadas durante el desarrollo de esta tesis, en el capítulo 6 denominado “Detección de defectos en tableros de fibra”.

Parte III

Desarrollo y metodología de la investigación

Capítulo 5

Planificación de Tareas en Sistemas Grid: Planificador Local Balanceado y Meta-Planificador de Tareas con Sistemas Expertos basado en Reglas Borrosas

5.1. Introducción

Grid Computing es una tecnología reciente que comparte recursos heterogéneos localizados en distintos puntos de la geografía [Foster04a]. El uso de estos recursos, conectados con redes de alta velocidad, permite resolver problemas con gran coste computacional [Foster01]. Como se muestra en la figura 5.1, un Grid es un sistema complejo y altamente heterogéneo que utiliza recursos de distinta naturaleza y organizaciones o Nodos Grid (GN) para resolver un objetivo común entre las organizaciones formantes del sistema.

Grid Computing implica nuevos retos. Uno de estos retos implica la adaptación de aplicaciones y planificadores paralelos, desarrollados para recursos homogéneos, a los recursos dinámicos y heterogéneos del Grid, con una interferencia mínima en la planificación de recursos o en el balanceo de carga. Los sistemas Grid tienen una serie de características, como la heterogeneidad, autonomía, escalabilidad y adaptabilidad, que hacen que el problema de balanceo de carga sea aun más difícil. Un algoritmo de planificación tradicional intenta equilibrar la carga de los recursos para mejorar el tiempo de finalización de las tareas velando por la máxima utilización de los recursos disponibles.

Un sistema Grid está formando por múltiples recursos de muy diversos ámbitos. Para

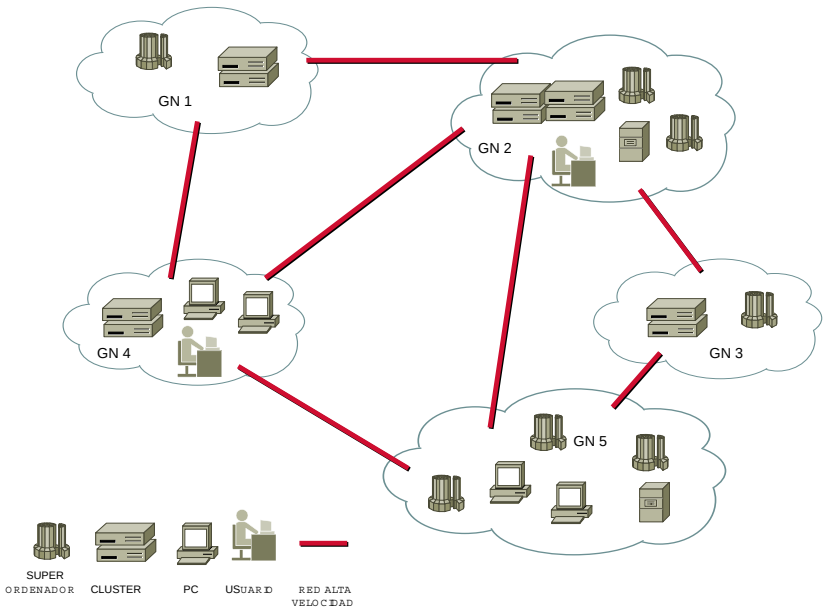


Figura 5.1: Ejemplo de Sistema Grid

realizar una buena planificación de tareas y obtener un tiempo de ejecución mínimo es necesario conocer el estado actual de estos recursos. La información sobre el estado de los recursos de un Grid es muy importante para cualquier planificador, especialmente en un sistema Grid, debido a la naturaleza dinámica y heterogénea de sus componentes.

En el tratamiento de información sujeta a imprecisiones es importante destacar el papel de la lógica borrosa. La lógica borrosa se fundamenta en la idea básica de que el razonamiento humano es aproximado por naturaleza y provee de una metodología para la representación del conocimiento borroso en condiciones donde, debe tolerarse un cierto grado de imprecisión en la información [Cordon01]. Una de las mayores ventajas de estos sistemas viene dada por su capacidad para gestionar información ruidosa o imprecisa que se presenta en sistemas dinámicos.

En este capítulo se describe el desarrollo de dos planificadores a diferentes niveles. En el nivel inferior, un planificador local asociado a cada Nodo Grid; en el nivel superior, un Meta-Planficador Grid basado en reglas borrosas (MFS), que será capaz de gestionar la incertidumbre e imprecisión inherente a los recursos formantes de los distintos nodos de un sistema Grid. Para medir el rendimiento del Meta-Planificador, éste debe ser capaz de conseguir dos objetivos enfrentados. En primer lugar, reducir el tiempo de finalización de tareas y; en segundo lugar, conseguir una buena distribución de tareas a través de los nodos que forman el Sistema Grid. Según [Wieczoreka09] este MFS es un planificador

multi-criterio, al utilizar los dos criterios descritos anteriormente.

Ambos planificadores son ideados para formar parte de un sistema Grid formado por recursos de una Pequeña y Mediana Empresa (PYME), por lo que podemos catalogar el sistema Grid como un Desktop Grid. Como se indicó en el capítulo 2 un Desktop Grid usa los tiempos ociosos de recursos ya existentes, por lo tanto, analizar su funcionamiento requiere una gran cantidad de tiempo y un gran número de pruebas sobre dichos recursos que se encuentran actualmente en uso. Por ello, para ofrecer una solución a este problema se usa el simulador de sistemas Grid "GridSim" [Sulistio08]. La herramienta GridSim proporciona facilidades para el modelado y simulación de recursos, Nodos Grid y conectividad de red con diferentes capacidades, configuraciones y dominios. Es compatible con las primitivas de composición de aplicaciones, servicios de información para la localización de recursos e interfaces para la asignación de tareas a los recursos y la gestión de su ejecución. Estas características pueden ser utilizadas para simular planificadores de Grid (Meta y Local) y evaluar su rendimiento.

5.2. Componentes del Sistema Grid

Un Grid es un sistema de alta diversidad formado por aplicaciones, componentes middleware y recursos. El proceso de planificación de tareas de estos sistemas Grid está formado por dos niveles, como se observa en la figura 5.2. En nivel superior se encuentra un Meta-Planificador (MS), este es el encargado de seleccionar a qué Nodo Grid (GN) se debe enviar una tarea. En el nivel inferior existe un Planificador Local (LS), el cual en cada nodo elige el recurso que debe ejecutar una tarea. Mientras que un LS está centrado en el problema de asignar tareas a una serie de recursos, un MS requiere en un alto nivel, el descubrimiento, agregación y selección de recursos.

Básicamente, un MS recibe tareas de los usuarios del Grid y selecciona nodos con recursos viables para estas tareas de acuerdo a la información obtenida del estado del sistema. La selección de estos recursos se realiza siguiendo uno o varios objetivos o un comportamiento predefinido de los recursos. A diferencia de los planificadores de sistemas paralelos tradicionales los MS normalmente no tienen control directo sobre los recursos, aunque existen propuestas de brokers o agentes [Berman03] en los que el MS sí tiene una interacción con los recursos. La figura 5.2 muestra la planificación en Grid Jerárquicos. También es posible encontrar planificadores organizados con diferentes estructuras (centralizados, jerárquicos y descentralizados [Rahman11]).

La información sobre el estado de los recursos de un Grid es muy importante para cualquier planificador, especialmente con la naturaleza dinámica y heterogénea de un sistema Grid. El papel del Grid Resource Center (GRC) es ofrecer esta información al MS. GRC es el responsable de reunir y publicar la información sobre el estado de los recursos, como puede

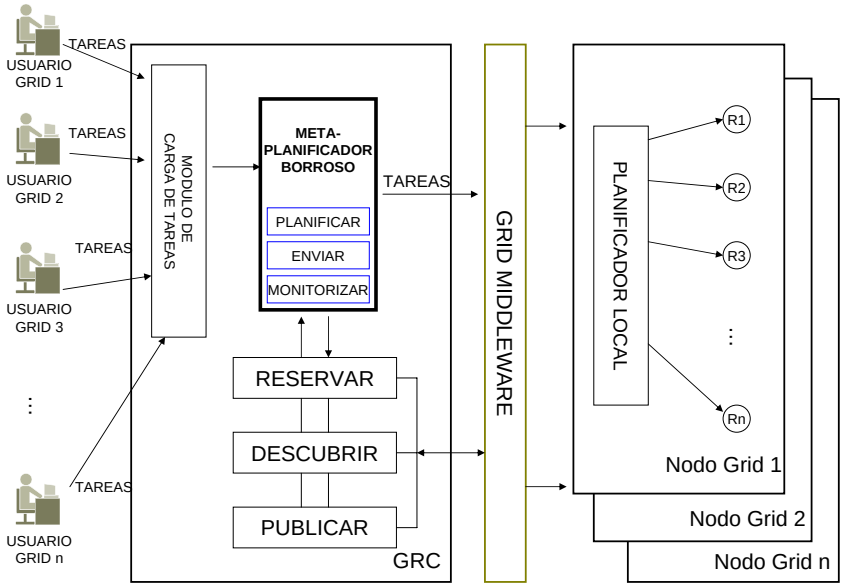


Figura 5.2: Un Sistema Grid con dos niveles de planificación

ser capacidad de CPU, tamaño de memoria, ancho de banda disponible, posibilidades software o carga de trabajo en un momento dado.

El MS de un sistema Grid, a parte de la función propia de planificar las tareas, también tiene las funciones de envío y monitorización de las mismas.

Dentro de cada Nodo Grid (GN) se encuentra un Planificador Local que es responsable de dos objetivos: planificar las tareas dentro de un nodo, donde no solo llegan tareas desde el exterior por parte de los usuarios Grid, sino que, también son ejecutadas tareas de usuarios locales e informa al GRC del estado de los recursos realizando, todo ello de una manera equilibrada, sin sobrecargar determinados recursos.

En el ámbito del Meta-Planificador de esta tesis, el sistema Grid está formado por un conjunto de Nodos Grid. Cada GN está constituido por varios recursos computacionales heterogéneos. El MS no tiene información previa del estado de estos recursos, por lo tanto no tiene ningún control sobre ellos, y sólo interactúa con el planificador local de cada GN. Esta interacción es proporcionada por el módulo GRC, figura 5.2. En este caso el sistema Grid está formado por un planificador de dos niveles [Ranaldo09], un Meta-Planificador y un Planificador Local para cada GN.

5.3. Planificador Local Balanceado

Un planificador local toma la decisión de ejecutar una tarea en un determinado recurso, en un ámbito local, es decir, dentro de un Nodo Grid. Esta ejecución de tareas tiene como fin maximizar (o minimizar) una función objetivo, por ejemplo; el tiempo de ejecución, el número de tareas no ejecutadas, costes de ejecución, etc.

En la literatura previa, la longitud de una tarea es considerada como una constante [Hwang08], que no se puede cambiar durante la ejecución. Esta suposición, para un sistema Grid, que es un entorno muy dinámico, puede hacer los algoritmos de planificación de tareas simples y eficaces, pero puede minar la posibilidad de aplicar la planificación de tareas en algunas situaciones.

En el momento de la planificación, la carga de trabajo de los recursos del Sistema Grid puede ser desconocida. Además, esta carga de trabajo puede cambiar con el tiempo, ya que, las tareas se inician y terminan de forma periódica.

En esta sección se muestra el desarrollo de un planificador local balanceado para una PYME que permita el uso de los ciclos de inactividad de los recursos disponibles para ejecutar una serie de tareas que, en un único recurso local, llevarían varios días de ejecución. Las ejecución de estas mismas tareas en un Nodo Grid sólo tardarían varias horas. La utilización de este planificador local no implica obtener una formación adicional, contratar a nuevos empleados o invertir en nuevos recursos de computación; con el planificador propuesto, se obtienen dos objetivos, minimizar el tiempo de ejecución de un conjunto de tareas y distribuir, de forma proporcionada, estas tareas entre los recursos disponibles obteniendo así, una ejecución balanceada del conjunto total de tareas. Todo ello sin incremento en los costes de operación ya que se utilizan recursos ya existentes.

El planificador local diseñado divide el flujo de trabajo total según los ciclos de inactividad de cada recurso. De este modo, el planificador local balancea la carga de trabajo entre todos los recursos disponibles. Así, todos los recursos deben terminar la tarea que les ha sido asignada en el mismo tiempo. Este planificador local es fácil de usar, flexible y sólo necesita, como entrada, información de la capacidad de proceso libre de cada recurso.

5.3.1. Algoritmo Propuesto

El diseño de planificadores locales para Sistemas Grid es, en general, un campo muy amplio. En primer lugar, se puede realizar el diseño según la función objetivo que el usuario quiere minimizar o maximizar. Por ejemplo, minimizar el tiempo total de finalización del trabajo, minimizar el tiempo de comunicación o maximizar la utilización de recursos o el rendimiento. En segundo lugar, se puede diseñar, siguiendo los requisitos del trabajo (o tareas), los modelos de rendimiento del trabajo, y modelos de recursos del Nodo Grid que

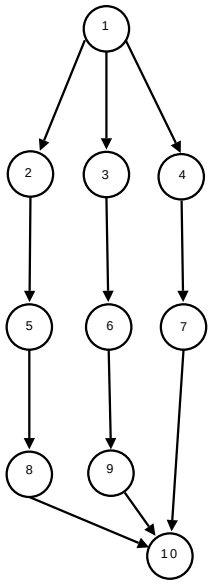


Figura 5.3: Ejemplo de un Grafo Dirigido Acíclico

se especifican y utilizan. El planificador también debe elegir cuidadosamente entre diferentes opciones como reprogramación o la re-planificación de tareas. En este caso, el objetivo del planificador local balanceado es minimizar el tiempo total de ejecución de un trabajo o conjunto de tareas.

Por lo general hay dos tipos de problemas en la planificación de tareas relacionados con el flujo de trabajo: un problema de optimización, formado por tareas independientes, y los Grafos Acíclicos Dirigidos (DAG), con tareas dependientes. En la figura 5.3 se muestra un ejemplo de un DAG simple formado por 10 tareas. En este grafo, las tareas 5, 6 y 7, no pueden comenzar hasta que las tareas anteriores (2, 3 y 4) hayan terminado. En el caso de la última tarea, 10, no puede comenzar hasta que termine la ejecución de las tareas 8 y 9, aunque la tarea 7 haya terminado.

Este planificador local balanceado permite planificar y ejecutar estos dos tipos problemas: por un lado, un conjunto de tareas independientes, que suele ser usadas para llevar a cabo problemas de optimización, y por otra parte un problema de un DAG, cuyas tareas contienen dependencias.

5.3.1.1. Modelo

Para verificar la validez del planificador local balanceado se desarrolla un modelo general. Este modelo está formado por un conjunto de tareas agrupadas en una iteración. Un conjunto de iteraciones forman un flujo de trabajo. Hasta que un flujo de trabajo no finaliza la

ejecución de todas sus iteraciones y tareas asociadas (dependientes o independientes), no puede comenzar el siguiente flujo de trabajo. Un ejemplo típico de esta situación se produce en los procesos de optimización, como son los algoritmos genéticos, en el que hasta que una generación no se ha evaluado por completo no es posible evaluar la siguiente generación.

A fin de formular un modelo integrado, son introducidos los siguientes parámetros:

- n: número de recursos.
- $iter$: número de iteraciones.
- N: carga (workload) para cada iteración.
- N_i: tareas por cada recurso.
- μ_i: instrucciones por segundo (MIPS).
- T_i: tiempo de iteración.
- Ts_i: tiempo de envío de una tarea a un recurso.
- Tr_i: tiempo de recepción de una tarea desde un recurso.
- OT: tiempo total de un flujo de trabajo.

La función objetivo del planificador local balanceado es minimizar el tiempo total de finalización de un flujo de trabajo. Para conseguir este objetivo es necesario contabilizar el tiempo máximo de ejecución de la peor relación flujo de trabajo-recurso. Este tiempo puede ser expresado con la ecuación matemática 5.1.

$$OT = \sum_{i=1}^{iter} MAX_i \left(\frac{N_i}{\mu_i} + Ts_i + Tr_i \right) \tag{5.1}$$

El tiempo de ejecución de una iteración (perteneciente a un flujo de trabajo), ecuación 5.2, es dado por el mayor tiempo de ejecución de una tarea en el peor recurso disponible.

$$T_i = \sum_{k=1}^{tasks} MAX_i \left(\frac{N_i}{\mu_i} + Ts_i + Tr_i \right) \tag{5.2}$$

Donde Ts_i y Tr_i es el tiempo de envío de una tarea al recurso y de recepción, respectivamente, de una tarea desde el recurso. Se consideran estos tiempos insignificantes porque si este tiempo es mayor que el tiempo de ejecución no tiene sentido enviar la tarea al sistema Grid. Por lo tanto el tiempo de una iteración se puede simplificar como se expresa en 5.3.

$$T_i \cong MAX_i \left(\frac{N_i}{\mu_i} \right) \tag{5.3}$$

Para minimizar este tiempo T_i, se asigna una carga proporcional a cada recurso de manera que, se elimina la dependencia con el peor recurso.

$$N_i = N \times \frac{\mu_i}{\sum_{i=1}^{n} \mu_i} \tag{5.4}$$

La innovación de este trabajo radica en que antes de enviar una tarea a un recurso se realiza un estudio previo de sus ciclos ociosos adaptando la longitud de la tarea a esta capacidad. Con esta propuesta se obtienen dos resultados: por una parte se consigue un mejor balance de carga aprovechando más los mejores recursos y, por otra parte se reduce el tiempo de ejecución de un flujo de trabajo.

5.3.1.2. Un ejemplo de ejecución

En la figura 5.4 se muestra el proceso de ejecución de un flujo de trabajo en el modelo desarrollado, simplificándolo con una iteración y tres recursos. En el primer paso el planificador local identifica tres recursos en el sistema. De estos tres recursos obtiene como información, la cantidad de ciclos ociosos disponibles para el flujo de trabajo actual. Cada recurso informa al planificador local del número de ciclos disponibles en ese momento. En el segundo paso, una nueva tarea llega al planificador local. Esta tarea es dividida proporcionalmente en subtareas según el número de ciclos ociosos de cada recurso.

En el siguiente paso se observa como cada recurso está ejecutando su subtarea asignada, finalizando todos en un tiempo similar, con lo que se elimina la dependencia con respecto al peor recurso.

En el cuarto y último paso el planificador local obtiene los resultados de la ejecución de las subtareas y vuelve a preguntar a cada recurso sus ciclos disponibles para la siguiente ejecución.

Como se observa, el planificador local expuesto es altamente dinámico, puesto que al finalizar cada iteración vuelve a obtener los ciclos disponibles de cada recurso.

5.3.2. Resultados experimentales

Debido a las dificultades asociadas a los experimentos reales, los beneficios del planificador local balanceado han sido verificados mediante simulaciones, usando para ello un simulador de Grid. El uso de simulaciones ofrece grandes ventajas frente a otros métodos para estudiar el comportamiento del sistema:

- Permitir la experimentación con el sistema sin interrumpir la actividad real del sistema y evitar la posibilidad de causar daños en el mismo.

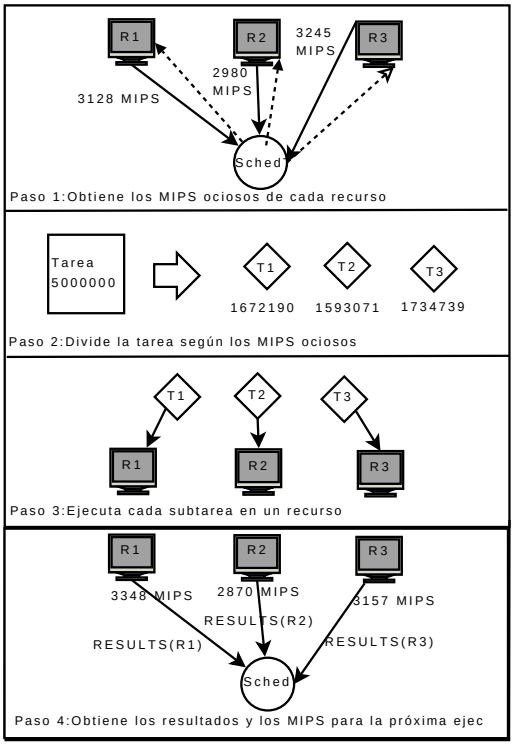


Figura 5.4: Ejemplo de ejecución para una iteración.

Tipos de Recursos	R1	R2	R3	R4	R5
S.O.	Win	Unix	Win	Win	Win
ARQ.	IBM	Solaris	IBM	IBM	IBM
P.E.	1	1	1	1	1
MAX. MIPS	2000	2500	300	3500	4000

Tabla 5.1: Tipos de recursos usados en la simulación

- La simulación es independiente de la existencia del proceso o sistema real, por ello, es posible utilizar en las etapas anteriores a la existencia real, siendo capaz de servir incluso para tomar decisiones antes de su materialización en el mundo físico.
- Las simulaciones permiten mejorar la comprensión del comportamiento del sistema gracias a la obtención de resultados numéricos.
- Las simulaciones permiten realizar análisis comparativos de las diferentes escenas posibles.

Por lo tanto, la funcionalidad del planificador local balanceado es comprobada usando GridSim [Sulistio08]. GridSim es una herramienta de simulación para la distribución de recursos y la aplicación de modelos de programación en paralelo y sistemas de computación distribuida.

En la tabla 5.1 se muestran los tipos de recursos usados en nuestro análisis. Los MIPS para cada elemento de proceso indican el valor máximo de MIPS libres que un recurso ofrece al sistema Grid. Este valor puede cambiar dependiendo de las necesidades del usuario en cada momento, las cuales, han sido modeladas siguiendo una distribución uniforme. Es asumido que el tiempo de transmisión de tareas es insignificante comparado con el tiempo de ejecución de una tarea.

Nuestra propuesta es comparada con algunos de los algoritmos de planificación tradicionales, como son First Come First Served (FCFS) y Round Robin (RR). Los experimentos están divididos en dos apartados. En primer lugar ha sido ejecutado un flujo de tareas en el que todas ellas son independientes simulando un problema de optimización y en segundo lugar se ha ejecutado un DAG.

5.3.2.1. Problema de Optimización

Para este primer problema se han llevado a cabo diversas simulaciones cambiando el número de recursos y de iteraciones. El número de recursos varía entre 25 y 100 y el número de iteraciones es de 100, 500 y 1000 iteraciones. Cada iteración está compuesta por un flujo de tareas y para esta iteración existen unos recursos disponibles en cada momento. Se define la longitud de un flujo de tareas lo suficientemente grande, como para que el sistema Grid

Longitud Flujo de Trabajo= 500.000.000(MI)/Nº recursos= 25					
Nº iteraciones=100		Nº iteraciones=500		Nº iteraciones=1000	
Algoritmo	*Tiempo(Seg.)*	*Algoritmo*	*Tiempo(Seg.)*	*Algoritmo*	*Tiempo(Seg.)*
FCFS	815.681,27	FCFS	4.070.361	FCFS	8.153.256,08
RR	1.028.379,16	RR	5.108.821,63	RR	10.218.636,6
Balanceado	**751.859,56**	Balanceado	**3.761.625,95**	Balanceado	**7.532.137,76**

Tabla 5.2: Experimento formado por 25 recursos

Longitud Flujo de Trabajo= 500.000.000(MI)/Nº recursos= 100					
Nº iteraciones=100		Nº iteraciones=500		Nº iteraciones=1000	
Algoritmo	*Tiempo(Seg.)*	*Algoritmo*	*Tiempo(Seg.)*	*Algoritmo*	*Tiempo(Seg.)*
FCFS	350.501,83	FCFS	1.748.339,75	FCFS	3.502.678,19
RR	343.266,82	RR	1.722.092,89	RR	3.438.388,70
Balanceado	**270.239,52**	Balanceado	**1.350.322,72**	Balanceado	**2.698.851,20**

Tabla 5.3: Experimento formado por 100 recursos

tenga ventaja sobre un sistema no Grid, y moderadamente pequeña de forma que no sean sobrecargados los recursos menos eficientes.

El primer experimento está formado por 25 recursos y 100, 500 y 1000 iteraciones. Los resultados se muestran en la tabla 5.2, en la que se observa que el planificador local balanceado obtiene un tiempo de ejecución menor que los algoritmos tradicionales en todos los casos.

La figura 5.5 muestra una comparación entre el tiempo ofrecido por los tres planificadores para 50 recursos y 100, 500 y 1000 iteraciones. En la figura 5.5 se observa que para un mayor número de iteraciones (1000) la diferencia del tiempo de ejecución con respecto a los algoritmos tradicionales es mayor, lo cual indica que usando los algoritmos FCFS y RR el tiempo degenera cuando es incrementado el número de iteraciones.

En la figura 5.6 se observa una comparativa para 75 recursos y 100, 500 y 1000 iteraciones. Al igual que en las dos pruebas anteriores el planificador local balanceado obtiene un mejor tiempo de ejecución en los tres casos.

El último experimento se ha llevado a cabo con 100 recursos y el mismo número de iteraciones que en los experimentos anteriores. Se comprueba en la tabla 5.3 que el planificador balanceado aumenta su distancia con el resto de algoritmos tradicionales y continúa con su mejora de resultados respecto a los algoritmos tradicionales cuando el número de iteraciones aumenta.

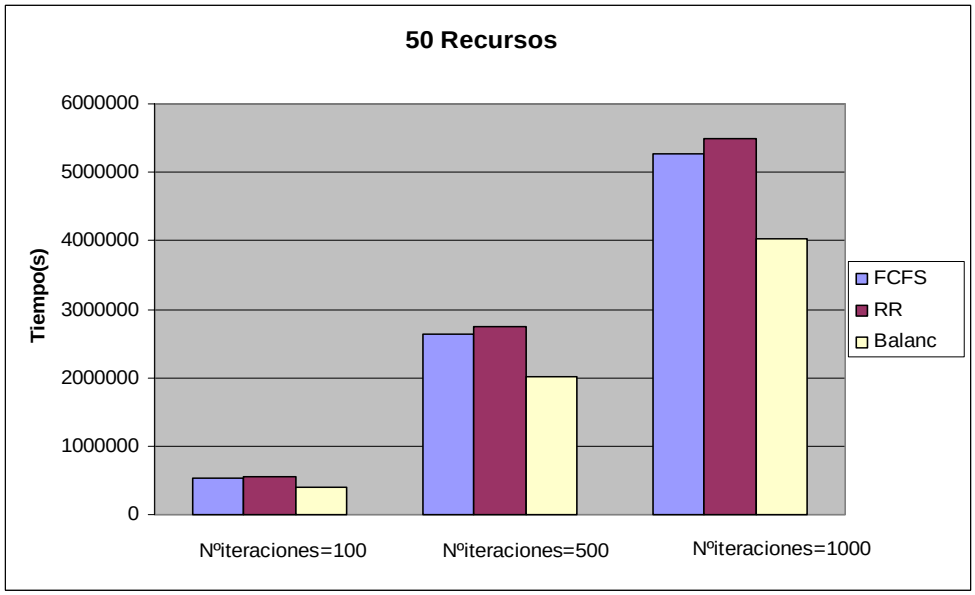


Figura 5.5: Experimento con 50 recursos

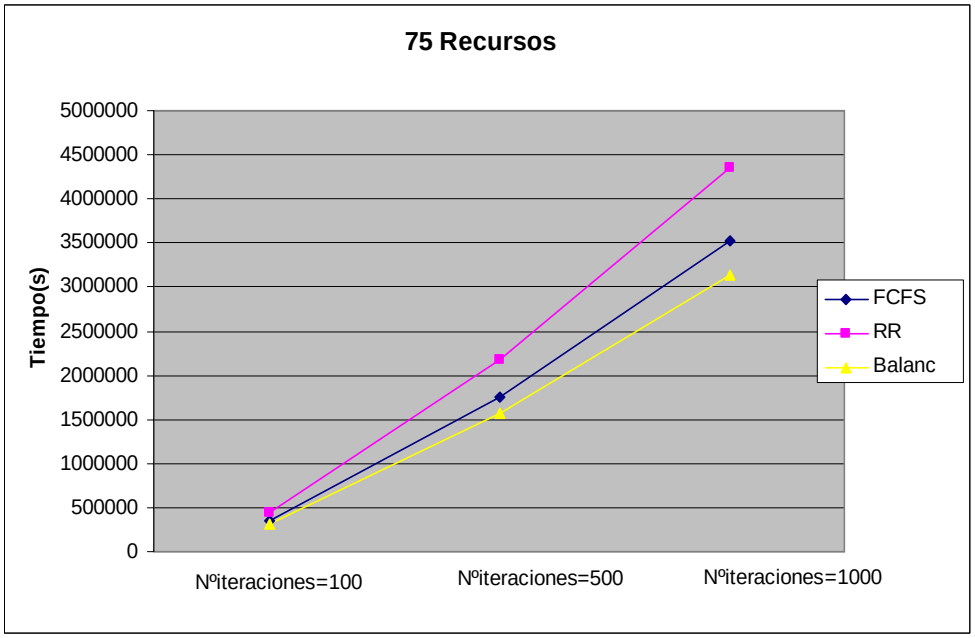


Figura 5.6: Experimento con 75 recursos

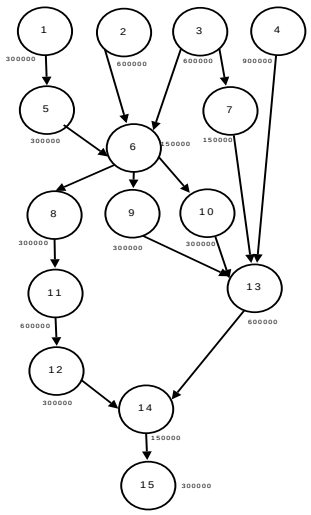


Figura 5.7: Flujo de tareas

5.3.2.2. DAG

En matemáticas y ciencias de la computación, un grafo dirigido acíclico (DAG), es un grafo dirigido sin ciclos, es decir, está formado por una colección de vértices y aristas dirigidos, en el que cada arista conecta un vértice a otro, siguiendo una sóla dirección.

En este experimento ha sido ejecutado un DAG estándar que es utilizado en diversos trabajos [Yu06b]. La estructura de este DAG se muestra en la figura 5.7. Este DAG posee una estructura híbrida con una compleja combinación de ejecución de tareas paralelas y secuenciales. Este DAG contiene 15 tareas y la longitud de cada tarea varía entre 150.000 MI y 900.000 MI (MI=millón de instruciones).

La ejecución de este problema se ha realizado usando una combinación de 25 recursos, cuyas características son indicadas en la tabla 5.1. Para aproximar el DAG de la figura al modelo desarrollado, se considera que todas las tareas que forman el grafo se corresponden con un único flujo de trabajo y que cada una de las tareas forma una iteración, por lo tanto el número de iteraciones para este problema es de 15. Cada iteración o tarea será dividida en subtareas de manera proporcional a la capacidad de cada recurso, para obtener un tiempo mínimo en la ejecución del flujo de trabajo.

Los resultados de este segundo experimento son expresados en la tabla 5.4, donde se muestra el tiempo de ejecución del flujo de trabajo del DAG formado por 15 iteraciones para los algoritmos FCFS, RR y Balanceado. Se observa que el planificador balanceado obtiene una mejora significativa respecto al tiempo de ejecución conseguido por los planificadores tradicionales.

Algoritmo	Tiempo (Seg.)
FCFS	3.916,23
RR	3.741,82
Balanceado	**2.702,52**

Tabla 5.4: Tiempo de Ejecución del DAG

5.4. Meta-Planificador de Tareas basado en Reglas Borrosas

En el núcleo de un sistema Grid la entidad principal de gestión de tareas es comúnmente conocida como Meta-Planificador o Grid Resource Broker. Un Meta-Planificador se utiliza cuando la ejecución de una aplicación puede ser asignada a más de un recurso en diferentes Nodos Grid. Por lo tanto, un Meta-Planificador es responsable de coordinar los planificadores locales para ejecutar una planificación global. El objetivo de un Meta-Planificador es manejar los recursos del Sistema Grid de manera eficiente a través de la planificación y coordinación de los Nodos que lo forman. Un Meta-Planificador redirige todos los trabajos y tareas recibidas a las colas de tareas de los planificadores locales de cada Nodo Grid siguiendo una determinada política de planificación, por ejemplo, tiempo de ejecución mínimo, carga de trabajo, siguiendo criterios de uso de recursos, etc. Su rendimiento es evaluado teniendo en cuenta varios de estos criterios.

5.4.1. Objetivos del Meta-Planificador de Tareas basado en Reglas Borrosas

Para formular el problema de planificación, se considera T_j un conjunto de tareas, que forman una Bolsa de Tareas, de usuarios independientes para ser asignadas a G_i nodos heterogéneos con un propósito dual: minimizar el tiempo de ejecución y usar estos GN de forma eficiente para balancear la carga entre los nodos y la aumentar utilización de sus recursos.

$$T_j = (j \in \{1, 2, 3, ...m\}) \tag{5.5}$$

$$G_i = (i \in \{1, 2, 3, ...n\}) \tag{5.6}$$

El primer objetivo a mejorar es el tiempo de ejecución de una Bolsa de Tareas. El tiempo de ejecución de T es definido como el tiempo en que empieza la ejecución de la primera tarea de la bolsa hasta el tiempo de finalización de la última tarea. La ecuación 5.7 muestra como evaluar el tiempo de ejecución de T, donde M es el tiempo de ejecución o makespan,

$Time_{s1}$ es el tiempo de comienzo de ejecución de la primera tarea y $Time_{end}$ es el tiempo de finalización de la tarea T_m.

$$M = \text{máx}(Time_{end} - Time_{s1}) \tag{5.7}$$

El balanceo de carga se ha definido en la literatura previa como la distribución de tareas que permite utilizar todos los recursos que están disponibles para maximizar el rendimiento y minimizar el tiempo de respuesta. En un sistema Grid que utiliza este tipo de balanceo de carga, el nodo con los recursos con mejor rendimiento, será el responsable de ejecutar gran cantidad de las tareas contenidas en una bolsa de tareas, dejando a otros nodos sin usar.

Nuestro planificador de tareas está diseñado para ser implementado en un sistema Grid, que utiliza recursos que se encuentran ociosos. El uso de estos recursos es compartido entre los propietarios y el sistema Grid, por lo que no es recomendable utilizar siempre los recursos de los mismos nodos para ejecutar las tareas. Por lo tanto, es relevante introducir un nuevo concepto de balanceo de carga, que da opción a los peores nodos para que también sean utilizados en la ejecución de tareas, con lo que obtener una ejecución equilibrada de las tareas con la participación de todos los nodos del Grid. Con esta nueva definición de balanceo de carga, los Nodos Grid con más recursos y mayor poder de cómputo van a ejecutar menos tareas que usando la visión tradicional de balanceo de carga, por lo que estos nodos no se van a sobrecargar. Este enfoque asegura que los nodos con menor poder de cómputo también son utilizados para ejecutar tareas. Por lo tanto, durante la ejecución, algunos de los mejores Nodos puede que no procesen tareas, aún estando disponibles.

El segundo objetivo, Balance de Carga (WLB), es definido en la ecuación 5.8. Varios parámetros relacionados con un Nodo Grid son usados para calcular este objetivo:

- Número de recursos de un Nodo Grid (NumRecursosNodo) con respecto del total de recursos (TotalRecursosGrid) del Sistema Grid.
- Número de tareas ejecutadas en este nodo (CargaNodo) con respecto al total de tareas ejecutadas (CargaTotalGrid) del Sistema Grid.
- Número de Nodos Grid del Sistema (NumNodos).

$$WLB = \frac{NumRecursosNodo}{TotalRecursosGrid} - \left| \frac{CargaNodo}{CargaTotalGrid} - \frac{1}{NumNodos} \right| \tag{5.8}$$

Así se comprueba que todos los Nodos Grid participan en la ejecución de tareas de forma proporcional y equilibrada. WLB se evalúa para cada nodo Grid cuando la última tarea de la Bolsa de tareas ha sido ejecutada. Valores cercanos a cero indican un balanceo de carga óptimo.

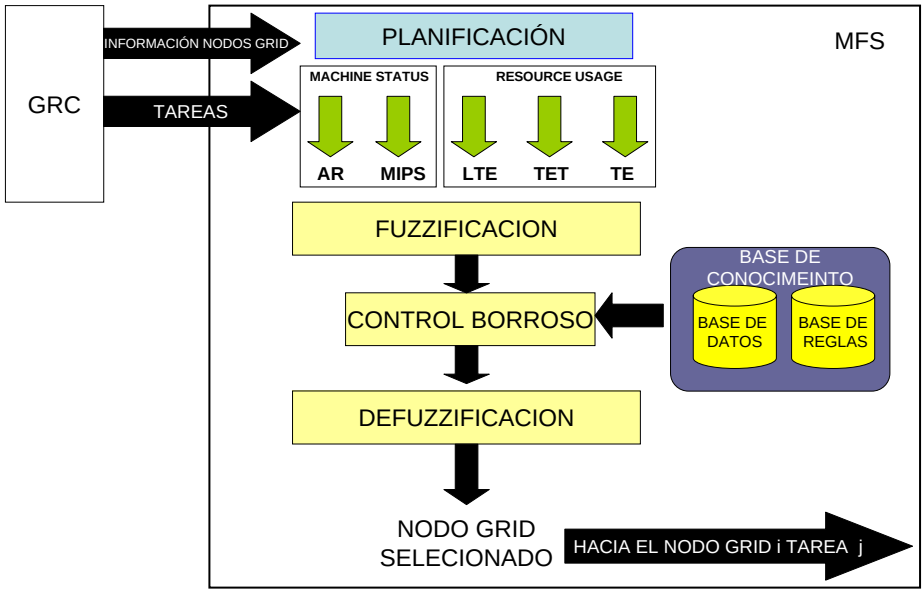


Figura 5.8: Estructura del Meta-Planificador Borroso

Es posible concluir que el balance de carga (WLB) junto con el tiempo de ejecución (makespan), conforman dos objetivos contradictorios. Según nuestra propuesta, para obtener un buen balance de carga es recomendable utilizar todos los nodos, incluidos aquellos nodos que tienen unas características menos favorables, por lo que usando estos nodos, el tiempo de ejecución de la bolsa de tareas puede verse empeorado.

5.4.2. Estructura del Meta-Planificador Borroso

Un Meta-Planificador dinámico, como el que se propone, requiere cierta información del estado actual de cada Nodo Grid para ejecutar las tareas asociadas a una bolsa de tareas. Con esta información, proporcionada por cada planificador local al Grid Resource Center (GRC) nuestro MFS decide a qué nodo debe ser enviada la siguiente tarea con la finalidad de cumplir dos objetivos: minimizar el tiempo de ejecución y obtener un valor de WLB cercano a cero. En el GN seleccionado, el planificador local envía la tarea al mejor recurso disponible. La estructura de nuestro Meta-Planificador Borroso es mostrada en la figura 5.8.

El estado de un Nodo Grid puede cambiar en cualquier momento por varias razones. Antes de enviar una nueva tarea al sistema Grid, nuestro MFS comprueba el estado actual de todos los nodos que forman el sistema Grid, figura 5.8. Según este estado cada nodo conseguirá un Factor de Selección (FS). Este valor es obtenido a partir de cinco medidas sobre el estado

actual e histórico del Nodo. Estos cinco parámetros varían entre 0 y 1 y están relacionados con el estado de cada Nodo Grid.

5.4.2.1. Estado de un Nodo Grid

Para definir el estado de un nodo Grid, y así calcular mediante lógica borrosa su Factor de Selección, se han usado cinco parámetros clasificados en dos grupos. Los parámetros relacionados con el estado actual del Nodo son definidos y agrupados como "Machine Status". "Resource Usage" agrupa parámetros históricos, relacionados con las tareas ejecutadas o en ejecución por el Nodo hasta este momento.

Los parámetros de Machine Status están altamente relacionados con el primer objetivo (makespan), porque estos parámetros simbolizan la capacidad de cómputo de un Nodo Grid. Un nodo con alto poder computacional puede finalizar tareas en un menor tiempo. En este grupo son definidos dos parámetros:

- Tasa de recursos disponibles (AR): un recurso disponible en un Nodo Grid es un elemento de computación (cluster, servidor, PC,...) con capacidad de cómputo suficiente para ejecutar la siguiente tarea contenida en la bolsa de tareas. Por diversas causas (corte de luz, reinicio, caída de la red, etc), un recurso puede no estar disponible en un momento dado. Esta entrada permite conocer el número de recursos disponibles de cada Nodo para su utilización por el sistema Grid. La ecuación de normalización para esta entrada es:

$$AR(i) = \frac{RecursosDisponibles(i)}{RecursosTotales(i)} \tag{5.9}$$

  donde i es el identificador de un Nodo del Sistema Grid.

- Tasa de MIPS libres (MIPS): en un Nodo Grid, cada recurso disponible tiene una capacidad de cómputo libre que puede ser aprovechada por el Sistema Grid. Esta entrada indica la capacidad de computación ociosa que tiene un Nodo Grid. La ecuación de normalización es la siguiente:

$$MIPS(i) = \frac{MipsLibres(i)}{TotalMips(i)} \tag{5.10}$$

  donde i es el identificador de un Nodo del Sistema Grid.

Los parámetros agrupados en Resource Usage están relacionados con el segundo objetivo WLB. Estos parámetros muestran información histórica de las tareas que, previamente, se han ejecutado en el Nodo Grid para la bolsa de tareas actual. Estos parámetros están destinados a distribuir tareas en Nodos con menor número de tareas ejecutadas o con menor carga de ejecución recibida. De esta forma los Nodos Grid con mayor capacidad de cómputo

no son sobrecargados con nuevas tareas para ejecutar. Dos de estos parámetros contienen información relacionada con las tareas ya ejecutadas y se utilizan para comprobar si un nodo está sobrecargado. Estos parámetros son: la duración de las tareas finalizadas y el tiempo de uso del Nodo. También son utilizadas para la evaluación el número de tareas ejecutadas por un nodo durante la ejecución de una bolsa de tareas:

- Tamaño de tareas ejecutadas (LTE): en la bolsa de tareas cada una de las tareas tiene un tamaño distinto, por lo tanto este parámetro es necesario para estudiar la carga histórica de un Nodo Grid. Esta entrada expresa el tamaño de las tareas que han sido ejecutadas por el Nodo, en relación con el total de tareas que han sido ejecutadas para la bolsa de tareas actual. La ecuación de normalización para esta entrada es la siguiente:

$$LTE(i) = \frac{TamTareasEjecutadas(i)}{\sum_{i=1}^{n} TamTareasEjecutadas} \tag{5.11}$$

  donde i es un Nodo Grid y n es el número total de Nodos del Grid.

- Tiempo ejecutando Tareas (TET): cada Nodo es diferente, cambiante y heterogéneo, por lo que el tiempo de ejecución de la misma tarea varía en cada nodo. Este parámetro relaciona el tiempo total empleado por un Nodo para ejecutar las diferentes tareas que le han sido asignadas, respecto al tiempo total de ejecución en todos los Nodos. La ecuación de normalización para este parámetro es la siguiente:

$$TET(i) = \frac{TiempoEmpleado(i)}{\sum_{i=1}^{n} TiempoEmpleado} \tag{5.12}$$

  donde i es un Nodo Grid y n es el número total de Nodos del Grid.

- Tareas Ejecutadas (TE): número de tareas ejecutadas y completadas en cada Nodo para la bolsa de tareas actual, en relación con el total de tareas finalizadas por todos los Nodos formantes del Sistema Grid. La ecuación de normalización para esta entrada es la siguiente:

$$TE(i) = \frac{TareasFinalizadas(i)}{\sum_{i=1}^{n} TareasFinalizadas} \tag{5.13}$$

  donde i es un Nodo Grid y n es el número total de Nodos del Grid.

5.4.3. Conjuntos Borrosos

Para estas cinco entradas se calcula el grado de pertenencia para cada conjunto borroso. Las funciones de pertenencia son definidas de tipo triangular. Son usadas funciones triangulares

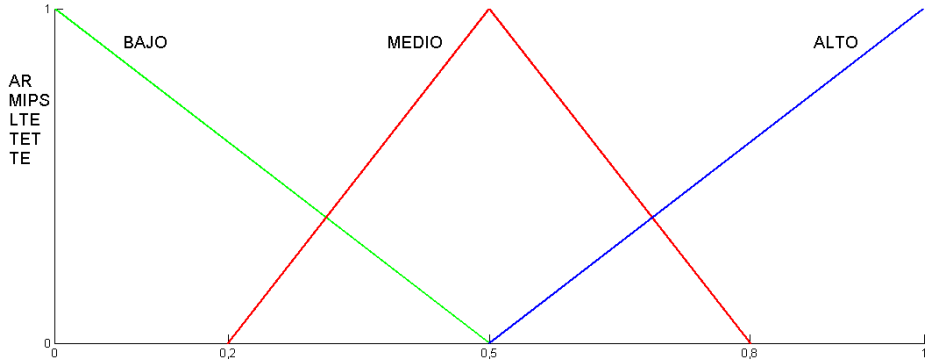


Figura 5.9: Funciones de Pertenencia para las entradas

ya que han sido ampliamente utilizadas en aplicaciones de tiempo real debido a sus fórmulas simples y eficiencia computacional. La ecuación de está función triangular es mostrada en 5.14, con valores en el intervalo [0,1].

$$\mu(x) = \begin{cases} 0, x \leq a \\ \frac{(x-a)}{(m-a)}, x \in (a, m] \\ \frac{(b-x)}{(b-m)}, x \in (m, b] \\ 0, x \geq b \end{cases} \tag{5.14}$$

La variable de salida es definida como Factor de Selección de un Nodo Grid. Como se observa en la figura 5.10, esta variable de salida contendrá cinco conjuntos difusos, a diferencia de los conjuntos de la variable de entrada, figura 5.9, que están compuestos de tres variables lingüísticas. Las etiquetas del conjunto borroso de salida indican la idoneidad del Nodo Grid para ser seleccionado. Si un determinado Nodo obtiene como salida un Factor de Selección "Alto" o "Muy Alto", este nodo será un buen candidato para recibir la próxima tarea a ejecutar.

5.4.3.1. Reglas Borrosas

Las reglas de inferencia borrosas, tipo Mamdani [Mamdani75], son desarrolladas usando las variables de entrada y la variable de salida que se ha definido previamente. Estas reglas tienen el mismo peso y todas usan el conector AND. Según nuestra experiencia en Sistemas Grid, han sido desarrolladas diversas reglas para el Meta-Planificador borroso, las cuales consideran los dos objetivos mostrados anteriormente. Las reglas usadas son las siguientes:

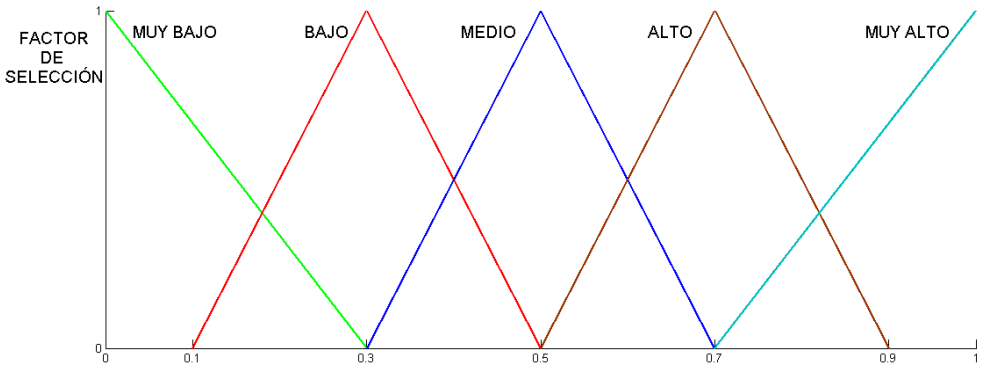


Figura 5.10: Función de Pertenencia para la salida

- Regla 1. *si* **AR** *es BAJO y* **MIPS** *es BAJO y* **LTE** *es BAJO y* **TET** *es BAJO y* **TE** *es BAJO entonces* **FS** *es BAJO*
- Regla 2. *si* **AR** *es BAJO y* **MIPS** *es BAJO y* **LTE** *es MEDIO y* **TET** *es MEDIO y* **TE** *es MEDIO entonces* **FS** *es BAJO*
- Regla 3. *si* **AR** *es BAJO y* **MIPS** *es BAJO y* **LTE** *es ALTO y* **TET** *es ALTO y* **TE** *es ALTO entonces* **FS** *es MUYBAJO*
- Regla 4. *si* **AR** *es MEDIO y* **MIPS** *es MEDIO y* **LTE** *es BAJO y* **TET** *es BAJO y* **TE** *es BAJO entonces* **FS** *es MEDIO*
- Regla 5. *si* **AR** *es MEDIO y* **MIPS** *es MEDIO y* **LTE** *es MEDIO y* **TET** *es MEDIO y* **TE** *es MEDIO entonces* **FS** *es BAJO*
- Regla 6. *si* **AR** *es MEDIO y* **MIPS** *es MEDIO y* **LTE** *es ALTO y* **TET** *es ALTO y* **TE** *es ALTO entonces* **FS** *es BAJO*
- Regla 7. *si* **AR** *es ALTO y* **MIPS** *es ALTO y* **LTE** *es BAJO y* **TET** *es BAJO y* **TE** *es BAJO entonces* **FS** *es MUYALTO*
- Regla 8. *si* **AR** *es ALTO y* **MIPS** *es ALTO y* **LTE** *es MEDIO y* **TET** *es MEDIO y* **TE** *es MEDIO entonces* **FS** *es ALTO*
- Regla 9. *si* **AR** *es ALTO y* **MIPS** *es ALTO y* **LTE** *es ALTO y* **TET** *es ALTO y* **TE** *es ALTO entonces* **FS** *es MEDIO*

Con el fin de explicar el nuevo concepto balanceo de carga usado en este trabajo, se explican dos de las reglas usadas. En la regla 9 las cinco entradas tienen el valor *ALTO*. Usando un Meta-Planificador tradicional, que no tiene en cuenta el WLB definido en esta tesis, el Factor de Selección para esta regla debería ser *MUYALTO*, puesto que este Nodo tiene una gran capacidad de cómputo (AR es *ALTO* y MIPS es *ALTO*). Sin embargo siguiendo

nuestra nueva propuesta de WLB, el Factor de Selección en esta regla es *MEDIO*. Este valor medio se produce porque el Nodo que cumple esta regla se encuentra sobrecargado, como indican las entradas históricas (LTE, TET y TE) con valor *ALTO*, por lo que se aconseja, en la medida de lo posible, elegir otro Nodo para ejecutar esta tarea. A pesar de que este Nodo tiene un alto poder de cómputo para ejecutar tareas (AR y MIPS altos), ya ha ejecutado un gran número de tareas (TE), con grandes tamaños (LTE) y ha empleado mucho tiempo (TET), para la ejecución de tareas de la bolsa de tareas actual. Si este Nodo es seleccionado para ejecutar la tarea, la tasa de WLB del Nodo podría empeorar, ya que este nodo tiene actualmente un alto valor para las entradas de Resource Usage. Utilizando nuestro enfoque de balanceo de carga, al estar las tres entradas relacionadas con tareas (LTE, TET, TE) en valor *ALTO*, el Nodo Grid no debe ser el más adecuado para recibir más tareas para ejecutar.

En la regla 7, se observa que las entradas para el grupo de Resource Usage son *BAJO* y el estado del nodo es *ALTO*. Un nodo Grid con estos valores es excelente para recibir y ejecutar tareas, ya que este Nodo tiene mucho margen para un posible empeoramiento de su WLB y sus recursos son capaces de ejecutar una tarea en un tiempo mínimo.

Hay muchos métodos para defuzzificar y obtener la salida: bisectriz, centroide, centro de gravedad, máximo, etc. En este trabajo, se utiliza el método más popular de defuzzificación que es el método centroide. Este método devuelve el centro del área bajo la curva.

5.4.4. Experimentos y Resultados

En esta sección se propone un Meta-Planificador Borroso para lograr un menor tiempo de ejecución y un balance de carga mínimo en la ejecución de una serie de tareas. Como se ha comentado anteriormente, en este estudio, las tareas de simulación se recogen en una Bolsa de Tareas. Se considera planificar usando bolsa de tareas, ya que son aplicaciones paralelas cuyas tareas son independientes. Las aplicaciones con BoT son usadas en una gran variedad de escenarios [Netto11], [Lee07], incluyendo biología computacional, imágenes por computador, minería de datos, cálculos y simulaciones fractales. Además, a causa de la independencia de las tareas, las aplicaciones de BoT pueden ser ejecutadas con éxito en entornos distribuidos geográficamente, como son los Sistemas Grids, como demuestra el proyecto SETI@home.

5.4.4.1. Parámetros de simulación y escenarios heterogéneos

El tamaño de la Bolsa de Tareas consideradas varía entre 500 y 1500 tareas. El MFS despacha cada tarea una vez, no realiza ningún tipo de reubicación de tareas. El escenario de simulación en este estudio está formado por cinco Nodos Grid y 240 recursos repartidos de forma heterogénea entre los Nodos.

Heterogeneidad	Tareas	Recursos
HH	High [1,300.000]	High [1,10.000]
HL	High [1,300.000]	Low[1,1.000]
LH	Low[1,10.000]	High [1,10.000]
LL	Low[1,10.000]	Low[1,1.000]

Tabla 5.5: Escenarios de simulación

Los supuestos realizados en este estudio son los siguientes: las tareas de un trabajo han sido divididas previamente formando una única unidad atómica; todas las tareas son independientes y el paso de parámetros entre tareas no es considerado, sin embargo, este MFS puede ser utilizado en escenarios con tareas dependientes.

Los escenarios propuestos para llevar a cabo nuestro estudio son derivados de los escenarios propuestos en [Braun01]. La heterogeneidad de las tareas es categorizada en dos tipos: (1) alto, cuyos valores están en el intervalo [1,300.000] y (2) bajo, variando entre [1,10.000]. La heterogeneidad de los recursos también está clasificada en dos tipos: (1) alto, con un rango entre [1,10.000] y (2) bajo, con valores en el intervalo [1,1.000]. Estos cuatro escenarios se muestran en la tabla 5.5.

Como ejemplo, cuando ambas heterogeneidades, tareas y recursos, son altas, la longitud de cada una de las tareas pertenece al intervalo [1, 300.000] Millones de Instrucciones (MI) y para un recurso su capacidad de cómputo expresada en Millones de Instrucciones por Segundo (MIPS) pertenece al intervalo [1, 10.000]. Según el teorema Central de Límite en estadística, cuando el número de ejemplos excede o es igual a 30, el resultado de la muestra es muy similar a la de la distribución normal. Por lo tanto, cada heurística de planificación incluye cuatro escenarios, cada uno de los cuales cuenta con 30 muestras diferentes para cada tamaño de la bolsa de las tareas.

Debido a las dificultades asociadas a las pruebas en entornos reales, las bondades de este Meta-Planificador Borroso han sido verificadas mediante simulaciones. Por lo tanto, el Meta-Planificador ha sido desarrollado usando el simulador propuesto en [Sulistio08]. GridSim es un conjunto de herramientas de simulación ampliamente utilizado para el modelado de recursos y las aplicaciones de planificación en sistemas de computación paralela y distribuida.

El Meta-Planificador borroso es comparado con tres heurísticas dinámicas tradicionales en modo batch para Grid Computing (Max-Min, Min-Min y Sufferage) y con una extensión de la heurística Min-Min: Segmented Min-Min, con la sub-política 1 (Media). También ha sido realizada una pequeña modificación de nuestra propuesta, que consiste en aplicar un simple pre-procesamiento a la Bolsa de Tareas. En este caso, la BoT es ordenada según el tamaño

de las tareas, por lo que las tareas con mayor longitud van a ser planificadas y ejecutadas en primer lugar. En la planificación sin preprocesamiento las tareas son enviadas a los Nodos, según se han generado, siguiendo una distribución uniforme aleatoria, sin realizar ningún tipo de ordenación o preprocesamiento.

El sistema Grid para estas simulaciones está formado por un conjunto de Nodos Grid. Cada Nodo tiene los siguientes parámetros: el número de recursos varía entre 6 y 144, cada recurso tiene un único procesador y el ancho de banda de comunicación entre los Nodos es de 2,5 Gbs.

Los parámetros de simulación son aplicados a cada heurística para obtener los resultados experimentales. Las propiedades de tareas y recursos permiten obtener cuatro escenarios clasificados según el grado de heterogeneidad:

- Tareas y recursos están caracterizados por alta Heterogeneidad (HH).
- Tareas altamente heterogéneas y recursos poco heterogéneos (HL).
- Tareas poco heterogéneas y recursos altamente heterogéneos (LH).
- Tareas y recursos tienen poca heterogeneidad (LL).

5.4.4.2. Resultados

Con el fin de minimizar el tiempo de ejecución (makespan) resulta idóneo usar los mejores Nodos disponibles para finalizar la ejecución de las tareas en un tiempo menor. Para minimizar el objetivo de balance de carga (WLB) es necesario que sean utilizados todos los Nodos disponibles en el Sistema Grid, incluyendo los Nodos con el capacidad de computación de bajo. Si los Nodos con poca capacidad de computación se incluyen la ejecución de una Bolsa de Tareas el tiempo puede resultar penalizado. Por lo tanto, como se indicó anteriormente, makespan y WLB se convierten en objetivos contradictorios.

En las siguientes tablas, el rendimiento de las heurísticas dinámicas y del MFS se definen con dos medidas: makespan y WLB. Se puede observar que el Meta-Planificador Borroso obtiene un buen funcionamiento en los diferentes escenarios.

Escenario HH

En los experimentos de esta sección, el rendimiento del Meta-Planificador Borroso es constatado mediante un escenario en el que las tareas y recursos tienen una alta heterogeneidad.

En la figura 5.11, es mostrado el tiempo de ejecución para este escenario (HH). Según los resultados, para un menor tamaño de la BoT (500-600 tareas) nuestra propuesta obtiene un tiempo de ejecución similar al conseguido por las heurísticas dinámicas. Con un mayor tamaño de la bolsa de tareas (700-1500 tareas) nuestro Meta-Planificador mejora los resultados de las heurísticas tradicionales. Usando el pre-procesamiento, descrito con anterioridad, el tiempo de ejecución para cualquier número de tareas mejora en un 2 % con

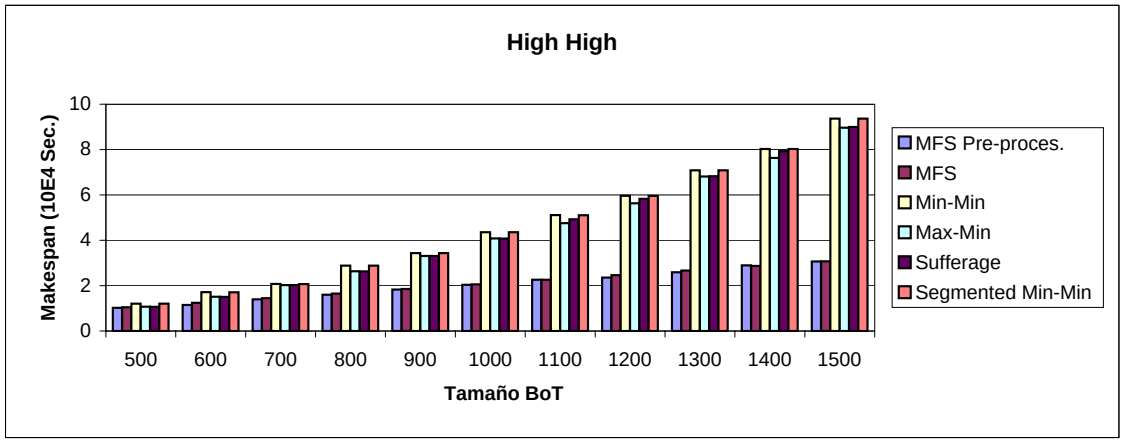


Figura 5.11: Makespan para escenario HH

respecto a los resultados obtenidos sin pre-procesamiento. Para este escenario Max-Min es la heurística tradicional que logra un mejor tiempo. La mejora de tiempo entre Min-Min y Segmented Min-Min es del 1 % a favor de la extensión de Min-Min.

La tabla 5.6 muestra los valores obtenidos para el objetivo de balance de carga (WLB) en el escenario HH. Para nuestras propuestas el valor obtenido es cercano al óptimo. Se observa que sin pre-procesamiento (MFS) el Meta-Planificador Borroso obtiene un mejor valor para el balanceo de carga. En cualquier heurística, tamaños grandes para la bolsa de tareas, empeoran el resultado de WLB. Las heurísticas dinámicas tradicionales logran peores resultados, siendo la heurística Segmented Min-Min la que consigue el mejor resultado, aunque este valor está muy lejos del valor óptimo.

En este escenario con alta heterogeneidad de tareas y alta heterogeneidad de recursos, para cualquier tamaño de la bolsa de tareas, nuestro Meta-Planificador obtiene un menor tiempo de ejecución y un valor cercano al óptimo para el balance de carga, mejorando en todo caso los resultados ofrecidos por las heurísticas dinámicas tradicionales.

Escenario HL

En este apartado es mostrado el rendimiento del Meta-Planificador para una alta heterogeneidad de tareas y una heterogeneidad baja para los recursos. En este escenario el valor máximo de capacidad de cómputo para un recurso son 1000 MIPS, por lo que el tiempo de ejecución es incrementado con respecto al escenario anterior.

Para pequeños tamaños de la BoT, el tiempo de ejecución es similar en todas las políticas. Con un tamaño mayor de la bolsa de tareas, la mejora de nuestra propuesta y las heurísticas tradicionales es mayor. El tiempo de ejecución para el escenario HL es mostrado en la figura 5.12. Con pre-procesamiento, se obtiene una mejora del 1.5 % con respecto al no pre-procesamiento.

	Meta-Schedulers					
BoT	**MFS Pre-Pro**	**MFS**	Max-Min	Min-Min	Suff	Seg Min-Min
500	0.0091	0.007	0.2671	0.2626	0.2626	0.2471
600	0.0089	0.0071	0.26	0.26	0.26	0.251
700	0.0074	0.0065	0.2646	0.2646	0.2646	0.2246
800	0.0073	0.0061	0.2618	0.2618	0.2618	0.2515
900	0.0073	0.0064	0.2602	0.2602	0.2602	0.2236
1000	0.0071	0.0062	0.2634	0.2634	0.2634	0.2543
1100	0.0125	0.0113	0.2641	0.2641	0.2641	0.2569
1200	0.0115	0.0108	0.2609	0.2609	0.2609	0.2496
1300	0.0114	0.0108	0.2657	0.2657	0.2657	0.2531
1400	0.012	0.0111	0.2593	0.2593	0.2593	0.2427
1500	0.0121	0.0111	0.2637	0.2637	0.2637	0.2584

Tabla 5.6: Balanceo de Carga para escenario HH

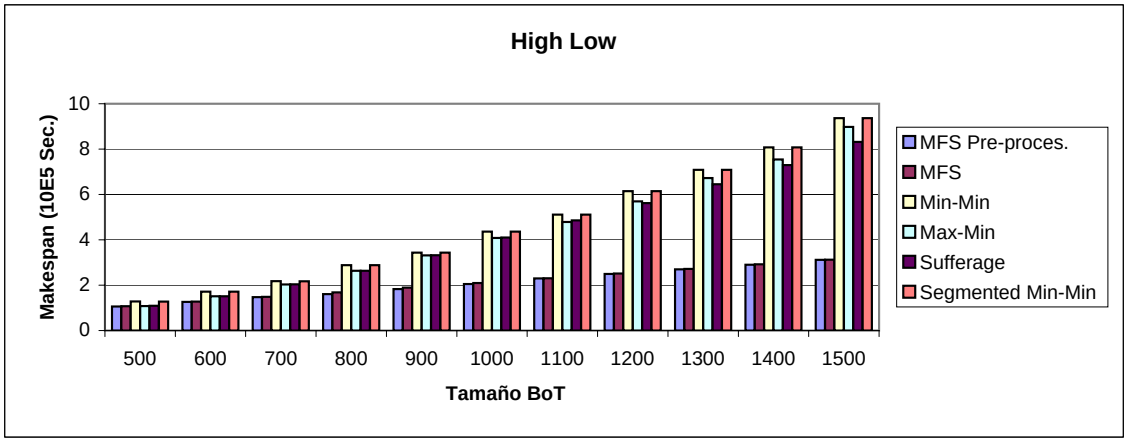


Figura 5.12: Makespan para escenario HL

	Meta-Schedulers					
BoT	**MFS Pre-Pro**	**MFS**	Max-Min	Min-Min	Suff	Seg Min-Min
500	0.035	0.0274	0.2746	0,256	0,2613	0.2512
600	0.0325	0.0272	0.2556	0.2556	0.2625	0.2493
700	0.0317	0.0254	0.2658	0.2618	0.2713	0.2584
800	0.0312	0.0256	0.2597	0.2636	0.2577	0.2515
900	0.0307	0.0251	0.2608	0.2687	0.2626	0.2594
1000	0.0301	0.0257	0.2643	0.2556	0.26	0.2534
1100	0.0305	0.0255	0.2612	0.2659	0.2646	0.2558
1200	0.0306	0.0247	0.2638	0.2597	0.2618	0.2562
1300	0.0311	0.0256	0.264	0.2608	0.2602	0.2578
1400	0.0309	0.0252	0.2625	0.2643	0.2634	0.2523
1500	0.0308	0.025	0.2601	0.2632	0.2623	0.2565

Tabla 5.7: Balanceo de Carga para escenario HL

La tabla 5.7 muestra el balanceo de carga para el escenario HL. En este escenario el rendimiento del MFS es peor que en el escenario previo. Así, con una combinación de tareas altamente heterogéneas y recursos poco heterogéneos, el tiempo empleado por un recurso para finalizar la ejecución de una tarea es mayor; lo que produce una sobrecarga de los Nodos. A pesar de esta situación, nuestras propuestas obtienen un menor WLB que las heurísticas tradicionales. Para el MFS, este valor de balanceo de carga es penalizado para una combinación de tareas altamente heterogéneas y recursos poco heterogéneos.

Escenario LH

Los resultados presentados en esta sección corresponden al escenario en el que la heterogeneidad de las tareas es baja y la de heterogeneidad de los recursos alta. Con esta configuración de escenario el tiempo de ejecución de tareas será el menor de todos los escenarios planteados.

Para este escenario nuestra propuesta obtiene un menor tiempo de ejecución para cualquier tamaño de bolsa de tareas. En la figura 5.13 se muestra que el Meta-planificador borroso propuesto logra el menor tiempo que el resto de heurísticas dinámicas. Con un tamaño de bolsa de tareas pequeña, la diferencia de tiempo entre nuestra propuesta y las heurísticas es pequeña; pero usando una bolsa con 700 o más tareas, esta diferencia es incrementada obteniendo nuestra propuesta los mejores resultados. Las heurísticas tradicionales obtienen tiempos similares entre ellas. La diferencia entre nuestra propuesta con pre-procesamiento y no pre-procesamiento es un 2 %.

Según los resultados expresados en la tabla 5.8, el Meta-Planificador borroso logra el mejor comportamiento en términos de WLB comparado con sus competidores. Para este escenario

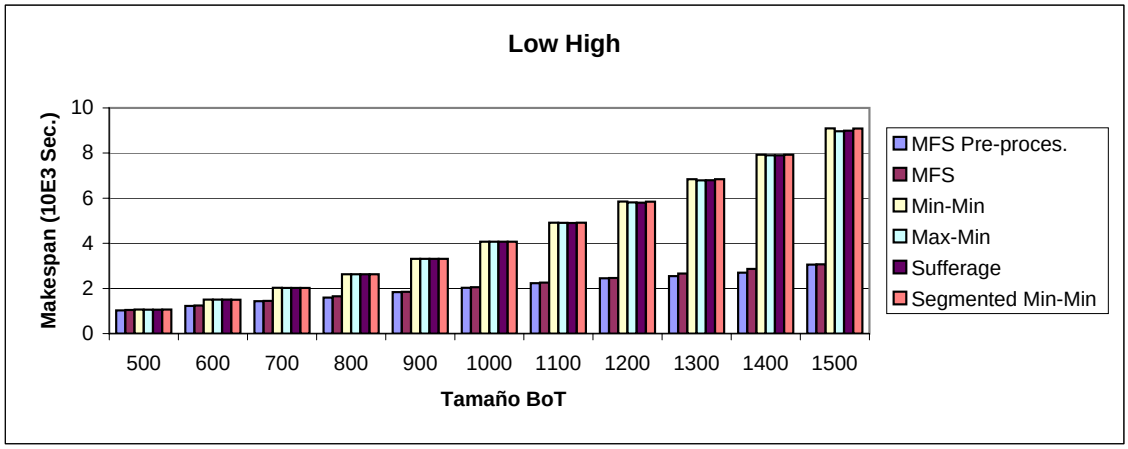


Figura 5.13: Makespan para escenario LH

nuestra propuesta obtiene valores cercanos a cero. El balanceo de carga es mejor con un menor número de tareas. La longitud ideal para la bolsa de tareas oscila entre 500 y 1.000 tareas.

Escenario LL

En el siguiente conjunto de resultados se muestra el tiempo de ejecución y el balance de carga para baja heterogeneidad de Tareas y Recursos. En la figura 5.14 se presenta el resultado para el tiempo de ejecución en el escenario LL.

Para un tamaño de bolsa de tareas bajo (500 tareas) nuestro Meta-Planificador obtiene unos resultados similares a las heurísticas tradicionales. Con un mayor tamaño de BoT, se observa un aumento en la diferencia de tiempo entre nuestro MFS y las heurísticas. Este tiempo es similar para todas las heurísticas tradicionales con cualquier tamaño de bolsa. La mejora de tiempo entre pre-procesamiento y no pre-procesamiento es de un 1 % a favor de pre-procesamiento.

La tabla 5.9 ilustra el balanceo de carga para el escenario LL. En terminos de WLB, nuestro MFS consigue valores cercanos al óptimo con un tamaño de tareas pequeño. Las heurísticas dinámicas obtienen valores alejados de cero en este escenario.

El mejor valor de balanceo de carga en este escenario es obtenido con 600 tareas, para nuestro MFS sin pre-procesamiento. Con más de 1.000 tareas el planificador sin pre-procesamiento consigue mejor valor para WLB que el planificador con pre-procesamiento.

	Meta-Schedulers					
BoT	**MFS Pre-Pro**	**MFS**	Max-Min	Min-Min	Suff	Seg Min-Min
500	0.0077	0.0073	0.2626	0.2626	0.2626	0.2545
600	0.0074	0.0067	0.26	0.26	0.26	0.2507
700	0.0072	0.0065	0.2646	0.2646	0.2646	0.2568
800	0.007	0.0063	0.2618	0.2618	0.2618	0.2496
900	0.007	0.0063	0.2602	0.2602	0.2602	0.2475
1000	0.0069	0.0061	0.2634	0.2634	0.2634	0.2567
1100	0.0119	0.0112	0.2641	0.2641	0.2641	0.2525
1200	0.0121	0.0113	0.2613	0.2613	0.2613	0.2563
1300	0.0113	0.0106	0.2657	0.2657	0.2657	0.2529
1400	0.0116	0.0107	0.2593	0.2593	0.2593	0.2435
1500	0.02	0.011	0.2637	0.2637	0.2637	0.2578

Tabla 5.8: Balanceo de Carga para escenario LH

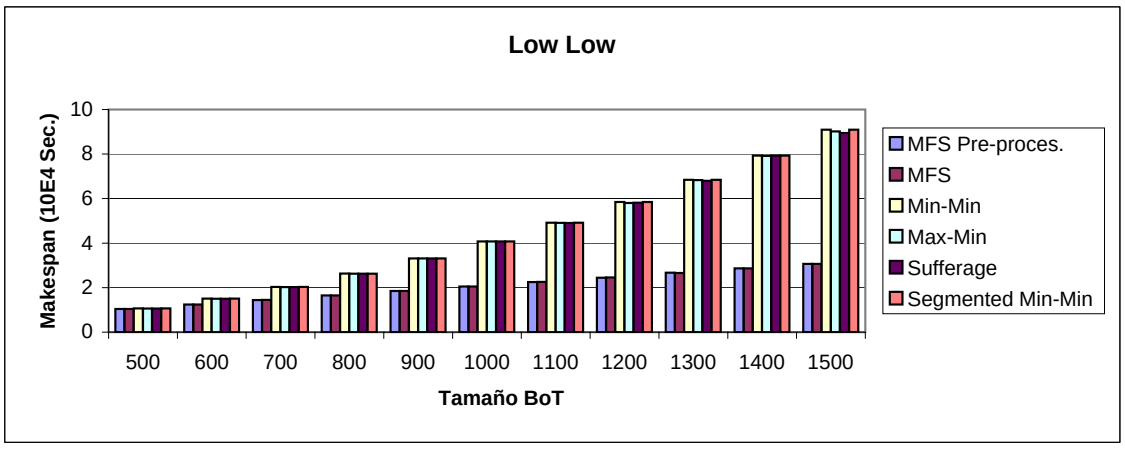


Figura 5.14: Makespan para escenario LL

	Meta-Schedulers					
BoT	**MFS Pre-Pro**	**MFS**	Max-Min	Min-Min	Suff	Seg Min-Min
500	0.0065	0.0056	0.2687	0.2687	0.2687	0.2523
600	0.0059	0.0051	0.2556	0.2556	0.2556	0.2421
700	0.0062	0.0056	0.2653	0.2653	0.2653	0.2523
800	0.0071	0.0059	0.2597	0.2597	0.2597	0.2479
900	0.0068	0.0057	0.2608	0.2608	0.2608	0.2486
1000	0.0067	0.0057	0.2643	0.2643	0.2643	0.2458
1100	0.0088	0.0076	0.2612	0.2612	0.2612	0.2505
1200	0.0092	0.0081	0.2613	0.2613	0.2613	0.2512
1300	0.0095	0.0084	0.264	0.264	0.264	0.2538
1400	0.0091	0.008	0.2625	0.2625	0.2625	0.2529
1500	0.0093	0.0081	0.2601	0.2601	0.2601	0.2536

Tabla 5.9: Balanceo de carga para escenario LL

5.5. Resumen y Conclusiones

Un entorno Grid es de naturaleza heterogénea ya que trabaja con diferentes capacidades de computación, diferentes recursos, diferentes políticas de planificación, pertenecientes a las diversas organizaciones participantes. Esta heterogeneidad representa un reto para, de manera efectiva, realizar un reparto de tareas equilibrado y obtener un tiempo de ejecución aceptable. En este capítulo ha sido explorada la planificación de tareas en Sistemas Grid desarrollando dos planificadores para cada uno de los niveles que componen un Grid.

En la primera parte del capítulo ha sido propuesto un planificador local balanceado para un sistema Grid. Con el desarrollo de este planificador local se han alcanzado tres objetivos. El primer objetivo conseguido ha sido disminuir el tiempo de ejecución de una serie de tareas, con respecto a los algoritmos de planificación tradicionales. Como objetivo segundo, se ha realizado la ejecución de tareas de forma proporcional a la posibilidad de cada recurso. De esta manera, se permite obtener una mejor distribución de carga de tareas entre los recursos. Por último estos recursos, ya existentes, son aprovechados al máximo ya que para la ejecución de tareas se usan ciclos ociosos de esos recursos.

En la segunda parte del capítulo ha sido desarrollado un Meta-Planificador basado en reglas borrosas que es capaz de gestionar, de forma eficiente, la incertidumbre asociada a los recursos formantes de un sistema Grid. Para comprobar su funcionalidad han sido diseñados cuatro escenarios heterogéneos de Sistemas Grid. En esta sección ha sido examinado el comportamiento del Meta-Planificador Borroso con respecto a diversas heurísticas dinámicas tradicionales, en función del tiempo de ejecución y el balanceo de carga (WLB). Los resul-

tados experimentales, mostrados en la sección, demuestran que el tiempo de ejecución para las heurísticas dinámicas tradicionales crece, en gran medida, según aumenta el tamaño de la bolsa de tareas. Sin embargo, con el Meta-Planificador Borroso, este crecimiento es lineal; mientras que, para las heurísticas tradicionales este crecimiento es exponencial. Como se mencionó anteriormente, la propuesta con pre-procesamiento obtiene el mejor tiempo de ejecución para todos los escenarios diseñados, aunque esta diferencia se incrementa cuando el tamaño de la bolsa de tareas es mayor.

Observando las dos propuestas de Meta-Planificador (con pre-procesamiento y sin pre-procesamiento), se comprueba que el tiempo de ejecución para cualquier tamaño de la bolsa de tareas es muy similar entre ellas, con una pequeña mejora si se realiza pre-procesamiento. Sin embargo, el balanceo de carga para el Meta-Planificador borroso con pre-procesamiento es menos favorable que el obtenido por el Meta-Planificador sin pre-procesamiento. Por lo tanto, para conseguir un menor tiempo de ejecución conviene ordenar la bolsa de tareas en orden descendente, aunque esta política empeora el balance de carga. MFS sin pre-procesamiento obtiene el mejor balanceo de carga, pero emplea un tiempo mayor en conseguirlo. Por lo tanto, se recomienda el uso de pre-procesamiento si la carga de los Nodos que forman el Sistema Grid, no es significativa. Si el uso de los Nodos Grid, por parte de sus propietarios, es alto, la mejor opción es no utilizar pre-procesamiento para obtener un mejor balanceo de carga.

Capítulo 6

Detección de defectos en Tableros de Fibra recubiertos con Papel Melamínico

6.1. Introducción

En el capítulo anterior han sido propuestos dos planificadores, un planificador local balanceado y un Meta-Planificador Borroso. Los resultados experimentales demuestran que, ambas propuestas, disminuyen el tiempo de ejecución de una serie de tareas con gran necesidades computacionales y realizan un buen balance de carga entre los recursos y Nodos formantes del Sistema Grid. Además se ha comprobado que el planificador basado en reglas borrosas gestiona, de forma eficiente, la incertidumbre asociada a los recursos de un sistema Grid.

Estos sistemas Grid consiguen grandes cantidades de computación agregando múltiples recursos, así permiten que aplicaciones con altas necesidades de cómputo obtengan resultados o cumplan objetivos en cortos plazos de tiempo. Una de estas aplicaciones son las técnicas de análisis de imágenes [Sonka99]. En esta tesis, las técnicas de análisis de imágenes, van a ser usadas para determinar defectos en Tableros de Fibra recubiertos con Papel Melamínico (TFPM). Un TFPM está formado por una fina capa de madera, o papel simulando madera, normalmente de 0,5 a 1 mm de ancho. Esta capa puede ser obtenida pelando el tronco de un árbol con una cuchilla. Después de este proceso inicial, la fina lámina es pegada en un tablero para producir paneles para cocinas, muebles o decoración interior. La madera es un elemento muy complejo; cada tronco de árbol es diferente y los factores que afectan a las propiedades de la madera no son comprendidas todavía. El principal escollo a abordar tiene que ver con la singularidad de cada lámina de madera, no hay dos láminas idénticas de un tipo de madera determinado, incluso si tienen su origen en el mismo árbol.

Una de las principales tareas que consumen un mayor tiempo en todo el proceso de producción es aquella que permite controlar los defectos. Tradicionalmente, la inspección directa en la fabricación de TFPM ha sido llevada a cabo por personal humano. Este trabajo es subjetivo y depende de la experiencia de los inspectores de calidad [Deng07]. Por lo tanto, las diferencias en la calidad de la inspección visual, así como en la fiabilidad en todo el proceso de inspección puede suponer un problema importante para este tipo de entornos de producción. Los beneficios de la inspección visual automatizada en la industria de TFPM pueden ser notables. En la producción de estos tableros, los volúmenes son enormes y por lo tanto, incluso pequeñas mejoras en la calidad y/o en el rendimiento pueden proporcionar un ahorro considerable.

Durante las últimas décadas, ha crecido una necesidad cada vez mayor de hardware rápido y eficaz para procesamiento. El cuello de botella en el rendimiento de las aplicaciones de análisis de imágenes ha estado causado por la velocidad limitada del hardware. El desarrollo de las Tecnologías de Información ha producido un incremento en la adquisición y uso de recursos y redes heterogéneas en las empresas y entornos de negocio [Huang07]. Grid Computing es una tecnología reciente que conecta recursos heterogéneos y distribuidos a través de redes de alta velocidad [Foster04a]. Con esta tecnología es posible acceder a capacidades computacionales superiores a las ofrecidas por supercomputadores o clusters. Usando esta alta capacidad de computación es posible procesar imágenes de TFPM en un tiempo razonable, sin interrumpir el proceso de producción. Para ello, se usarán los ciclos ociosos de los recursos ya existentes en la compañía, de tal manera que las empresas no tienen que hacer frente al gran coste que supone la adquisición de recursos informáticos de alta capacidad.

Tradicionalmente, el análisis de imágenes ha sido considerada una técnica que requiere altas capacidades de cómputo [Peng08]. Usando computación Grid es posible reducir el tiempo de procesamiento para la detección de defectos en una imagen de un TFPM. Así, este tiempo puede ser acortado usando los recursos ociosos disponibles en la empresa, de tal manera que no es necesario llevar a cabo grandes inversiones en nuevas tecnologías. El tiempo de procesamiento es un factor clave durante la inspección y control de calidad en líneas de producción. Los factores más relevantes que se deben considerar para realizar una inspección en tiempo real son la reducción del tiempo de procesamiento y aumento en la tasa de defectos identificados [Kumar08].

En este capítulo, en primer lugar se realiza un repaso de diversos sistemas de análisis de imágenes aplicados en el campo industrial. A continuación, se expone el planificador local borroso que gestionará los recursos locales pre-existentes, para formar un Desktop Grid, sobre el que se simulará la ejecución de los algoritmos de análisis de imágenes capaces de detectar defectos en tableros de fibra recubiertos con papel melamínico. Con la utilización del sistema Grid, se pretende comprobar que es posible obtener un tiempo de detección, de un posible defecto, mínimo, sin interrumpir el proceso de fabricación. Con el nuevo sistema

de análisis de imágenes será posible aumentar la tasa de tableros con defectos detectados.

6.2. Análisis de imágenes en la industria

El procesamiento de imágenes ofrece soluciones innovadoras para la automatización de procesos industriales. Una gran cantidad de las actividades industriales se han beneficiado de la aplicación de la tecnología de visión por computador en los procesos de fabricación. Estas actividades incluyen, entre otros, la electrónica dedicada a fabricación de componentes, la producción textil, producción de metales, fabricación de vidrio, impresión de los productos y muchos otros. Esta tecnología proporciona, a las industrias que la emplean, un aumento de la productividad, una mejor gestión de la calidad y una serie de ventajas competitivas.

En la literatura, existen diversos trabajos que permitan llevar a cabo inspecciones en una gran variedad de superficies sin la implicación directa de personal humano. El análisis de imágenes juega un papel importante en la inspección visual de las superficies para detectar irregularidades en madera [Silvén03], [Bharati03], acero inoxidable [Killing09], tela [Murino04], papel de aluminio [Zhai09], cerámica [Smith00], u hormigón [Vlahovic12]. Estas superficies son planas, pero también existen trabajos en la detección de defectos en elementos no planos, tales como alimentos [Chen05] o incluso las juntas de soldadura en el interior de algunas estructuras [Neubauer97].

Para captar las imágenes sobre las que aplicar la inspección, la mayoría de los trabajos utilizan cámaras CCD. En [Prasad09], una cámara se utilizó para la obtención de imágenes para evaluar el estado de la herramienta de corte. En [Yin09], con una cámara CCD obtuvieron imágenes de los tejidos para detectar defectos. Un sistema de reconocimiento del iris con imágenes de una cámara CCD fue desarrollado en [Ma03]. Pero también existe una amplia gama de aplicaciones con sensores especiales como rayos X [Leban04], láser [Simonaho04], o híbridos, como en [Lu06] donde se forma el sistema con una cámara CCD y un láser.

La detección de bordes es un proceso que tiene como objetivo la captura de propiedades importantes de los objetos en una imagen. Estas propiedades incluyen discontinuidades en ciertas características de los objetos. El propósito de la detección de bordes es señalar e identificar estas variaciones o discontinuidades. En la literatura, esta técnica es utilizada en varios trabajos. En [Ke09], la detección de bordes se utiliza, en combinación con otras técnicas, para conseguir y predecir los movimientos en los objetos de forma eficaz. Un detector de bordes se diseñó en [Qu09] para distinguir los cambios causados por la unión de varias imágenes. Los efectos de varias técnicas de procesamiento de imágenes, incluida la detección de bordes, fueron analizados en [Ortalana09] para la reconstrucción tomográfica. El detector de bordes de Canny se utiliza en [Lau06] para extraer los bordes de una imagen aérea de una zona determinada para analizar las rutas seguidas por las palomas.

Por lo tanto, es posible desarrollar una serie de algoritmos de análisis de imágenes, usando como entrada las imágenes obtenidas con una cámara CCD, que permitan realizar una inspección de la superficie de tableros de fibra recubiertos con papel meláminico, para la detección e identificación de defectos.

6.3. Planificador Local Borroso

En los sistemas de computación Grid se distinguen dos tipos de planificadores. Los Meta-planificadores, que son sistemas de planificación que gestionan Nodos Grid formados por recursos pertenecientes a distintas organizaciones y los Planificadores Locales, que administran recursos locales, propiedad de un único Nodo Grid u organización.

Dentro de una organización, un planificador local es el encargado de recibir tareas, administrar y seleccionar el recurso adecuado para su ejecución y gestionar los resultados. Los recursos gestionados por un planificador local pueden ser compartidos por sus usuarios para formar parte del sistema Grid. Por lo tanto, la capacidad de estos recursos no es ofrecida completamente al sistema Grid, sino que varía según las necesidades de cada usuario en cada momento.

Como se demuestra en el capítulo anterior, los sistemas expertos basados en reglas borrosas, son capaces de gestionar la incertidumbre asociada a los recursos formantes de un sistema Grid. Por ello, para reducir el tiempo de ejecución de los algoritmos de análisis de imágenes que detectan defectos en TFPM y gestionar los recursos de la organización, normalmente infrautilizados, formantes del sistema, se usará un planificador local basado en reglas borrosas.

6.3.1. Estructura del Planificador Local Borroso

En un sistema de planificación local centralizado existen dos componentes claramente diferenciados: el servidor y los clientes. El servidor contiene toda la lógica del sistema: gestiona los clientes registrados y su estado, las tareas a ejecutar, su planificación, envío de tareas a clientes y es el encargado de recibir y gestionar los resultados. Por otra parte, los clientes, que no realizan ninguna acción de administración, son los encargados de recibir tareas, ejecutarlas y enviar los resultados al servidor.

La parte más importante del servidor es el planificador de tareas. Éste requiere información constante del estado de los recursos para determinar el recurso idóneo encargado de la ejecución de la siguiente tarea. Para gestionar esta información incierta se desarrolla un planificador local borroso, cuya estructura es mostrada en la figura 6.1. En la figura se observa que el planificador recibe como entrada la información del estado actual de cada uno de los clientes que forman el sistema Grid. Esta información es utilizada por el planificador

borroso para determinar que recurso o cliente es el adecuado para ejecutar la siguiente tarea.

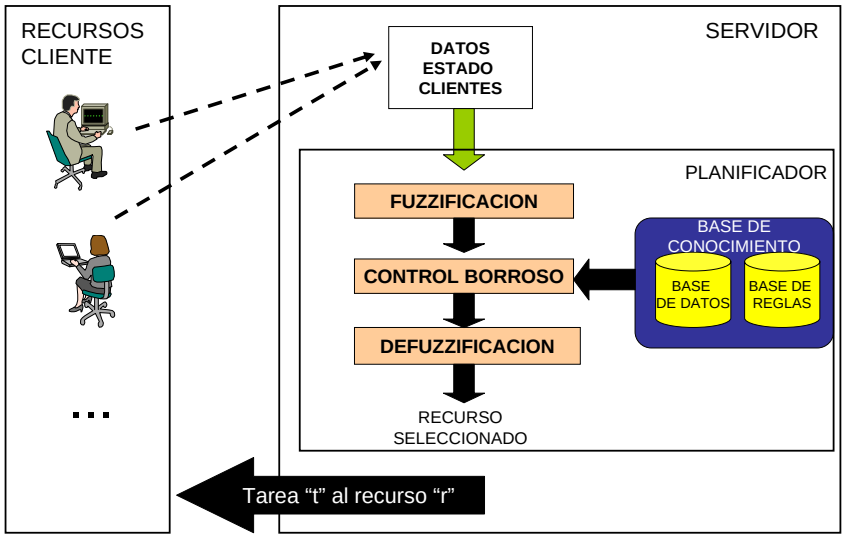


Figura 6.1: Estructura del planificador local borroso

La información relacionada con el estado de cada recurso es enviada al planificador local borroso de forma continuada. Cada recurso informa al planificador sobre su estado según dos parámetros:

- Procesadores libres (PF). Por el uso del propietario del recurso, éste puede tener una tasa de capacidad de procesamiento ocupada ejecutando tareas propias. Esta entrada, indica el porcentaje de procesadores libres pertenecientes a un mismo recurso. La ecuación de normalización para esta entrada es:

$$PF(i) = \frac{ProcesadoresLibres(i)}{ProcesadoresTotales(i)} \tag{6.1}$$

  donde i es el identificador de un recurso del Sistema Grid.

- Procentaje de MIPS libres (MIPS). Expresa la tasa de MIPS ociosos o libres que tiene un recurso y que puede ser utilizada para ejecutar tareas pertenecientes al sistema Grid. La ecuación de normalización para este parámetro de entrada es:

$$MIPS(i) = \frac{MipsLibres(i)}{TotalMips(i)} \tag{6.2}$$

  donde i es el identificador de un recurso del Sistema Grid.

Usando sólo los dos parámetros anteriores, se produce una sobrecarga de recursos, ya que los mejores recursos disponibles son elegidos para la ejecución de tareas del Grid. Para no

utilizar siempre los mismos recursos, el servidor contabiliza tres medidas históricas sobre la utilización de cada recurso. Estos parámetros son:

- Tamaño de tareas ejecutadas (LTE). Cada tarea a ejecutar tiene un tamaño diferente. Este parámetro indica el tamaño total de las tareas ejecutadas por un recurso en relación con el total de tareas ejecutadas. La ecuación de normalización para este parámetro de entrada es:

$$LTE(i) = \frac{TamTareasEjecutadas(i)}{\sum_{i=1}^{n} TamTareasEjecutadas} \tag{6.3}$$

  donde i es un recurso Grid y n es el número total de recursos del Grid.

- Tiempo empleado en su ejecución (TET). Tiempo total durante el cual un recurso ha estado ejecutando tareas pertenecientes al sistema Grid en relación con el tiempo total empleado por todos los recursos que forman el sistema. La ecuación de normalización para esta entrada es:

$$TET(i) = \frac{TiempoEmpleado(i)}{\sum_{i=1}^{n} TiempoEmpleado} \tag{6.4}$$

  donde i es un recurso Grid y n es el número total de recursos Grid.

- Numero de tareas ejecutadas (TE). Cuenta el número de tareas finalizadas por un recurso con respecto al total de tareas finalizadas por todos los recursos formantes del sistema Grid. La ecuación de normalización para esta entrada es:

$$TE(i) = \frac{TareasFinalizadas(i)}{\sum_{i=1}^{n} TareasFinalizadas} \tag{6.5}$$

  donde i es un recurso Grid y n es el número total de recursos del Grid.

6.3.1.1. Conjuntos Borrosos

Estas cinco medidas son utilizadas por el planificador local borroso para determinar el recurso destinado a ejecutar la tarea actual. Para estas cinco entradas se calcula el grado de pertenencia a cada conjunto borroso. Se observa en la figura 6.2, que el conjunto borroso para las variables de entrada es de tipo triangular y está formado por tres etiquetas lingüísticas (Bajo, Medio y Alto). La variable de salida es definida como Recurso Seleccionado (RS), indica la tasa de selección para cada uno de los recursos. El recurso que obtiene la salida RS mayor será elegido para ejecutar la siguiente tarea del sistema Grid. Esta variable de salida, figura 6.3, contiene cinco conjuntos difusos de tipo triangular.

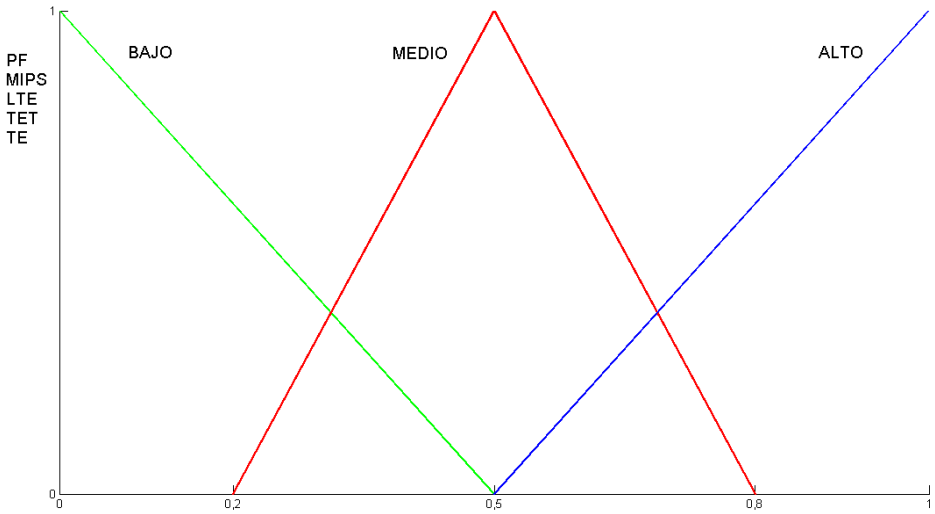


Figura 6.2: Función de Pertenencia para la entrada

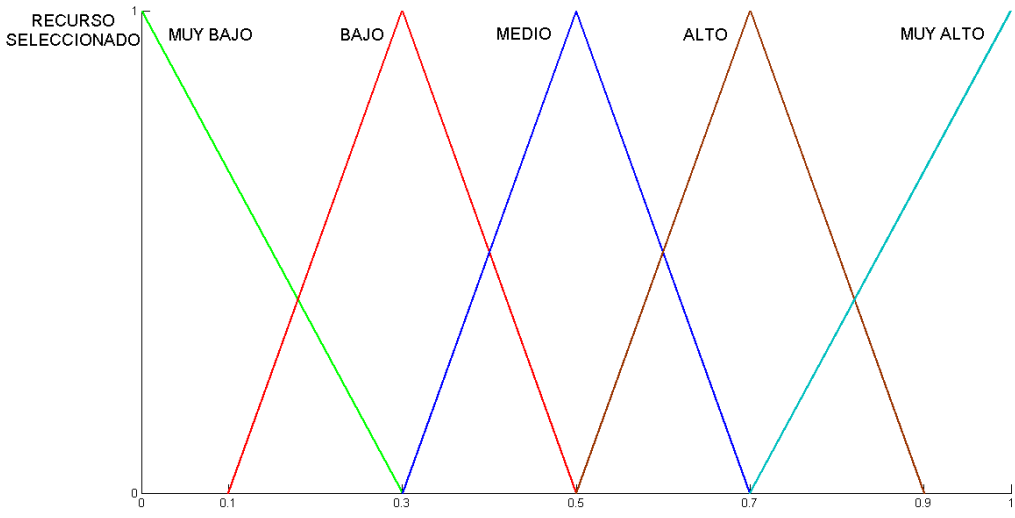


Figura 6.3: Función de Pertenencia para la salida

6.3.1.2. Reglas Borrosas

Las reglas de inferencia borrosas son desarrolladas utilizando las cinco variables de entrada y la variable de salida definidas previamente. Todas tienen el mismo peso y usan el conector AND. Las reglas usadas en el planificador local borroso son las siguientes:

- Regla 1. *si* **PF** *es BAJO y* **MIPS** *es BAJO y* **LTE** *es BAJO y* **TET** *es BAJO y* **TE** *es BAJO entonces* **RS** *es BAJO*
- Regla 2. *si* **PF** *es BAJO y* **MIPS** *es BAJO y* **LTE** *es MEDIO y* **TET** *es MEDIO y* **TE** *es MEDIO entonces* **RS** *es BAJO*
- Regla 3. *si* **PF** *es BAJO y* **MIPS** *es BAJO y* **LTE** *es ALTO y* **TET** *es ALTO y* **TE** *es ALTO entonces* **RS** *es MUYBAJO*
- Regla 4. *si* **PF** *es MEDIO y* **MIPS** *es MEDIO y* **LTE** *es BAJO y* **TET** *es BAJO y* **TE** *es BAJO entonces* **RS** *es MEDIO*
- Regla 5. *si* **PF** *es MEDIO y* **MIPS** *es MEDIO y* **LTE** *es MEDIO y* **TET** *es MEDIO y* **TE** *es MEDIO entonces* **RS** *es BAJO*
- Regla 6. *si* **PF** *es MEDIO y* **MIPS** *es MEDIO y* **LTE** *es ALTO y* **TET** *es ALTO y* **TE** *es ALTO entonces* **RS** *es BAJO*
- Regla 7. *si* **PF** *es ALTO y* **MIPS** *es ALTO y* **LTE** *es BAJO y* **TET** *es BAJO y* **TE** *es BAJO entonces* **RS** *es MUYALTO*
- Regla 8. *si* **PF** *es ALTO y* **MIPS** *es ALTO y* **LTE** *es MEDIO y* **TET** *es MEDIO y* **TE** *es MEDIO entonces* **RS** *es ALTO*
- Regla 9. *si* **PF** *es ALTO y* **MIPS** *es ALTO y* **LTE** *es ALTO y* **TET** *es ALTO y* **TE** *es ALTO entonces* **RS** *es MEDIO*

Para realizar la tarea de defuzzificacion y obtener la salida se emplea el método del centroide.

6.4. Inspección de Superficies

6.4.1. Proceso de Identificación de Defectos

En los sistemas de inspección de calidad en tiempo real, la tasa de defectos identificados y el tiempo empleado en detectar un defecto son dos factores básicos. En esta tesis, el tiempo de respuesta para determinar si un TFPM es defectuoso es un factor clave. Si este tiempo de identificación es minimizado, el volumen de producción se incrementará y por lo tanto, la calidad general mejorará.

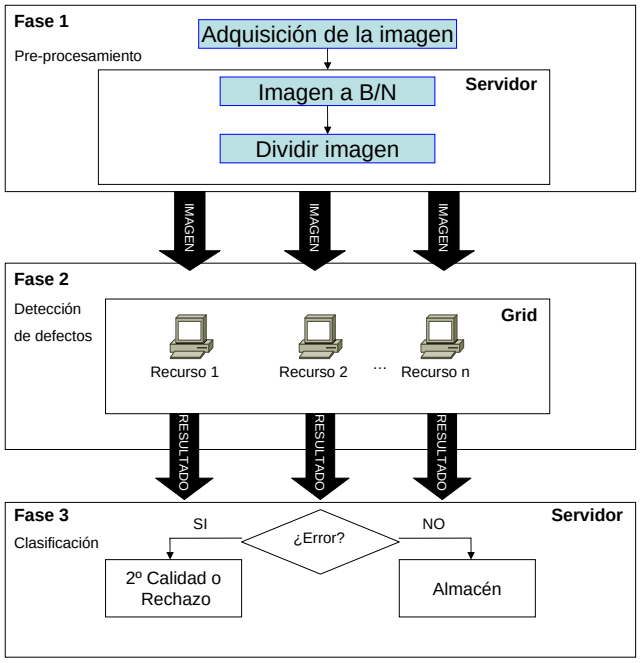


Figura 6.4: Proceso de detección de defectos

La inspección visual automática consiste en un proceso formado por tres pasos: adquisición de imágenes, procesamiento y análisis. En la figura 6.4, se muestran estos tres pasos para realizar el proceso de identificación de defectos. En el primer paso, es obtenida una imagen de un TFPM. Esta imagen es recibida en un ordenador central o servidor, que realiza el primer pre-procesamiento. En este paso, la imagen es transformada a formato blanco y negro (b/n) para reducir su tamaño. De este modo se reduce el tiempo de envío de la imagen a los recursos formantes del sistema Grid. Dentro de esta fase de pre-procesamiento, la imagen es dividida, de manera que la imagen a procesar sea enviada a distintos recursos ociosos del sistema Grid. Como se muestra en la figura 6.5 no es realizada una división sencilla, sino que se realiza superponiendo cada división, añadiendo el 10 %. Se realiza este solapamiento con el fin de no perder la precisión, ya que algunos defectos pueden ser muy pequeños.

En el siguiente paso, cada división es enviada al recurso que determine el planificador. Cada recurso ejecutará diversos algoritmos de detección de defectos asociados al tipo de TFPM que está siendo procesado en este momento. Debido al gran número de diseños papel melamínico, es necesario un algoritmo específico para cada tipo. Esto no constituye un inconveniente importante ya que el tipo de papel que se está procesando se sabe siempre de antemano.

Finalmente, cada recurso informa al servidor si ha identificado un defecto. Si un recurso

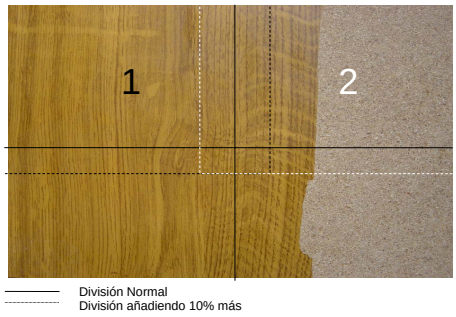


Figura 6.5: División con superposición

detecta un defecto en el tablero examinado, el proceso de detección en todos los recursos será suspendido. A continuación el tablero en cuestión será etiquetado como defectuoso, por lo que es descartado para su embalaje, pasando a reciclado. En caso de que ningún recurso informe al servidor de la detección de un defecto, el tablero continuará a través de la línea de producción y será finalmente enviado al almacén de producto terminado.

Debido al gran tamaño de los tableros fabricados por la empresa colaboradora, los experimentos se han realizado sobre muestras de defectos realizas en laboratorio, que reproducen al 100 % los defectos obtenidos en la cadena de producción. Las imágenes, que han sido usadas para este trabajo, de las diferentes muestras han sido obtenidas usando una cámara CCD.

6.4.2. Clasificación de defectos

Los defectos considerados en este trabajo, suman aproximadamente el 95 % de los defectos detectados y catalogados en la empresa colaboradora. En la tabla 6.1, se indica el nombre de cada defecto, una breve descripción de como se produce y su forma o apariencia. Cada uno de estos defectos puede ser de dos tipos: segunda calidad o rechazo.

Un tablero es considerado de segunda calidad si el defecto ocupa un espacio inferior al 10 % del tablero total. Un tablero de este tipo es enviado para su distribución a un precio inferior. En caso contrario, se rechaza y se envía al área de reciclaje. En la figura 6.6 se presenta una imagen de cada uno de los defectos y sus tipos, de segunda calidad y de rechazo, tratados en esta tesis y una imagen de un tablero sin ningún tipo de defecto.

6.4.2.1. Técnicas de análisis de imágenes

Debido a la alta variabilidad en el material de TFPM, es probable que sea imposible encontrar rasgos capaces de diferenciar las zonas defectuosas de las correctas. Existe una inmensa y variable cantidad de diseños de papel melamínico. Puesto que en este trabajo

Nombre	Descripción	Forma y Apariencia
Papel Roto	El papel melamínico se ha rasgado al ser pegado al tablero. El tablero contiene una grieta que aparece al ser pegado el papel.	Línea discontinua y no recta. Puede tomar cualquier dirección. No es detectado al tacto. Se confunde fácilmente con una veta.
Papel Desplazado	El papel se ha desplazado al ser pegado al tablero. Un lateral del tablero aparece sin papel.	Forma una línea que cubre todo el tablero paralela a la veta. Línea irregular, con posibilidad de curvas y recortes. El defecto es apreciable al tacto.
Restos de Prensado	Resto de papel de un diseño anteriormente usado. Aparece en el nuevo tablero prensado.	Ambos papeles son distintos. Forma irregular. Puede aparecer en cualquier parte. No es apreciable al tacto.
Restos de Resina	Resto de resina o pegamento usado para pegar el papel al tablero.	Manchas pequeñas de color gris. Forma redondeada. Puede aparecer en cualquier parte. Apreciable al tacto.

Tabla 6.1: Clasificación de defectos

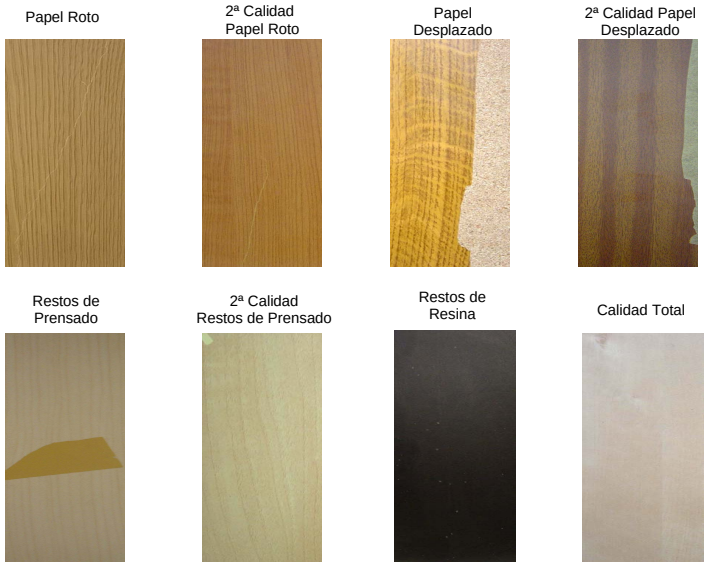


Figura 6.6: Tipos de defectos

ID	Nombre	Descripción	Técnicas usadas
1	Papel Roto Rechazo	Defecto ocupa más del 10 % del tablero	Filtro Prewitt + Detector de Bordes Canny, umbral alto=0,15, umbral bajo=0,06
2	Papel Roto 2ª Calidad	Defecto ocupa menos del 10 % del tablero	Dilatar + Detector de Bordes Canny, umbral alto=0,9, umbral bajo=0,7
3	Papel Desplazado Rechazo	Defecto ocupa más del 10 % del tablero	Filtro Prewitt + Detector de Bordes Canny, umbral alto=0,4, umbral bajo=0,16
4	Papel Desplazado 2ª Calidad	Defecto ocupa menos del 10 % del tablero	Dilatar + Detector de Bordes ZeroCross, umbral=0,005.
5	Restos de Resina	Defecto ocupa menos del 10 % del tablero	Detector de Bordes ZeroCross, umbral=0,005.
6	Restos de Prensado Rechazo	Defecto ocupa más del 10 % del tablero	Detector de Manchas: convolución derivada segunda de gaussiana varias escalas. Elección de mínimos locales como defecto
7	Restos de Prensado 2ª Calidad	Defecto ocupa menos del 10 % del tablero	

Tabla 6.2: Defectos y su procesado

se va a analizar la superficie recubierta por este tipo de papel, es necesario utilizar un determinado filtro o algoritmo para cada uno de los diseños que se disponen. En algún caso, el algoritmo usado será el mismo pero cambiando el valor de sus parámetros de entrada. En la tabla 6.2, son mostrados cada uno de los ejemplos estudiados en esta tesis, si el defecto pertenece a la categoría de segunda calidad o de rechazo y las técnicas usadas para detectar el defecto.

Como se observa en la tabla 6.2, para los tipos *Papel Roto Rechazo* (1) y *Papel Desplazado Rechazo* (3) se sigue el mismo proceso para la detección del posible defecto. En primer lugar a la imagen de entrada se realiza un filtro de Prewitt, seguido de una detección de bordes usando el método de *Canny*. La diferencia de proceso entre ambos defectos es el umbral aplicado al detector de bordes Canny. Para el defecto *Papel Roto 2ª Calidad* (2), se realiza una dilatación de la imagen inicial, y a continuación a la imagen resultante se le aplica el detector de bordes Canny con los umbrales indicados en la tabla. Para detectar el defecto *Papel Desplazado 2ª Calidad* (4) se realiza una dilatación de la imagen inicial, seguido de la aplicación del detector de bordes ZeroCross con umbra igual a 0,005. En el defecto *Restos de Resina* no es necesario aplicar ningún pre-proceso, directamente es empleado el detector de bordes ZeroCross con el umbral indicado en la tabla. Para los dos últimos defectos *Restos de Prensado Rechazo* se utiliza un detector de manchas.

6.4.3. Ejemplo de identificación de defectos

En la sección "Proceso de Identificación de Defectos", ha sido expuesto el proceso de identificación de defectos incluyendo el sistema Grid. En este apartado es mostrado el proceso de detección de defectos relacionado con el procesamiento de imágenes, sin importar el sistema sobre el que se realiza el procesamiento.

Como se mencionó anteriormente, en función del tipo de TFPM a examinar, es aplicado un

determinado algoritmo de filtro o de detección, por lo que la primera parte del proceso de detección es diferente para cada tipo de diseño, mientras que la segunda parte es la misma para todos los tipos.

La figura 6.7 muestra el proceso de detección de defectos para el tipo "Papel Desplazado Rechazo", ID #3 :

- En primer lugar la imagen es transformada a formato blanco y negro (b/n). Esta tarea es realizada en el servidor.
- En el recurso encargado de procesar la imagen (b/n) ya se encuentran los algoritmos de procesamiento de imágenes. Para este ejemplo, el primer paso es aplicar un filtro de Prewitt.
- En el siguiente paso, se aplica el detector de bordes "canny", con unos valores determinados para el umbral, sobre la imagen resultante del filtrado.
- Para finalizar, en una segunda fase, es comprobado el número de píxeles que forman parte del borde. Si este número de píxeles es mayor que un valor límite inferior, el tablero es considerado defectuoso. Por lo tanto, es necesario distinguir entre rechazo y 2ª calidad. Si el número de píxeles borde es mayor que un 10 %, el tablero se rechaza y se envía al área de reciclaje. De lo contrario, el tablero se considera de 2ª calidad, enviándose al almacén para su distribución a un precio inferior.

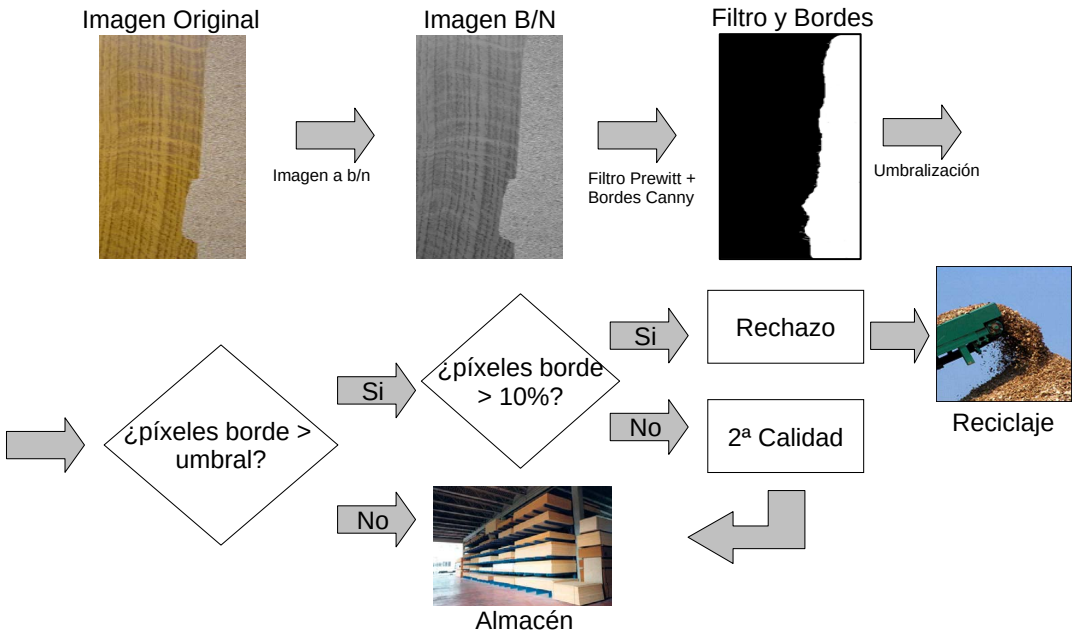


Figura 6.7: Procesamiento de imágenes para el defecto #3.

6.5. Experimentos y Resultados

En esta sección son mostrados los resultados experimentales. Estos resultados se basan en un análisis de texturas en la superficie de tableros de fibra recubiertos con papel melamínico, de manera que sea evaluado el método propuesto. Debido a la falta de recursos reales para abordar este problema y con el fin de facilitar el desarrollo de los experimentos, éstos han sido divididos en dos etapas. Cada una de las etapas contiene uno de los objetivos a cumplir:

- En la primera etapa, el objetivo consiste en obtener la mayor tasa de defectos identificados, estudiando las distintas resoluciones ofrecidas por la cámara CCD.
- En la segunda fase, el objetivo a cumplir es la obtención de un menor tiempo en el control de calidad de tableros. Para ello se simulará el planificador local borroso, descrito anteriormente, usando la plataforma de simulación de sistemas Grid "GridSim" [Sulistio08]. Para este experimento se simularán distintos tipos de ordenadores de escritorio, a los que se les aplicará un porcentaje de uso, modelado siguiendo una distribución uniforme, para de esta forma simular la utilización por sus propietarios.

6.5.1. Tasa de defectos

En esta sección se muestra la tasa de defectos identificados para cada uno de los defectos anteriormente descritos. Para cada uno de estos defectos se dispone de 25 muestras de tableros con un único defecto. Cada una de estas muestras ha sido fotografiada usando cinco resoluciones distintas. Antes del procesamiento, cada imagen ha sido dividida siguiendo el proceso descrito en la figura 6.5. Estos resultados son mostrados en las siguientes tablas. Cada tabla está compuesta por las siguientes columnas:

- Resolución. Indica la resolución utilizada para tomar la foto del tablero. Medida en píxeles.
- Muestras. Número de muestras distintas para el defecto tratado.
- FP (Falso Positivo). Un FP se produce cuando el algoritmo detecta un defecto en una zona donde no existe ningún defecto.
- FN (Falso Negativo). Un falso negativo [1] indica que un defecto existente no ha sido identificado.
- Tasa de identificación: Indica el porcentaje de defectos detectados con respecto al total. Un valor de 100 % indica que todos los defectos existentes en las muestras, para el tipo de defecto tratado, han sido detectados.

[1] Un FN se considera un error más grave que un FP, porque si un tablero defectuoso llega a un cliente implica un doble coste, ya que la empresa debe hacer frente a los costes de devolución.

Estas tablas no muestran el tiempo de ejecución, ya que nuestro objetivo, en esta primera fase, es la obtención de una resolución que maximice la tasa de defectos identificados. Por tanto, es importante en esta primera fase de resultados obtener una alta tasa de defectos detectados.

En la tabla 6.3, son mostrados los resultados para el defecto #1, Papel Roto Rechazo. Con la mayor resolución (2048x1536), el algoritmo produce un FP. Esto es debido a que usando una alta resolución, el algoritmo confunde vetas del papel melamínico con un defecto de papel roto. Con una resolución intermedia la tasa de identificación de defectos crece hasta llegar al 100 %. Con una menor resolución, el algoritmo detecta defectos donde no existen. En concreto, para la resolución (960x1280) se producen 14 FP y 2 FN, y para la menor resolución (480x640) se obtienen 18 FP. Por tanto para este defecto, las dos resoluciones intermedias (2048x1360) y (1200x1600) son las que obtienen un acierto total al tener una tasa de detección de defectos del 100 %.

Resolución	Núm. Muestras	FP	FN	Tasa de identificación
2048x1536	25	1	0	96 %
2048x1360	25	0	0	100 %
1200x1600	25	0	0	100 %
960x1280	25	14	2	36 %
480x640	25	18	0	28 %

Tabla 6.3: Resultados para el defecto #1

Los resultados de detección para el defecto #2 son mostrados en la tabla 6.4. Este defecto es etiquetado como Papel Roto 2ª Calidad, por lo tanto el defecto ocupa un tamaño menor al 10 % de la extensión del tablero. Por este motivo, la dificultad para detectar el defecto es mayor. Las resoluciones altas (2048x1536) y (2048x1360) obtienen una tasa de identificación baja. Para la resolución mayor la tasa de identificación supera por poco el 50 % y el número de falsos positivos FP es alto. Esto es debido a que las vetas son similares al defecto de papel roto y son enmascaradas como defecto. Por lo tanto, para este defecto la mejor tasa de identificación, un 68 %, es obtenida con una resolución media (1200x1600).

Resolución	Núm. Muestras	FP	FN	Tasa de identificación
2048x1536	25	12	0	52 %
2048x1360	25	7	3	60 %
1200x1600	25	6	2	68 %
960x1280	25	8	1	64 %
480x640	25	10	3	48 %

Tabla 6.4: Resultados para el defecto #2

En la tabla 6.5, se muestra la tasa de defectos identificados para el defecto #3, Papel Desplazado Rechazo. En este caso, para resoluciones altas son obtenidos los mejores resultados. En concreto, con las resoluciones (2048x1536), (2048x1360) y (1200x1600) se consigue una tasa de detección de detección del 96 %. Para el resto de resoluciones la tasa de detección es inferior, un 92 % de defectos detectados para una resolución de (960x1280) y un escaso 36 % para la resolución inferior (480x640). En este caso es considerada la resolución media (1200x1600) como la mejor, puesto que con un tamaño de imagen menor es capaz de obtener la misma tasa de defectos que las resoluciones mayores y, a diferencia de las otras dos resoluciones con la misma tasa de identificación, no produce ningún FN.

Resolución	Núm. Muestras	FP	FN	Tasa de identificación
2048x1536	25	0	1	96 %
2048x1360	25	0	1	96 %
1200x1600	25	1	0	96 %
960x1280	25	0	2	92 %
480x640	25	3	13	36 %

Tabla 6.5: Resultados para el defecto #3

En la tabla 6.6, son mostrados los resultados para el defecto #4, Papel Desplazado 2ª Calidad. En todas las resoluciones usadas se logra una tasa de detección del 100 %. Esta tasa de acierto tan elevada para todas las resoluciones se debe a la diferencia entre la rugosidad de la superficie del tablero y el papel melamínico, lo que hace que la detección sea más sencilla.

Resolución	Núm. Muestras	FP	FN	Tasa de identificación
2048x1536	25	0	0	100 %
2048x1360	25	0	0	100 %
1200x1600	25	0	0	100 %
960x1280	25	0	0	100 %
480x640	25	0	0	100 %

Tabla 6.6: Resultados para el defecto #4

Para el defecto #5, Restos de Resina, según es presentado en la tabla 6.7, se consigue la misma tasa de identificación para tres resoluciones. Para la resolución (2048x1536) se obtiene una tasa de acierto del 84 %, con tres falsos positivos y un falso negativo. Para la resolución (1200x1600) una tasa del 84 % es lograda, con cuatro falsos positivos y cero falsos negativos. También se obtiene una tasa de 84 % con la resolución (960x1280), con dos falsos positivos y dos falsos negativos. Tal y como se ha explicado anteriormente un FN produce un incremento de costes, por lo que es considerado un fallo "menor" un FP.

Según esto los mejores resultados son logrados con la resolución (1200x1600), puesto que no produce ningún FN.

Resolución	Núm. Muestras	FP	FN	Tasa de identificación
2048x1536	25	3	1	84 %
2048x1360	25	4	2	76 %
1200x1600	25	4	0	84 %
960x1280	25	2	2	84 %
480x640	25	2	5	72 %

Tabla 6.7: Resultados para el defecto #5

En la tabla 6.8, se observan los datos para el defecto #6, Restos de Prensado Rechazo. Para este defecto dos resoluciones obtienen una tasa de detección del 100 %, (1200x1600) y (960x1280). Con resoluciones altas, los restos de prensado no son identificados, obteniendo dos falsos positivos. Con la resolución inferior (480x640), la tasa de identificación desciende hasta el 64 %.

Resolución	Núm. Muestras	FP	FN	Tasa de identificación
2048x1536	25	2	0	92 %
2048x1360	25	2	1	88 %
1200x1600	25	0	0	100 %
960x1280	25	0	0	100 %
480x640	25	6	3	64 %

Tabla 6.8: Resultados para el defecto #6

En la tabla 6.9, aparece la tasa de defectos identificados para el defecto #7, Restos de Prensado 2ª Calidad. Como en el caso anterior, es obtenida una tasa del 100 % para las resoluciones (1200x1600) y (960x1280). Usando otras resoluciones la tasa de detección baja hasta el 76 %.

Resolución	Núm. Muestras	FP	FN	Tasa de identificación
2048x1536	25	16	0	36 %
2048x1360	25	6	0	76 %
1200x1600	25	0	0	100 %
960x1280	25	0	0	100 %
480x640	25	0	9	64 %

Tabla 6.9: Resultados para el defecto #7

En la tabla 6.10 se señala el promedio de tasa de identificación de defectos, agrupado por resolución, para todos los experimentos llevados a cabo en este apartado. Se obtiene una

tasa de defectos identificados del 92,57 %, con la resolución de 1200x1600 píxeles.

Resolución	Tasa Media de Identificación
2048x1536	79,42 %
2048x1360	85,14 %
1200x1600	**92,57** %
960x1280	82,28 %
480x640	58,85 %

Tabla 6.10: Tasa media de defectos identificados para cada resolución

6.5.2. Tiempo de detección de defectos

En la sección anterior, usando una resolución de 1600x1200 píxeles se ha obtenido la mayor tasa de identificación de defectos, como se observa en la tabla 6.10. Ahora, en este apartado, el trabajo se centra en el segundo objetivo, obtener un tiempo de inspección de tableros mínimo. Para esto se realiza una simulación de la ejecución de los algoritmos de análisis de imágenes sobre un sistema Desktop Grid. En estos experimentos, el objetivo es obtener un tiempo de detección menor que el obtenido actualmente en la inspección manual. Según información proporcionada por la empresa, el tiempo máximo empleado por un operador para revisar y clasificar un tablero (2ª Calidad, Calidad Total o Rechazo) es de unos 40 segundos.

Para minimizar el tiempo de proceso de las técnicas de análisis de imágenes se simula un sistema Desktop Grid formado por recursos locales, que son gestionados por el planificador local borroso descrito en la sección 6.3. Para simular la heterogeneidad asociada a los sistemas Grid, a cada recurso se le ha asignado un porcentaje de uso correspondiente a la utilización de parte de las capacidades del recurso por sus propietarios. De esta manera se obtienen dos de las características principales de los sistemas Desktop Grid: recursos heterogéneos y uso por parte de los propietarios de los recursos. Este proceso vacío ocupa entre un 20 % y un 50 % de utilización de la capacidad total disponible en el recurso, es modelado usando una distribución uniforme y se mantiene constante durante todo el experimento.

El tiempo de ejecución es presentado agrupado por el número de recursos formantes del sistema Desktop Grid. La simulación para cada tipo de defecto es realizada cinco veces, cambiando en cada prueba el número de recursos disponibles. Este número de recursos utilizados oscila entre 20 y 100.

En la figura 6.8, se muestran los tiempos de detección para cada defecto con cada grupo de recursos formantes del Desktop Grid: 20, 40, 60, 80 y 100. Los algoritmos que emplean más tiempo son el asociado con papel roto y el de restos de papel prensado. Para detectar estos

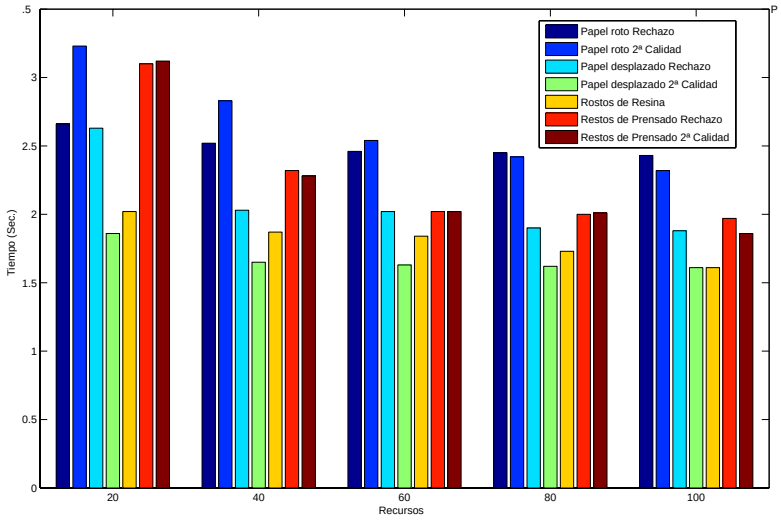


Figura 6.8: Tiempo de detección por defecto y número de recursos

defectos, los algoritmos necesitan operaciones de cómputo más costosas para confirmar la existencia de un defecto. El tiempo máximo para estos dos algoritmos es, aproximadamente, de 3,23 segundos, con 20 recursos y 2,20 segundos para 100 recursos. El resto de los defectos siempre se detectan en un tiempo menor a 2,5 segundos para cualquier combinación de recursos. La detección del defecto Papel Desplazado, necesita un tiempo menor ya que la diferencia de textura entre el papel melamínico y el tablero es enorme. En promedio, la tasa de mejora en tiempo entre 20 y 100 recursos es de un 25 %.

La tabla 6.11 muestra un resumen de la tasa de defectos identificados en la sección anterior. También se presenta la media de tiempo obtenida para todos los defectos en la simulación con 100 recursos. El tiempo promedio para todos los algoritmos aplicados es de 1,95 segundos, en el caso de utilizar 100 recursos. Como se observa en la tabla 6.11, el uso de un único recurso para la detección de defectos es mucho menos eficiente que el uso del Sistema Grid, ya que el tiempo medio de ejecución de un algoritmo en un único recurso de escritorio es de 129 segundos.

De acuerdo con los datos proporcionados por la empresa, su tasa de identificación defectos está cercana al 83 %. En este trabajo, se ha obtenido una mejora significativa, lo cual repercutiría directamente en ahorros de costos para la empresa, ya que los productos defectuosos no detectados son rechazadas por los clientes lo que significa que la empresa aumenta su coste de no calidad.

Estas pruebas se realizaron con imágenes tomadas de tableros con defectos conocidos de antemano. Por esta razón un sólo algoritmo de detección ha sido aplicado a cada defecto. En el sistema futuro, sería necesario aplicar los siete algoritmos de análisis de imágenes a

ID	FP	FN	Tasa de Identificación (%)	Tiempo 100 Recursos	Tiempo 1 Recurso
1	0	0	100 %	2,43 segundos	89,67 segundos
2	6	2	68 %	2,32 segundos	216,77 segundos
3	1	0	96 %	1,88 segundos	82,76 segundos
4	0	0	100 %	1,61 segundos	157,99 segundos
5	4	0	84 %	1,61 segundos	20,02 segundos
6	0	0	100 %	1,97 segundos	179,49 segundos
7	0	0	100 %	1,86 segundos	157,22 segundos
Media	-	-	**92,57** %	**1,95** segundos	129 segundos

Tabla 6.11: Tasa de identificación y tiempo medio de detección

un mismo tablero, ya que cada tablero puede contener diferentes tipos de defectos. En un sistema Desktop Grid con 100 recursos y de acuerdo a los tiempos indicados, la suma de tiempos de los siete algoritmos que se exponen necesitan aproximadamente 14 segundos para analizar las imágenes de un tablero de fibra. La mejora en el tiempo y la confiabilidad con respecto a la inspección visual humano (40 segundos) es enorme. Por lo tanto, existe una evidencia clara que apoya nuestro objetivo: el uso de un sistema de computación Desktop Grid, con planificador local borroso, aumenta la eficacia del sistema de gestión de calidad en este sector, al aumentar la calidad global del producto final y reducir el coste asociado al control de calidad.

6.6. Resumen y Conclusiones

En este capítulo, es diseñado e implementado un sistema de software que identifica defectos en tableros de fibra recubiertos con papel melamínico. Este sistema aumenta el porcentaje de defectos detectados, con respecto a los datos reales proporcionados por la empresa. Utilizando un sistema Desktop Grid, con planificador local borroso, el tiempo en el proceso de control de calidad es reducido.

Este sistema es totalmente viable y al mismo tiempo eficiente de acuerdo con los recursos necesarios para su plena aplicación. Con la utilización de este sistema en una empresa, ésta se ve recompensada con un incremento importante en la calidad y en la productividad. El sistema propuesto en este trabajo es capaz de reducir, al mismo tiempo, los costes de calidad y no calidad, ya que elimina la necesidad de implementar y mantener un sistema de control de calidad manual. No sólo se presenta esta reducción en los costes. El aumento de los índices de calidad y la reducción de los errores externos (los que se detectan fuera de la fábrica, en el cliente) mejora el desempeño global de calidad de la empresa en términos de eficacia de mercado. Los efectos directos sobre el ahorro de costes y aumento de la

productividad tendrá repercusiones sobre la posición de la empresa en el mercado a largo plazo. Aparte de estos efectos positivos, la utilización de un sistema Desktop Grid en este experimento, en particular, ofrece un resultado aún más poderoso que todos los anteriores, ya la eficiencia y eficacia se pueden obtener gracias al uso de los recursos existentes (recursos ociosos utilizados en el sistema Grid), por lo que no se necesitan nuevas inversiones.

Parte IV

Conclusiones, Líneas Futuras y Publicaciones Generadas

Capítulo 7

Conclusiones y Líneas Futuras de Investigación

En este capítulo se resumen las conclusiones de esta tesis, las publicaciones obtenidas y las líneas de investigación que permitirán continuar, en el futuro, el estudio comenzado en esta tesis.

7.1. Conclusiones y principales contribuciones

Los sistemas Grid son una plataforma atractiva para ejecutar cálculos de gran tamaño. Debido a su gran capacidad y escalabilidad, estos sistemas se pueden aplicar en una gran cantidad de campos con altas necesidades de cómputo. Estas aplicaciones, que no hace mucho requerían muchos días, meses e incluso años para obtener algún resultado, pueden, usando estos sistemas, finalizar y conseguir sus objetivos en tiempos muy inferiores. Una variación de los sistemas Grid son los Desktop Grid. Ambos sistemas comparten el mismo objetivo: aprovechar de la mejor manera los recursos informáticos que existen. Sin embargo, en un Desktop Grid estos recursos son ofrecidos por voluntarios, por lo que la infraestructura formada es más heterogénea, dinámica y escalable que en los sistemas Grid tradicionales.

Uno de los principales problemas asociados a cualquier sistema distribuido es la planificación de tareas. Una buena planificación de tareas permite obtener un mejor rendimiento de los recursos del sistema y por tanto mejores resultados. Debido a las características de los recursos que forman un sistema Desktop Grid, este problema se ve agravado. Estos recursos, son muy variados en sus propiedades y altamente dinámicos, puesto que su uso es compartido.

Para administrar y planificar un conjunto de tareas, en esta tesis, se propone un planifi-

cador local balanceado (capítulo 5). Este planificador permite la ejecución de un flujo de trabajo, con tareas dependientes o independientes, sobre un conjunto de recursos heterogéneos, equilibrando la carga de trabajo entre dichos recursos. Al distribuir las tareas de una forma proporcional a las características de cada recurso, se consigue que los recursos menos eficientes finalicen su trabajo en un tiempo adecuado, de manera que no retrasen la ejecución total del flujo de trabajo. Mediante las simulaciones llevadas acabo se ha comprobado que esta nueva propuesta de planificador local, permite minimizar el tiempo de ejecución de un flujo de trabajo.

Como se mencionó anteriormente, los recursos participantes en los sistemas Desktop Grid, son altamente dinámicos y de naturaleza imprecisa. Para gestionar estas características y conseguir una planificación de tareas eficiente se propone un Meta-Planificador basado en Reglas Borrosas (capítulo 5), que gestionará una serie de Nodos Grid. Esta planificación de tareas eficiente está basada en dos objetivos, minimizar el tiempo de ejecución y realizar una ejecución de tareas balanceada, implicando en la ejecución a todos los nodos que forman el sistema equilibrando la carga de trabajo. El sistema borroso asociado al Meta-Planificador contiene sus entradas agrupadas en dos grupos. Un grupo, de dos entradas, relacionadas con el estado actual del Nodo Grid y otro grupo, de tres entradas, relacionadas con el histórico de tareas ejecutadas por el Nodo. Los resultados de la evaluación demuestran que el mecanismo de planificación propuesto obtiene un mejor rendimiento y reduce la sobrecarga de recursos. En particular, la planificación propuesta completa su trabajo en menor tiempo y la distribución de tareas se ajusta dinámicamente a las características de los nodos grid.

A fin de comprobar las bondades de los sistemas borrosos en la planificación de tareas en sistemas Desktop Grid se han desarrollado una serie de algoritmos de análisis de imágenes con grandes necesidades de cómputo (capítulo 6). Estos algoritmos realizan de forma automática el proceso de verificación y detección de defectos en Tableros de Fibra recubiertos de Papel Melamínico. Usando un sistema Desktop Grid con planificador local borroso, se consigue disminuir el tiempo empleado en controlar la calidad de los productos y con los algoritmos de análisis de imágenes diseñados se aumenta la tasa de defectos detectados.

El estudio presentado en esta tesis permite mejorar el proceso proceso productivo de TFPM ya que la detección de la calidad de los productos se realiza en un tiempo menor al empleado actualmente, lo que permite incrementar la productividad. Además, este sistema no necesita realizar grandes inversiones en recursos de cómputación ni en complejos sistemas de visión por computador, por lo que supone enormes efectos positivos con una inversión mínima.

En resumen esta tesis tiene las siguientes contribuciones:

- Esta tesis discute los conceptos claves y las características de los sistemas Grid y además proporciona diversas taxonomías de estos sistemas.
- Esta tesis presenta las funcionalidades clave de planificación en sistemas Grid.

- Esta tesis propone dos planificadores de tareas, que se adaptan al entorno dinámico de un sistema Grid.
- Esta tesis comprueba la funcionalidad y eficacia de un planificador local borroso sobre un sistema Desktop Grid.

7.2. Publicaciones

Revistas incluidas en JCR

a) A.J. Sánchez Santiago, A.J. Yuste J.E. Muñoz Expósito, S. García Galán, R.P. Prado. A Multi-criteria Meta-Fuzzy-Scheduler for independent tasks in grid computing. *Computing and Informatics*. Vol. 30, No. 6, pp 1201-1224. Diciembre 2011.

b) A.J. Sánchez Santiago, A.J. Yuste, J.E. Muñoz Expósito, S. García Galán, R.P. de Prado, J.M. Maqueira, S. Bruque. Real-time image texture analysis in quality management using grid computing: an application to the MDF manufacturing industry. *The International Journal of Advanced Manufacturing Technology*. Springer London, Vol 58, No. 9-12, pp. 1217-1225. Febrero 2012.

Revistas revisión por pares

a) A.J. Sánchez Santiago, A.J. Yuste, J.E. Muñoz Expósito, S. García Galán, J.M. Maqueira, S. Bruque. A dynamic-balanced scheduler for genetic algorithms for grid computing. *WSEAS TRANSACTIONS on COMPUTERS*. Vol. 8, No. 1, pp. 11-20. Enero 2009.

Congresos internacionales

a) A.J. Sánchez Santiago, A.J. Yuste-Delgado, J.E. Muñoz-Expósito, S. García-Galán, J.M. Maqueira-Marín, S. Bruque. An Expert Fuzzy Grid Scheduler for Virtual Organizations. *Proceedings of the 2008 International Conference on Computational Intelligence for Modelling Control & Automation*. Conference proceedings, Viena (Austria), Diciembre 2008.

b) A.J. Sánchez Santiago, A.J. Yuste, J.E. Muñoz Expósito, S. García Galán, S. Bruque. A balanced scheduler for grid computing. *Proceedings of the 8th conference on Simulation, modelling and optimization*. Conference proceedings, Santander, Cantabria (España), Septiembre 2008.

7.3. Otras publicaciones relacionadas

A continuación se enumeran trabajos que utilizan resultados obtenidos a lo largo del desarrollo de esta tesis doctoral.

Congresos internacionales

a) R. Prado, S. Galán, A. Yuste, J. Expósito, A.J. Sánchez Santiago, S. Bruque. Evolutionary Fuzzy Scheduler for Grid Computing. *Bio-Inspired Systems: Computational and Ambient Intelligence (Lecture Notes in Computer Science)*. Vol 5517, pp. 286-293. Junio 2009.

Congresos nacionales

a) J. E. Muñoz, S. García Galán, A. J. Yuste Delgado, A. J. Sánchez Santiago, J. M. Maqueira Marín, S. Bruque Cámara. Grid para el intercambio de contenidos multimedia. *VII Jornadas de Ingeniería Telemática. JITEL 2008*. Conference procedings, Alcalá de Henares, Madrid, (España), Septiembre 2008.

7.4. Trabajos Futuros

Una línea de futuro que se desprende directamente del trabajo realizado, es llevar el estudio planteado en esta tesis a la práctica, es decir, crear un sistema de visión por computador, basado en cámaras que permita explorar el producto directamente en la cadena de producción. Asociado a esto se usaría el planificador local borroso para un sistema Desktop Grid formado por los recursos existentes en la empresa, para de esta forma, identificar defectos obteniendo los beneficios anteriormente expuestos.

En cuanto a los entornos Desktop Grid, contienen grandes dificultades de gestión, por lo que, han surgido muchas ideas para el trabajo futuro en el proceso de desarrollo de esta tesis. Se realiza un resumen de algunas ideas para el trabajo futuro que no se han investigado en esta tesis pero que podría dar lugar a nuevas mejoras en el ámbito de la planificación en Desktop Grids.

Como se ha comentado a lo largo de esta memoria, en el capítulo 5 se han desarrollado dos planificadores. Estos dos planificadores pertenecen a distintos niveles de la jerarquía Grid, esto es, el planificador local balanceado gestiona recursos y el Meta-Planificador borroso administra nodos grid que contienen recursos. Un trabajo futuro próximo reside en comprobar la eficiencia y funcionalidad de ambos planificadores de manera conjunta.

Para verificar el Meta-Planificador borroso se han utilizado un conjunto de tareas agrupadas en Bolsas de Tareas (BoT). Un tipo de verificación distinto es posible utilizando tareas independientes sin agrupar, esto es despacharlas según llegan al planificador. También es interesante, de cara al futuro, ordenar la bolsa de tareas, siguiendo algún algoritmo de ordenación o incluso aplicando cierta prioridad a determinadas tareas.

El planificador local borroso utilizado en esta tesis está basado en un Desktop Grid empresarial pensado para recursos conectados a través de una LAN, en el que todos los recursos son proporcionados por una empresa, por lo tanto estos recursos son conocidos y fiables. Un nuevo campo de investigación puede ser el uso de computación voluntaria basado en Internet. De esta forma los recursos usados para ejecutar las tareas no pertenecen a la empresa, siendo el control sobre ellos escaso. Estos recursos pueden ser muy volátiles, egoístas, o incluso maliciosos, por lo tanto, la planificación de tareas con este tipo de recursos requiere tener en cuenta nuevas funcionalidades como pueden ser la re-planificación de tareas, la tolerancia a fallos y/o la seguridad. Este planificador local borroso usa conjuntos difusos triangulares, una alternativa futura interesante, consiste en comprobar el funcionamiento de dicho planificador usando otros tipos de conjuntos difusos como pueden ser los gausianos.

Desktop Grid distribuidos han surgido recientemente como una alternativa al Desktop Grid centralizado, ya que puede resolver la sobrecarga y escalabilidad del Desktop Grid centralizado. Una adaptación de este trabajo a un Desktop Grid distribuido necesitaría la construcción de una red computacional y dotar una planificación de tareas distribuida ya que no existe un servidor central que actúe como planificador único.

Bibliografía

[Abbas04] Abbas A. *Grid computing: a practical guide to technology and applications*. Charles River Media, Hinghan, Massachusetts., 2004.

[Ahmad99] Ahmad M.B. and Tae-Sun Choi. Local threshold and boolean function based edge detection. *IEEE Transactions on Consumer Electronics*, 45(3).

[Altmann08] Altmann Jörn, Courcoubetis Costas, Stamoulis Georges D., Dramitinos Manos, Rayna Thierry, Risch Marcel, and Bannink Chris. GridEcon: A Market Place for Computing Resources. In *Grid Economics and Business Models*, volume 5206 of *Lecture Notes in Computer Science*, pages 185–196. Springer Berlin / Heidelberg, 2008.

[Anderson04] Anderson David P. Boinc: A system for public-resource computing and storage. In *5th IEEE/ACM International Workshop on Grid Computing*, pages 4–10, 2004.

[Andreeva08] Andreeva J., Campana S., Fanzago F., and Herrala J. High-energy physics on the grid: the atlas and cms experience. *Journal of Grid Computing*, 6:3–13, 2008.

[Aziz08] Aziz A. and El-Rewini H. On the use of meta-heuristics to increase the efficiency of online grid workflow scheduling algorithms. *Cluster Computing*, 11(4):373–390, 2008.

[Belgacem12] Belgacem Mohamed, Abdennadher Nabil, and Niinimaki Marko. Virtual ez grid: A volunteer computing infrastructure for scientific medical applications. *International Journal of Handheld Computing Research (IJHCR)*, 3:75–85, 2012.

[Berman03] Berman F., Wolski R., Casanova H., Cirne W., Dail H., Faerman M., Figueira S., Hayes J., Obertelli G., Schopf J., Shao G., Smallen S., Spring N., Su A., and Zagorodnov D. Adaptive computing on the grid using apples. *Parallel and Distributed Systems (TPDS)*, 14(4):369–382, 2003.

[Berstis02] Berstis V. IBM Corporation International Technical Support Organization, Austin, Texas, 2002.

[Bharati03] Bharati M. H., MacGregor J. F., and Tropper W. Softwood lumber grading through on-line multivariate image analysis techniques. *Industrial & Engineering Chemistry Research*, 42(21):5345–5353, 2003.

[Bolze06] Bolze Raphaël, Cappello Franck, Caron Eddy, Daydé Michel, Desprez Frédéric, Jeannot Emmanuel, Jégou Yvon, Lanteri Stephane, Antipolis Sophia, Leduc Julien, Melab Noredine, Mornet Guillaume, Namyst Raymond, Primet Pascale, Quetier Benjamin, Richard Olivier, Talbi El-Ghazali, and Touche Iréa. Grid'5000: A large scale and highly reconfigurable experimental grid testbed. *International Journal of High Performance Computing Applications*, 20(4):481–494, 2006.

[Brasileiro11] Brasileiro Francisco, Andrade Nazareno, Lopes Raquel, and Sampaio Lívia. Democratizing resource-intensive e-science through peer-to-peer grid computing. pages 53–80, 2011.

[Braun01] Braun T.D., Siegel H.J., and Beck N. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. *Journal of Parallel and Distributed Computing*, 61:810–837, 2001.

[Buyya02] Buyya R., Abramson D., Giddy J., and Stockinger H. Economic models for resource management and scheduling in grid computing. *J. of Concurrency and Computation: Practice and Experience*, 14(13-15):1507–1542, 2002.

[Buyya05] Buyya R., Abramson D., and Venugopal S. The grid economy. *Proceedings of the IEEE*, 93(3):698–714, 2005.

[Canny86] Canny John. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[Cao12] Cao CaiFeng and Jiang DeDong. Research and improvement of resource scheduling mechanism in alchemi desktop grid. In *Advances in Multimedia, Software Engineering and Computing Vol.1*, volume 128, pages 497–502. 2012.

[Cappello05] Cappello Franck, Djilali Samir, Fedak Gilles, Herault Thomas, Magniette Frédéric, Néri Vincent, and Lodygensky Oleg. Computing on large-scale distributed systems: Xtremweb architecture, programming models, security, tests and convergence with grid. *Future Generation Computer Systems*, 21(3):417–437, 2005.

[Chang06] Chang C.I., Du Y., Wang J., Guo S.M., and Thouin P.D. Survey and comparative analysis of entropy and relative entropy thresholding techniques. *IEEE Proceedings Vision, Image and Signal Processing*, 153(6):837–850, 2006.

[Chang09] Chang Ruay-Shiung, Chang Jih-Sheng, and Lin Po-Sheng. An ant algorithm for balanced job scheduling in grids. *Future Generation Computer Systems*, 25(1):20–27, 2009.

[Chauhan10] Chauhan S.S. and Joshi R.C. A weighted mean time min-min max-min selective scheduling strategy for independent tasks on grid. *IEEE 2nd International Advance Computing Conference (IACC)*, 26(4):608 – 621, 2010.

[Cheeseman90] Cheeseman Peter, Kelly James, Self Matthew, Stutz John, Taylor Will, and Freeman Don. Autoclass: A bayesian classification system. *Proc of the Fifth Intl Workshop on Machine Learning*, 27(11):54–64, 1988.

[Chen05] Chen X., Jing H., Tao Y., and Cheng X. Real-time image analysis for nondestructive detection of metal slivers in packed food. *Optical Sensors and Sensing Systems for Natural Resources and Food Safety and Quality*, 5996:120–129, 2005.

[Cheriet98] Cheriet M., Said J.N., and Suen C.Y. A recursive thresholding technique for image segmentation. *IEEE Transactions on Image Processing*, 7(6):918–921, 1998.

[Chien03] Chien Andrew, Calder Brad, Elbert Stephen, and Bhatia Karan. Entropia: Architecture and performance of an enterprise desktop grid system. *Journal of Parallel and Distributed Computing*, 63:597–610, 2003.

[Choi05] Choi SungJin, Baik MaengSoon, Gil MoonMin, Park ChanYeol, Jung Soon-Young, and Hwang ChongSun. Dynamic scheduling mechanism for result certification in peer to peer grid computing. In *Grid and Cooperative Computing - GCC 2005*, volume 3795 of *Lecture Notes in Computer Science*, pages 811–824. Springer Berlin / Heidelberg, 2005.

[Christodoulopoulos09] Christodoulopoulos K., Sourlas V., Mpakolas I., and Varvarigos E. A comparison of centralized and distributed meta-scheduling architectures for computation and communication tasks in grid networks. *Comput. Commun.*, 32:1172–1184, 2009.

[Chu04] Chu D.C. and Humphrey M. Mobile ogsi.net: Grid computing on mobile devices. *IEEE/ACM International Workshop on Grid Computing*, pages 182–191, 2004.

[Cooper04] Cooper K., Dasgupta A., Kennedyand K, Koelbel C., Mandal A., Marin G., Mazina M., Mellor-Crummey J., Berman F., Casanova H., Chien A., Dail H., Liu X., Olugbile A., Sievert O., Xia H., Johnsson L., Liu B., Patel M., Reed D., Deng W., Mendes C., Shi Z., Yarkhan A., and Dongarra J. New grid scheduling and rescheduling methods in the grads project. In *18th International Parallel and Distributed Processing Symposium (IPDPS 04)*, pages 199–206, 2004.

[Cordon01] Cordón O., Herrera F., Hoffmann F., and Magdalena L. Genetic fuzzy systems. evolutionary tuning and learning of fuzzy knowledge bases. *Advances in fuzzy systems. Applications and theory*, 19, 2001.

[Dabrowski09] Dabrowski Christopher. Reliability in grid computing systems. 21(8):927–959, 2009.

[Deelman05] Deelman E., Singhand M.-H., Suand J., Blytheand Y., Giland C., Kesselmanand G., Mehtaand K., Vahiand G.B., Berrimanand J., Goodand A., Laityand J.C. Jacob, and Katz D. Pegasus: a framework for mapping complex scientific workflows onto distributed systems. *J. Sci. Programm.*, 13(2):219–237, 2005.

[Deng07] Deng Jeremiah D. and Gleeson Matthew T. Automatic sapstain detection in processed timber. *Lecture Notes in Computer Science*, 4830/2007:637–641, 2007.

[Domingues07] Domingues Patricio, Sousa Bruno, and Moura Silva Luis. Sabotage-tolerance and trust management in desktop grid computing. *Future Gener. Comput. Syst.*, 23:904–912, 2007.

[Efford00] Efford Nick. *Digital Image Processing: A Practical Introduction Using Java*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000.

[Ellisman04] Ellisman M. and Peltier S. *Medical Data Federation: The Biomedical Informatics Research Network*. Morgan Kaufmann Publishers Inc, San Francisco, CA, USA., 2004.

[Ernemann04] Ernemann Carsten, Hamscher Volker, and Yahyapour Ramin. Benefits of global grid computing for job scheduling. In *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, GRID '04, pages 374–379, 2004.

[Escalera01] Escalera Hueso Arturo de la. *Visión por computador: fundamentos y métodos*. Prentice Hall, 2001.

[Falzon10] Falzon Geoffrey and Li Maozhen. Enhancing list scheduling heuristics for dependent job scheduling in grid computing environments. *The Journal of Supercomputing*, pages 1–27, 2010.

[Fayad07] Fayad C., Garibaldi J.M., and Ouelhadj D. Fuzzy grid scheduling using tabu search. In *IEEE International Conference on Fuzzy Systems*, pages 1054–1059, 2007.

[Foster01] Foster Ian, Kesselman Carl, and Tuecke Steven. The anatomy of the grid: Enabling scalable virtual organizations. *Int. J. High Perform. Comput. Appl.*, 15:200–222, 2001.

[Foster04a] Foster I. and Kesselman C. *Concepts and Architecture*. Morgan Kaufmann Publishers Inc, San Francisco, CA, USA., 2004.

[Foster04b] Foster I., Kesselman C., and Tuecke S. The open grid services architecture". pages 215–258, 2004.

[Franke07] Franke C., Lepping J., and Schwiegelshohn U. Genetic fuzzy systems applied to online job scheduling. In *IEEE International Fuzzy Systems Conference*, FUZZ-IEEE 2007., pages 1–6, 2007.

[Franke08] Franke C., Hoffmann F., Lepping J., and Schwiegelshohn U. Development of scheduling strategies with genetic fuzzy systems. *Appl. Soft Comput.*, 8(1):706–721, 2008.

[Garey90] Garey Michael R. and Johnson David S. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[Gelado10] Gelado Isaac, Stone John E., Cabezas Javier, Patel Sanjay, Navarro Nacho, and Hwu Wen mei W. An asymmetric distributed shared memory model for heterogeneous parallel systems. *SIGARCH Comput. Archit. News*, 38:347–358, 2010.

[Gentzch04] Gentzch W. *Grid Computing Adoption in Research and Industry*. Charles River Media Inc, Hingham, Massachusetts., 2004.

[Gentzsch06] Gentzsch Wolfgang. D-grid, an e-science framework for german scientists. *International Symposium on Parallel and Distributed Computing*, pages 12–13, 2006.

[Gonzalo01] Gonzalo Pajares Jesús M. de la Cruz. *Visión por Computador: Imágenes digitales y aplicaciones*. RaMa, 2001.

[Gose96] Gose Earl, Johnsonbaugh Richard, and Jost Steve. *Pattern recognition and image analysis*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.

[Gounaris06] Gounaris A., Sakellariou R., Paton N., and Fernandes A. A novel approach to resource scheduling for parallel query processing on computational grids. *Distrib. Parallel Databases*, 19(20):87–106, 2006.

[Hanada05] Hanada Yoshiko, Hiroyasu Tomoyuki, Miki Mitsunori, and Okamoto Yuko. Mega process genetic algorithm using grid mp. In *Grid Computing in Life Science*, Lecture Notes in Computer Science, pages 152–170. Springer Berlin / Heidelberg, 2005.

[He03] He XiaoShan, Sun Xianhe, and Laszewski Gregor von. Qos guided min-min heuristic for grid task scheduling. *Journal of Computer Science and Technology*, 18:442–451, 2003.

[Herrera97] Herrera F. and Magdalena L. Genetic fuzzy systems. *Tatra Mountains Mathematical Publications*, 13:93–121, 1997.

[Hey02] Hey Tony and Trefethen Anne E. The uk e-science core programme and the grid. *Future Generation Computer Systems*, 18(8):1017 – 1031, 2002.

[Hongbo10] Hongbo Liu, Ajith Abraham, and Aboul Ella Hassanien. Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm. *Future Generation Computer Systems*, 26(8):1336 – 1343, 2010.

[Huang07] Huang Chi-Yu., Holt Alan, Monk John, and Cheng Kai. The application of dependency management in an integrated manufacturing network framework. *Int J Adv Manuf Technol*, 33(3):354–364, 2007.

[Huang08] Huang Ye, Brocco Amos, Kuonen Pierre, Courant Michele, and Hirsbrunner Bat. Smartgrid: A fully decentralized grid scheduling framework supported by swarm intelligence. *International Conference on Grid and Cloud Computing*, pages 160–168, 2008.

[Huang09] Huang Peijie, Peng Hong, Lin Piyuan, and Li Xuezhenios. Static strategy and dynamic adjustment: An effective method for grid task scheduling. *Future Generation Computer Systems*, 25(8):884 – 892, 2009.

[Hwang07] Hwang Junseok and Park Jihyoun. Decision factors of enterprises for adopting grid computing. In *Grid Economics and Business Models*, volume 4685 of *Lecture Notes in Computer Science*, pages 16–28. Springer Berlin / Heidelberg, 2007.

[Hwang08] Hwang Reakook, Gen Mitsuo, and Katayama Hiroshi. A comparison of multiprocessor task scheduling algorithms with communication costs. *Comput. Oper. Res.*, 35:976–993, 2008.

[Iosup11] Iosup A. and Epema D.H.J. Grid computing workloads: Bags of tasks, workflows, pilots, and others. *IEEE Internet Computing*, 15:19–26, 2011.

[Kacsuk07] Kacsuk Péter, Podhorszki Norbert, and Kiss Tamás. Scalable desktop grid system. In *High Performance Computing for Computational Science - VECPAR 2006*, volume 4395 of *Lecture Notes in Computer Science*, pages 27–38. Springer Berlin / Heidelberg, 2007.

[Kane10] Kane K. and Dillaway B. Cyclotron: a secure, isolated, virtual cycle-scavenging grid in the enterprise. *Concurrency and Computation: Practice and Experience*, 22(3):241–260, 2010.

[Ke09] Ke Jia, Zhan Yongzhao, Chen Xiaojun, and Wang Manrong. Pseudo invariant line moment to detect the target region of moving vessels. *Lecture Notes in Computer Science*, 5754/2009:615–624, 2009.

[Keith08] Keith A., Brewster, Daniel B. Weber, Kevin W. Thomas, Kelvin K. Droegemeier, Yunheng Wang, Ming Xue, Suresh Marru, Dennis Gannon, Jay

Alameda, Brian F. Jewett, Jack S. Kain, Steven J. Weiss, and Marcus Christie. Use of the lead portal for on-demand severe weather prediction. *24th Conference on IIPS*, 2008.

[Kiepuszewski03] Kiepuszewski B., Hofstede A.H.M., and van der Aalst W.M.P. Fundamentals of control flow in workflows. *Acta Inform*, 39(3):143–209, 2003.

[Killing09] Killing J., Surgenorv B.W., and Mechefske C.K. A machine vision system for the detection of missing fasteners on steel stampings. *Int J Adv Manuf Technol*, 41(7-8):808–819, 2009.

[Kittler83] Kittler J. On the accuracy of the sobel edge detector. 1(1):37–42, 1983.

[Klusácek08] Klusácek Dalibor, Matyska Ludek, and Rudová Hana. Alea - grid scheduling simulation environment. *Parallel Processing and Applied Mathematics*, 4967:1029–1038, 2008.

[Kopetz11] Hermann Kopetz. *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Real-Time Systems Series. Springer, 2011.

[Kousalya08] Kousalya K. and Balasubramanie P. Ant algorithm for grid scheduling powered by local search. *Int. J. Open Problems Compt. Math.*, 1(2):30–44, 2008.

[Krishnapuram93] Krishnapuram R., , and Keller J.M. A possibilistic approach to clustering. *IEEE Transactions on Fuzzy Systems*, 1(2):98–110, 1993.

[Kumar08] Kumar A. Computer-vision-based fabric defect detection: A survey. 55(1):348–363, 2008.

[Kumar09] Kumar Subodha, Dutta Kaushik, and Mookerjee Vijay. Maximizing business value by optimal assignment of jobs to resources in grid computing. *European Journal of Operational Research*, 194(3):856 – 872, 2009.

[Lau06] Lau Kam-Keung, Roberts Stephen, Biro Dora, Freeman Robin, Meade Jessica, and Guilford Tim. An edge-detection approach to investigating pigeon navigation. *Journal of Theoretical Biology*, 239(1):71–78, 2006.

[Leban04] Leban J.M., Pizzi A., Wieland S., Zanetti M., Properzi, and Pichelin F. X-ray microdensitometry analysis of vibration-welded wood. *Journal of Adhesion Science and Technology*, 18(6):673–685, 2004.

[Lee06] Lee L.T., Liang C.H., and Chang H.Y. An adaptive task scheduling system for grid computing. In *The Sixth IEEE International Conference on Computer and Information Technology (CIT 06)*, pages 57–62, 2006.

[Lee07] Lee Y.C. and A.Y. Zomaya. Practical scheduling of bag-of-tasks applications on grids with dynamic resilience. *IEEE Transactions on Computers*, 56(6):815–825, 2007.

[Lee83] Lee C.C. Elimination of redundant operations for a fast sobel operator. 13:242–245, 1983.

[Li09] Li Y., Yanga Y., Mab M., and Zhoua L. A hybrid load balancing strategy of sequential tasks for grid computing environments. *Computers and Operations Research*, 35:819–828, 2009.

[Liao01] Liao Ping-sung, Chen Tse-sheng, and Chung Pau-choo. A fast algorithm for multilevel thresholding. *Journal of Information Science and Engineering*, 17:713–727, 2001.

[Lim07] Lim Dudy, Yew-Soon Ong, Yaochu Jin, Bernhard Sendhoff, and Bu-Sung Lee. Efficient hierarchical parallel genetic algorithms using grid computing. *Future Generation Computer Systems*, 23(4):658–670, 2007.

[Liu07] Liu Y., Wei J., Yu k., Yin Zhou W., and Xia J. A unified strategy for dynamic mapping in grid computing environments. In *International Conference on Mechatronics and Automation, ICMA 2007*, pages 2309–2313, 2007.

[Lu06] Lu R.S., Tian G.-Y., Gledhill D., and Ward S. Grinding surface roughness measurement based on the co-occurrence matrix of speckle pattern texture. *Applied Optics*, 45(35):8839–8847, 2006.

[Lumb04] Lumb Ian and Smith Chris. Grid resource management. chapter Scheduling attributes and platform LSF, pages 171–182. 2004.

[Luthon04] Luthon Franck, Liévin Marc, and Faux Francis. On the use of entropy power for threshold selection. *Signal Processing*, 84(10):1789 – 1804, 2004.

[Ma03] Ma Li, Tan Tieniu, Wang Yunhong, and Zhang Dexin. Personal identification based on iris texture analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(12):1519–1533, 2003.

[Magdalena96] Magdalena Luis and Velasco Juan R. Velasco. *Fuzzy Rule-Based Controllers that Learn by Evolving their Knowledge Base. In Herrera and J.L. Verdegay, editors, Genetics Algorithms and Soft Computing.* Physica-Verlag, 1996.

[Magdalena97] Magdalena Luis and Velasco Juan R. Velasco. *Evolutionary based learning of fuzzy controllers. In W. Pedrycz, editor, Fuzzy evolutionary computation.* Kluwer Academic Press, 1997.

[Mamdani74] Mamdani E. H. Application of fuzzy algorithms for control a simple dynamic plant. *Proceedings of the Institution of Electrical Engineers*, Part D 121:1585–1588, 1974.

[Mamdani75] Mamdani E. H. and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7:1–13, 1975.

[Maqueira08] Maqueira Marín J.M. and Bruque Cámara S. *Las Tecnologías Grid de la Información como Nueva Herramienta Empresarial*. Septem Ediciones, Jaén (España), 2008.

[Marr91] Marr D. and Hildreth E. Theory of edge detection. *Computer Vision*, pages 77–107, 1991.

[Mehta10] Mehta Hemant Kumar, Kanungo Priyesh, and Chandwani Manohar. Maximum utility meta-scheduling algorithm for economy based scheduling under grid computing. In *Contemporary Computing*, volume 95 of *Communications in Computer and Information Science*, pages 23–33. Springer Berlin Heidelberg, 2010.

[Mietzner09] Mietzner Ralph, Karastoyanova Dimka, and Leymann Frank. Business grid: Combining web services and the grid. In *Transactions on Petri Nets and Other Models of Concurrency II*, volume 5460 of *Lecture Notes in Computer Science*, pages 136–151. Springer Berlin / Heidelberg, 2009.

[Min-You00] Min-You W., Shu W., and Zhang H. Segmented min-min: a static mapping algorithm for meta-tasks on heterogeneous computing systems. pages 375–385, 2000.

[Murino04] Murino V., Bicego M., and Rossi I.A. Statistical classification of raw textile defects. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 4, pages 311–314, 2004.

[Neary05] Neary Michael O. and Cappello Peter. Advanced eager scheduling for java-based adaptive parallel computing: Research articles. *Concurr. Comput. : Pract. Exper.*, 17:797–819, 2005.

[Nebro07] Nebro Antonio, Alba Enrique, and Luna Francisco. Multi-objective optimization using grid computing. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 11:531–540, 2007.

[Netto11] Netto Marco A.S. and Rajkumar Buyya. Coordinated rescheduling of bag-of-tasks for executions on multiple resource providers. *Concurrency and Computation: Practice and Experience*, 2011.

[Neubauer97] Neubauer C. Intelligent x-ray inspection for quality control of solder joints. *IEEE Transactions on Components, Packaging, and Manufacturing Technology, Part C*, 20(2):111–120, 1997.

[Nimbarte10] Nimbarte N.M. and Mushrif M.M. Multi-level thresholding algorithm for color image segmentation. In *Second International Conference on Computer Engineering and Applications (ICCEA)*, pages 231–233, 2010.

[Ortalana09] Ortalana V., Herrera M., Morgan D.G., and Browning N.D. Application of image processing to stem tomography of low-contrast materials. *Ultramicroscopy*, 110(1):67–81, 2009.

[Otsu79] Otsu N. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1):62–66, 1979.

[Paletta09] Paletta M. and Herrero P. A simulated annealing method to cover dynamic load balancing in grid environment. *Advances in Soft Computing*, 50:1–10, 2009.

[Peng08] Peng Xiangqian, Chen Youping, Yu Wenyong, Zhou Zude, and Sun Guodong. An online defects inspection method for float glass fabrication based on machine vision. *The International Journal of Advanced Manufacturing Technology*, 39(11):1180–1189, 2008.

[Plaszczak05] Plaszczak Pawel and Wellner Richard. *Grid Computing: The Savvy Manager's Guide*. Morgan Kaufmann Publishers Inc., 2005.

[Prasad09] Prasad B.S and Sarcar M.M.M. Experimental investigation to predict the condition of cutting tool by surface texture analysis of images of machined surfaces based on amplitude parameters. *I J of Machining and Machinability of Materials*, 4(2-3):217–236, 2009.

[Priya07] Priya S.Baghavathi, Prakash M., and Dhawan K.K. Fault tolerance-genetic algorithm for grid task scheduling using check point. *Grid and Cloud Computing, International Conference on*, pages 676–680, 2007.

[Pun80] Pun Thierry. A new method for grey-level picture thresholding using the entropy of the histogram. *Signal Processing*, 2(3):223 – 237, 1980.

[Pun81] Pun Thierry. Entropic thresholding, a new approach. *Computer Graphics and Image Processing*, 16(3):210 – 239, 1981.

[Qu09] Qu Zhenhua, Qiu Guoping, and Huang Jiwu. Detect digital image splicing with visual cues. *Lecture Notes in Computer Science*, 5806/2009:247–261, 2009.

[Rahman11] Rahman M., Ranjan R., Buyya R., and Benatallah B. A taxonomy and survey on autonomic management of applications in grid computing environments. *Concurrency and Computation: Practice and Experience*, 23(16):1990–2019, 2011.

[Ranaldo09] Ranaldo N. and Zimeo E. Time and cost-driven scheduling of data parallel tasks in grid workflows. *IEEE Systems Journal*, 3(1):104–120, 2009.

[Sahu11] Sahu Rajendra and Chaturvedi Anand K. Many-objective comparison of twelve grid scheduling heuristics. *International Journal of Computer Applications*, 13(6):9–17, 2011.

[Sakellariou07] Sakellariou Rizos, Zhao Henan, Tsiakkouri Eleni, and Dikaiakos Marios. Scheduling workflows with budget constraints. In *Integrated Research in GRID Computing*, pages 189–202. Springer US, 2007.

[Shah04] Shah N., Chao K.M., Godwin N., Younas M., and Laing C. Exception diagnosis in agent-based grid computing. *IEEE International Conference on Systems, Man and Cybernetics*, 4:3213–3219, 2004.

[Shams08] Shams Ramtin, Hartley Richard, and Navab Nassir. Real-time simulation of medical ultrasound from ct images. 5242:734–741, 2008.

[Silvén03] Silvén Olli, Niskanen Matti, and Kauppinen Hannu. Wood inspection with non-supervised clustering. *Mach. Vision Appl.*, 13(5-6):275–285, 2003.

[Simonaho04] Simonaho Simo-Pekka, Palviainen Jari, Tolonen Yrj, and Silvennoinen Raimo. Determination of wood grain direction from laser light scattering pattern. *Optics and Lasers in Engineering*, 41(1):95–103, 2004.

[Smith00] Smith Melvyn L. and Stamp Richard J. Automated inspection of textured ceramic tiles. *Computers in Industry*, 43(1):73–82, 2000.

[Smith97] Smith Stephen M. and Brady J. Michael. Susan a new approach to low level image processing. *International Journal of Computer Vision*, 23:45–78, 1997.

[Sonka99] Sonka M., Hlavac V., and Boyle R. *Image Processing Analysis and Machine Vision*. Chapman & Hall, 1999.

[Sonmez09] Sonmez O., Grundeken B., Mohamed H., Iosup A., and Epema D. Scheduling strategies for cycle scavenging in multicluster grid systems. In *Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 12–19. IEEE Computer Society, 2009.

[Subhlok00] Subhlok J. and Vondran G. Optimal use of mixed task and data parallelism for pipelined computations. *J. Parallel Distrib. Comput.*, 60(23):297–319, 2000.

[Subramanian05] Subramanian R. and goodman B.D. IDEA group Publising, 2005.

[Subrata10] Subrata Riky, Zomaya Albert Y., and Landfeldt Bjorn. Cooperative power-aware scheduling in grid computing environments. *Journal of Parallel and Distributed Computing*, 70(2):84 – 91, 2010.

[Sulistio08] Sulistio Anthony, Cibej Uros, Venugopal Srikumar, Robic Borut, and Buyya Rajkumar. A toolkit for modelling and simulating data grids: an extension to gridsim. *Concurr. Comput. : Pract. Exper.*, 20:1591–1609, 2008.

[Sun07] Sun Ying, Zhao Shuqi, Yu Huashan, Gao Ge, and Luo Jingchu. Abcgrid: Application for bioinformatics computing grid. 23(9):1175–1177, 2007.

[Takagi85] Takagi T. and Sugeno M. Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, 15(1):116–132, 1985.

[Tang06] Tang Ming, Lee Bu-Sung, Tang Xueyan, and Yeo Chai-Kiat. The impact of data replication on job scheduling performance in the data grid. *Future Generation Computer Systems*, 22(3):254 – 268, 2006.

[Taufer05] Taufer M., Anderson D., Cicotti P., and Brooks Iii C. L. Homogeneous redundancy: a technique to ensure integrity of molecular simulation results using public computing. In *Proceedings of the 14th Heterogeneous Computing Workshop HCW (2005), in conjunction with IPDPS*, page 119, 2005.

[Taylor07] Taylor Nolan J. Public grid computing participation: An exploratory study of determinants. *Information amp; Management*, 44(1):12 – 21, 2007.

[Thain05] Thain Douglas, Tannenbaum Todd, and Livny Miron. Distributed computing in practice: the condor experience: Research articles. *Concurr. Comput. : Pract. Exper.*, 17:323–356, 2005.

[TienPing04] Tien Ping Tan, Chand Sodhy Gian, Huah Yong Chan, Haron Fazilah, and Buyya Rajkumar. A market-based scheduler for jxta-based peer-to-peer computing system. In *ICCSA (4)'04*, pages 147–157, 2004.

[Tinghuai10] Tinghuai Ma, Jian Ge, Hao Cao, and Yali Wang. Design and implementation of virtual resources management in meteorology grid. *International Conference on Grid and Cloud Computing*, pages 58–63, 2010.

[Tsai95] Tsai Wen-Hsiang. Document image analysis. chapter Moment-preserving thresholding: a new approach, pages 44–60. IEEE Computer Society Press, Los Alamitos, CA, USA, 1995.

[Tseng08] Tseng C.-T. and Liao C.-J. A discrete particle swarm optimization for lot-streaming flowshop scheduling problem. *European Journal of Operational Research*, 191(2):360–373, 2008.

[Tseng09] Tseng Li-Ya, Chin Yeh-Hao, and Wang Shu-Ching. A minimized makespan scheduler with multiple factors for grid computing systems. *Expert Systems with Applications*, 36(8):11118 – 11130, 2009.

[Velasco95] Velasco Juan R and Magdalena Luis. *Genetic Algorithms in Fuzzy Control Systems. In J. Periaux and G. Winter, editors, Genetic Algorithms in Engineering and Computer Science.* John Wiley and Sons Ltd, 1995.

[Venugopal05] Venugopal S. and Buyya R. A deadline and budget constrained scheduling algorithm for escience applications on data grids. In *6th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP-2005)*, pages 60–72, 2005.

[Vlahovic12] Vlahovic Milica M., Savic Maja M., Martinovic Sanja P., Boljanac T., and Volkov-Husovic Tatjana D. Use of image analysis for durability testing of sulfur concrete and portland cement concrete. *Materials amp; Design*, 34(0):346 – 354, 2012.

[Wang84] Wang Shyuan and Haralick Robert M. Automatic multithreshold selection. *Computer Vision, Graphics, and Image Processing.*

[Weng05] Weng C. and Lu X. Heuristic scheduling for bag-of-tasks applications in combination with qos in the computational grid. volume 21, pages 271–280, 2005.

[Wieczorek08] Wieczorek Marek, Podlipnig Stefan, Prodan Radu, and Fahringer Thomas. Bi-criteria scheduling of scientific workflows for the grid. *Cluster Computing and the Grid, IEEE International Symposium on*, pages 9–16, 2008.

[Wieczoreka09] Wieczoreka M., Hoheisel A., and Prodan R. Towards a general model of the multi-criteria workflow scheduling on the grid. *Future Generation Computer Systems*, 25(3):237–256, 2009.

[Woltman07] Woltman G. The great internet mersenne prime search (gimps). *http://www.mersenne.org*, 2007.

[Xhafa09] Xhafa F., Carretero J., Dorronsoro B., and Alba E. A tabu search algorithm for scheduling independent jobs in computational grids. *Computing and Informatics*, 28(2):237–250, 2009.

[Xhafa10a] Xhafa Fatos, Paniagua Claudi, Barolli Leonard, and Caballe Santi. A parallel grid-based implementation for real-time processing of event log data of collaborative applications. *Int. J. Web Grid Serv.*, 6:124–140, 2010.

[Xhafa10b] Xhafa Fatos and Abraham Ajith. Computational models and heuristic methods for grid scheduling problems. *Future Generation Computer Systems*, 26(4):608 – 621, 2010.

[Yin09] Yin Y. and Lei J. Prototype system of textile flaw detection based on wavelet reconstructions. *J of Information and Computational Science*, 5(1):207–214, 2009.

[Yu06a] Yu J. and Buyya R. Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms. In *Scientific Programming, IOS Press*, pages 14–217, 2006.

[Yu06b] Yu J. and Buyya R. A budget constrained scheduling of workflow applications on utility grids using genetic algorithms. In *15th IEEE International Symposium on High Performance Distributed Computing, HPDC*, Paris (France), 2006.

[Yu08] Yu J., Buyya R., and Ramamohanarao K. Workflow scheduling algorithms for grid computing. In *Metaheuristics for Scheduling in Distributed Computing Environments*, volume 146, pages 173–214. 2008.

[YuMing08] Yu Kun-Ming and Chen Cheng-Kwan. An adaptive scheduling algorithm for scheduling tasks in computational grid. *International Conference on Grid and Cloud Computing*, pages 185–189, 2008.

[Zadeh65] Zadeh L.A. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[Zadeh73] Zadeh L.A. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man, and Cybernetics*, 3:28–44, 1973.

[Zadeh75] Zadeh L.A. The concept of a linguistic variable ans its applications to aproximate reasoning. parts i, ii and iii. *Information Sciences*, pages 8–9, 199–249, 301–357, 43–80, 1975.

[Zhai09] Zhai Ming, Jing Zhongliang, Fu Shan, and Luo Xinbin. Defect detection in aluminum foil by input-estimate-based chi-square detector. *Optical Engineering*, 48(11):119–129, 2009.

[Zhuge05] Zhuge Hai. China's e-science knowledge grid environment. *IEEE Intelligent Systems*, 19(1):13–17, 2005.