PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

DISEÑO DE UN ALGORITMO METAHEURÍSTICO GRASP PARA LA MEJORÍA DE UN ALGORITMO MININCREASE APLICADO A LA ASIGNACIÓN EFICIENTE DE INCIDENTES EN UNA MESA DE AYUDA

Tesis para optar por el Título de Ingeniero Informático, que presenta el bachiller:

Julio César Rodríguez Ramos

ASESOR: Ing. Rony Cueva Moscoso

Lima, Diciembre del 2014

RESUMEN

La mesa de ayuda es un área importante en la resolución de incidentes de tecnologías de información en las empresas, tanto dentro (para la misma empresa y sus empleados) como fuera (para los clientes que la empresa ofrece sus servicios y productos).

Sin embargo, la planificación de la resolución de incidentes se hace difícil debido a la imprevisibilidad y espontaneidad de éstos. Dichos incidentes afectan de manera diversa a la continuidad de negocio con consecuencias y tiempo de resolución de diversa magnitud. Asimismo, los técnicos en la mesa de ayuda tienen un tiempo de resolución diverso, con experiencia laboral distinta y son un número finito de personas. Dicho problema se le conoce en problemas de asignación de tareas como “asignación estocástica en línea”.

El algoritmo MinIncrease permite la resolución de problemas de asignación estocásticos en línea. Sin embargo, el problema reside en que los técnicos son personas de diversa experiencia que pueden estar divididos en técnicos con mucha o poca experiencia en el ambiente de una mesa de ayuda. No es preciso que al mejor técnico se le asignen incidentes triviales ni que algún técnico no trabaje hasta que aparezca un incidente de su dificultad apropiada. Es por ello que el algoritmo MinIncrease sólo no basta.

El siguiente proyecto presenta el diseño de un algoritmo metaheurístico GRASP para la mejoría de un algoritmo MinIncrease. La combinación de estos algoritmos permitirá que los incidentes, a pesar de que su aparición sea imprevista, puedan asignarse a los técnicos de la mesa de ayuda de manera eficiente.

FACULTAD DE
CIENCIAS E
INGENIERÍA
ESPECIALIDAD DE
INGENIERÍA INFORMÁTICA

PONTIFICIA
UNIVERSIDAD
CATÓLICA
DEL PERÚ

TEMA DE TESIS PARA OPTAR EL TÍTULO DE INGENIERO INFORMÁTICO

TÍTULO: **DISEÑO DE UN ALGORITMO METAHEURÍSTICO GRASP PARA LA MEJORÍA DE UN ALGORITMO MININCREASE APLICADO A LA ASIGNACIÓN EFICIENTE DE INCIDENTES EN UNA MESA DE AYUDA**

ÁREA: Ciencias de la Computación

PROPONENTE: Rony Cueva Moscoso

ASESOR: Rony Cueva Moscoso

ALUMNO: Julio César Rodríguez Ramos

CÓDIGO: 20090453

TEMA N°: 548

FECHA: 12 de Setiembre de 2014

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
Facultad de Ciencias e Ingeniería
01 DIC. 2014
MIGUEL MEJIA PUENTE
DECANO

DESCRIPCIÓN

La mesa de ayuda es un área importante en las empresas, en donde técnicos con diversas capacidades pueden otorgar soluciones a los problemas e incidentes de tecnologías de información que ocurran a clientes y/o usuarios. Es por este motivo que es necesario que se permita realizar una asignación eficiente de los incidentes a sus técnicos para otorgar una respuesta en el menor tiempo posible.

La peculiaridad del entorno hace imposible una asignación exacta debido a la naturaleza imprevista de los incidentes (su dificultad y cantidad), el número de técnicos y sus diferentes capacidades y conocimientos, y el tiempo no exacto de respuesta a un problema.

El algoritmo MinIncrease es un algoritmo de asignación de tareas que puede responder a un problema estocástico (sin tiempos de resolución exactos de las tareas) y en línea (sin previo conocimiento del momento de la aparición de las tareas), como lo es éste problema, a través de una fórmula que toma en cuenta la velocidad de resolución y el trabajo por terminar de cada uno de los técnicos. Sin embargo, esto no es suficiente debido a que los técnicos no son idénticos, y por lo tanto escoger el primero mejor incurre en fallas de eficiencia.

Para ello, se utilizará el algoritmo metaheurístico GRASP para la mejora de la asignación. El algoritmo GRASP permite "relajar" el criterio de selección con una constante de relajación obteniendo un conjunto de los mejores candidatos aptos, de los cuales se seleccionará uno al azar. Este proceso se realizará repetidas veces en su proceso de mejora para quedarse con el mejor candidato posible. En este caso, GRASP relajará el criterio de "primero mejor" de MinIncrease, obteniendo una selección de posibles técnicos a resolver un incidente.

Av. Universitaria 1801
San Miguel, Lima – Perú

Apartado Postal 1761
Lima 100 – Perú

Teléfono:
(511) 626 2000 Anexo 4801

OBJETIVO GENERAL

Diseñar un algoritmo metaheurístico GRASP para la mejora del algoritmo MinIncrease aplicado a la asignación eficiente de incidentes a técnicos en una mesa de ayuda.

OBJETIVOS ESPECÍFICOS

1. Definir el algoritmo a usar que permita una asignación de tareas a máquinas en paralelo en un entorno estocástico y en línea.
2. Definir la función de mérito para determinar la asignación de un técnico a un incidente entre una lista de candidatos.
3. Definir el grado de relajación (constante α) a utilizar en el algoritmo GRASP para la selección del técnico en la lista restringida.
4. Implementar las estructuras de datos para el soporte del algoritmo, incidentes y técnicos.
5. Desarrollar una experimentación numérica para demostrar la superioridad del apoyo del algoritmo GRASP contra MinIncrease sólo.
6. Construir el programa simulador de la solución.

ALCANCE

El algoritmo pretende otorgar una solución a la asignación de los incidentes, mas no apoya a la solución de los incidentes en sí.

Para este proyecto no se considerarán objetivos de fechas de entrega, ni pre-procesamiento de incidentes ("adelantar el trabajo"), ni reasignación de incidentes de un técnico a otro.

La mejora se evidenciará en un número grande de técnicos en la mesa de ayuda y un flujo de incidentes constante.

El programa simulador arrojará un documento de registro de asignaciones y cumplimientos por momentos, así como indicadores de eficiencia por técnico e indicadores de eficiencia total.

Máximo: 100 páginas

Hey… si no puedes hacerlo… no lo hagas.
Pero si puedes hacerlo… hazlo.
El único que puede ganar contra ti eres tú mismo.
Así que pelea.

…

No importa como resulte, ¡continúa y corre!
¡Despega del pasado y enfrenta el presente!
Haz algo, haz memorias.
Vence el hoy y apunta hacia el futuro.

-IDOLM@STER, "Dream".

Dedicado a todos aquellos que ahora mencionaré…

Agradecimientos:

A mis padres, Julio César y Maria Esther. Por todo el amor y cariño que me han dado, por todos los sacrificios que han hecho para que llegue a este punto, por todas las lecciones de vida que me han dado. Es por ustedes que esto ha sido posible. Les estoy infinitamente agradecido. Ésta es una victoria para ustedes también. ¡Papá, Mamá, lo hemos logrado! Y seguro que será una de muchas más victorias de nosotros que vendrán en el futuro. No ha habido mejor fortuna que tenerlos a ustedes.

A mis hermanos, Christian y Oscar. Por todo el apoyo y buenos momentos que hemos pasado, por las lecciones de juventud, por todos los consejos que me dieron. Gracias a ustedes es que ahora tengo mi propio camino. Los he visto crecer y ellos me han visto a mí. Ahora los dos toman sus propias decisiones, como yo haré ahora que soy un profesional. ¡Gracias totales!

A mis tíos, primos y sobrinos. Todo el cariño y buenos ratos de la familia no se olvida nunca. Gracias por estar durante mis 23 años de vida. Y que sean muchos más.

A mis amigos: la 37, los Chichos, los del Bunka, los sobrevivientes de Informática, Rath's Edge, Starwing, Scarlet Library, los del Magic, los Bronies, los del ASD, los de Touhou en Perú, y muchos otros más. Gracias a todos ustedes es que tuve geniales momentos de ocio, diversas lecciones y puntos de vista en la vida, y me permitieron crecer como persona. Todos ustedes han sido parte importante en mi vida. ¡Espero nos veamos pronto!

A mis profesores y asesores: Miss Nérida, Carlos Granthon, Tania Peso, Reynaldo de Amore, Rony Cueva, Manuel Tupia. Gracias a ellos es que pude conocer y escoger mi propio camino a seguir. Muchas gracias por todo lo aprendido. Espero algún día transmitir su experiencia a todo el que tenga a mi cuidado.

A mis colegas de Interbank, Wolfram y Equifax. Ustedes me han permitido desarrollarme profesionalmente, y me permitieron seguir con mis estudios a la vez que aprendía en el mundo laboral. Gracias por la oportunidad.

A muchos personajes ficticios, en los libros, series, animes y videojuegos. Todos los buenos momentos de ocio y alguna que otra moraleja.

Es por todos, todos ustedes que me permiten decir éstas palabras:

"La victoria hoy es mía".

Que ustedes también sean victoriosos en lo que se propongan.

Y usted también, lector.

-Julio Rodríguez

Tabla de contenido

CAPÍTULO 1

1 Problemática

Según ITIL, un área importante de las empresas que venden o utilizan productos o servicios de Tecnologías de Información (en adelante TI), es el área de mesa de ayuda o "help desk", debido a que tiene como objetivo ser un punto de contacto único entre los usuarios y los que proveen el servicio (ITIL Foundation 2014). Es por ello que son responsables de proveer soluciones a los incidentes, problemas o preguntas de TI que ocurran a un usuario relacionado con los productos o servicios de la empresa que estén interrumpiendo la continuidad del negocio. Las vías de comunicación más frecuentes a esta área son llamadas telefónicas o por correo electrónico.

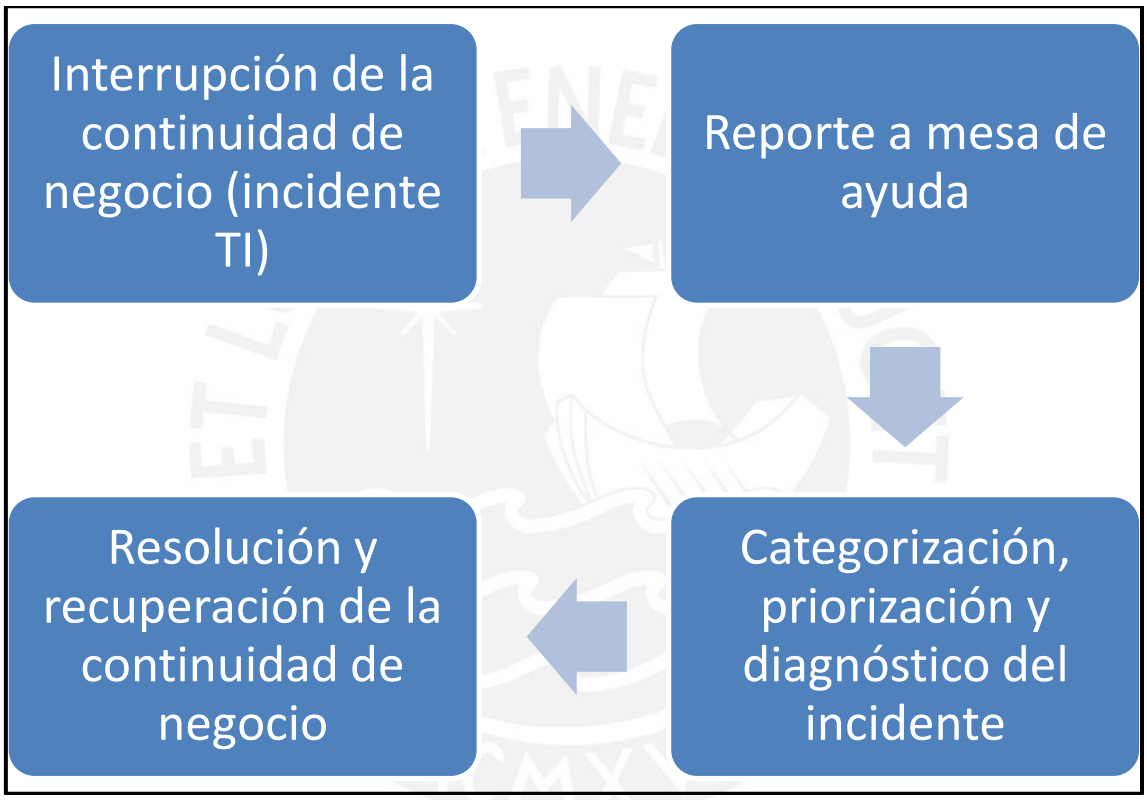


Figura 1.1: Ejemplo de flujo de trabajo de una mesa de ayuda contra incidentes.

Los usuarios pueden reportar en su servicio de TI problemas que no les permita laborar de manera normal. Mientras no haya sido resuelto, el incidente puede incurrir en perjuicios laborales de niveles variados dependiendo del impacto negativo del incidente. En casos extremos de un impacto negativo severo, puede afectar los procesos críticos del modelo de negocio, incurriendo en pérdidas de recursos monetarios y de tiempo.

The Economist reporta que 60% de las organizaciones empresariales tienen un equipo de respuesta a incidentes y un plan de respuesta. Compañías que estén preparadas con un plan de respuesta a incidentes sienten que están preparados o algo preparados en un 94% comparado contra los que no tienen un plan de respuesta con un 38%. Además, reporta que los incidentes más comunes no fueron deliberadamente maliciosos ni deliberados de fuera de la empresa, sino accidentales por acción de empleados (The Economist 2014: 7-8). Según los resultados del ESET Security Report

2013, en América Latina, 3 de cada 4 empresas no tienen claramente definido que realizar en casos de incidentes y solo un 26% cuenta con un plan de respuesta. (Diario TI 2014)

Entre los planes de respuesta, se encuentra la delegación de incidentes a un equipo técnico especializado en la resolución de incidentes, conocido como Gestión de Incidentes, quienes cubren los fallos, preguntas o consultas hechas por los usuarios (Tupia 2013: 134). Es por ello que es importante que se deleguen y asignen los incidentes de manera apropiada de manera que el impacto negativo se reduzca y se restablezca la función normal del equipo de TI para la continuación del modelo de negocio. A este problema se le conoce como asignación de tareas o "task scheduling", la manera más óptima de asignar tareas a máquinas o personas dado ciertas condiciones, restricciones y objetivos a cumplir.

Si bien se intenta de varias maneras que las tareas lleguen a un determinado tiempo y se calcula cuando una máquina demora en procesar esta tarea, en el mundo real el escenario ciertamente diferente. Tómese como ejemplo una fábrica en la cual se procesa materia prima: si bien una correcta logística de compras y producción pueden determinar el arribo de la materia prima y calcular el tiempo con que una máquina o máquinas procesan la materia en un intervalo de tiempo; pueden haber imprevistos como un arribo tardío o dentro de un espacio de tiempo, así como las máquinas pueden sufrir imprevistos que puedan ponerla fuera de uso, tales como averías o desgaste.

En nuestro caso de estudio tenemos un set de condiciones dadas tomadas de una empresa. En este trabajo simularemos una mesa de ayuda o "help desk" en una empresa. Dicha mesa de ayuda está conformado por técnicos de niveles de experiencia diversos los cuales tienen como objetivo recibir y solucionar los problemas relacionados a las Tecnologías de Información o TI que ocurran en la empresa, los cuales pueden ocurrir en cualquier momento. Esta situación tiene como objetivo:

- Asignar los técnicos con suficiente experiencia para resolver un incidente asignado y resolverlo en el menor tiempo posible.
- Priorizar los incidentes de mayor urgencia; es decir, que tengan un impacto negativo severo en la continuidad del negocio.
- Minimizar los tiempos muertos de los técnicos. Todos los técnicos deben de tener en lo posible trabajos asignados.

Por otro lado, se debe de tener en cuenta las limitaciones del problema en el caso de una mesa de ayuda, en donde es posible que se tenga en consideración diferentes clasificaciones de la severidad de un incidente. También debe de considerarse que puede haber diferentes rangos entre los técnicos, divididos principalmente por sus desempeños y conocimientos y/o divididos por la estructura organizacional que tenga la empresa. Con esto, podemos decir que los problemas que sean clasificados como alta prioridad, requerirán un técnico altamente capacitado, dejando de lado los que tengan menos experiencia.

En el presente trabajo de fin de carrera se realizará la aplicación del algoritmo GRASP al algoritmo MinIncrease para la asignación eficiente de incidentes en una mesa de ayuda.

2 Marco teórico

En este apartado veremos conceptos que se relacionan con el problema:

2.1 Marco conceptual

2.1.1 Conceptos relacionados al problema

2.1.1.1 Complejidad Algorítmica

Este proyecto posee una complejidad algorítmica alta. Esto es por todas las combinaciones posibles que posee el problema para otorgar un resultado final. Sin embargo, a pesar de las numerosas combinaciones posibles, solo unas cuantas son resultados “ideales”.

La complejidad algorítmica divide estos problemas en dos grupos:

- Problemas P o Polinomiales: los cuales su tiempo de procesamiento a la solución crece en proporción a los datos insertados. Estos problemas son considerados sencillos y son capaces de otorgar una solución óptima; es decir, la mejor solución posible.
- Problemas NP o No Polinomiales: los cuales no poseen una solución óptima por no tener un algoritmo polinomial que los resuelva debido a la cuantiosa demanda de cálculo computacional requerido. Sin embargo, se esfuerza por obtener una solución cercano al óptimo o satisfactorio.

Además de esto, se consideran como “NP-Complete” a los problemas con dificultad mayor a las cuales está comprobado de que no existe solución polinómica posible debido a la gran cantidad de datos de entrada. Los “NP-Hard” o NP-difícil son problemas en donde la complejidad es tan difícil como un NP-Complete y va más allá. Los problemas de optimización como el que se plantea dar una solución con este proyecto se ubican en NP-Hard por su complejidad (Brassar, Bratley 1996).

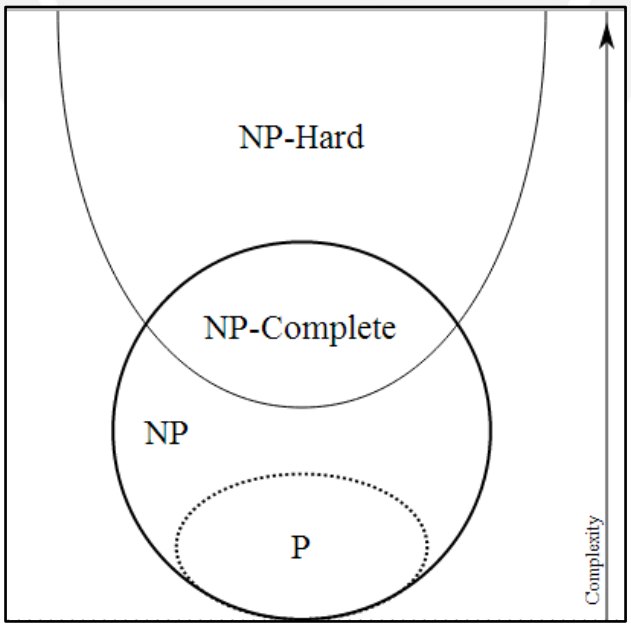


Figura 1.2: Diagrama de Euler para la ubicación de los problemas P, NP, NP-Completo y NP-Hard.

2.1.1.2 Optimización Combinatoria

La optimización combinatoria es un tema que reside en las matemáticas y las ciencias de la computación. Este tema habla acerca de encontrar un objeto óptimo de una serie de objetos finitos, hallando un set de soluciones posibles, y tratando de hallar la mejor solución de todas ellas. El objeto que se busca puede ser un número, una lista de números, un gráfico, etc. La complejidad algorítmica de estos algoritmos es de NP a NP-Hard. El reto es desarrollar algoritmos que sean efectivamente o prácticamente mejores que enumerar todas las soluciones posibles (Lee 2004).

La instancia de un problema de optimización es la siguiente (Papadimitriou, Steiglitz 1998: 4):

Dado un par *(F, c),* donde F es el set de puntos factibles; c la función costo, el mapeo:

$$c: F \rightarrow R$$

El problema es hallar un $f \in F$ donde:

$$c(f) \leq c(y), \text{ para todo } y \in F$$

Dicho punto *f* es llamado la solución óptima global en una determinada instancia o también simplemente llamado la solución óptima.

La optimización combinatoria abarca un conjunto de problemas clásicos a los temas de inteligencia artificial, algunos ejemplos son:

- Problema del vendedor viajero – cuyo objetivo es ubicar la ruta más corta posible entre la distancia de las ciudades visitándolos todos una sola vez y regresando a su punto de partida.
- Problema del ruteo – un equipo de vehículos que deben entregar paquetes a cierta cantidad de clientes, ubicando las rutas más óptimas.
- Problema de asignación – se tienen un número determinado de agentes y tareas. Se requiere asignar las tareas a los agentes de manera que el costo total de la asignación se minimize.

De estos problemas, se pueden desprender varios usos en temas de optimización para las áreas de producción, informática, transporte, etc. En dichos casos, se busca la mejor combinación para el uso de recursos.

En su forma más general se puede plantear de esta manera:

Optimizar: $\boldsymbol{f(x)}$
Sujeto a: $\boldsymbol{g_i(x) \geq 0, i = 1 \ldots m}$
$\boldsymbol{h_j(x) \geq 0, j = 1 \ldots n}$

Donde:

- $\boldsymbol{f(x)}$ es la función objetivo y representa un valor que debe de ser optimizado (maximizado o minimizado).
- Tanto $\boldsymbol{g_i(x) \geq 0, i = 1 \ldots m}$ y $\boldsymbol{h_j(x) \geq 0, j = 1 \ldots n}$ representan a las restricciones del problema y especifican las condiciones de toda solución viable para el mismo (Tupia 2009: 92).

2.1.1.3 Problema de Asignación Generalizada

El Problema de Asignación Generalizada o GAP, es uno de los problemas que se apoyan en la Optimización Combinatoria. El GAP consiste en tener un número de

agentes (que pueden ser objetos o personas) y un número de tareas. Cada recurso puede ser asignado a una tarea, incurriendo en un costo y ganancia que puede variar dependiendo de la combinación recurso-tarea. Asimismo, cada agentes limitado por una capacidad –por ejemplo, un presupuesto- y no puede tomar una tarea que exceda la totalidad de sus recursos. El objetivo es encontrar la lista de asignaciones de "n" trabajos a "m" agentes con el mínimo costo posible, tal que cada trabajo es asignado a un único agente sujeto a sus restricciones de capacidad (Dress, Yinfeng, Binhai 2007).

Para este caso de estudio, este problema puede aplicarse al contexto:

- Los recursos serán el personal que se encuentra en el centro de servicio.
- La capacidad del personal es la experiencia que tiene dicha persona en algún tema.
- Las tareas serán los incidentes a resolver que requieren de cierta experiencia para ser resueltas.
- El coste de la tarea será el tiempo que demora en terminar dicha tarea respecto a la dificultad de dicho problema dependiendo que técnico lo resuelva.

Las soluciones algorítmicas para este problema pueden ser exactas o heurísticas. Los algoritmos exactos o exhaustivos permiten dar la respuesta óptima al problema. El problema de las soluciones exactas radica en el costo computacional alto para resolver dicho problema de manera exacta por su búsqueda exhaustiva. Los métodos heurísticos solucionan dicho problema a costo de la solución exacta, pero dando una solución razonable a un costo razonable.

2.1.1.4 Asignación de Tareas

La asignación de tareas o “Scheduling” consiste en obtener un plan de ejecución en la cual se asignan tareas a máquinas, con el objetivo de que dicho plan cumpla un objetivo de la manera más efectiva posible. Este objetivo puede ser: cumplir las tareas en el menor tiempo posible, realizar los trabajos más importantes en un intervalo de tiempo, minimizar tardanzas, etc. (Pinedo 2010, 1).

Las máquinas pueden representar distintas cosas: máquinas de taller, pistas de aterrizaje en un aeropuerto, personal de construcción, etc. Las tareas pueden tener tiempo de aparición y un tiempo de entrega.

Generalmente un problema de asignación de tareas se suele representar por medio de los tres campos de Graham:

$$\alpha \mid \beta \mid \gamma$$

En donde:

- α representa el entorno de las máquinas que se va a trabajar: una máquina, máquinas idénticas en paralelo, máquinas distintas, Flow Shop, etc.
- β representa las características y restricciones en el problema: tiempos de lanzamiento, pre procesamiento, restricción de elecciones, tiempos de mantenimiento, etc.
- γ representa el objetivo a alcanzar: minimización del tiempo de finalización del problema (makespan), tardanzas máximas, tiempo total de cumplimiento con peso, etc.

Los modelos de planificación suelen ser de dos tipos:

- Modelos de Planificación Determinísticos: Estos modelos asumen que se tienen un finito número de trabajos que tienen que ser asignados teniendo en cuenta uno o más objetivos a minimizar. En los trabajos, se tiene conocimiento de los tiempos de procesamiento, de los tiempos de arribo y los tiempos de entrega antes del procesamiento.
- Modelos de Planificación Estocásticos: Similar a la planificación anterior, con la diferencia que es posible que no se tenga conocimiento de los tiempos de procesamiento; es decir, tiene un tiempo de procesamiento variable. Dicho conocimiento del tiempo exacto se tendrá después del procesamiento.

Además, puede tener otras dos clasificaciones:

- Modelos Fuera de Línea: Estas planificaciones contienen todas las tareas a presentarse desde el día cero, o bien se pueden presentar en diferentes días (en un tiempo llamado "release date" o fecha de lanzamiento), pero se sabe cuándo y cuántas tareas van a llegar, así como datos de peso o prioridad.
- Modelos en Línea: Estas planificaciones tienen como característica el desconocimiento de la llegada de las tareas y la cantidad de ellas. No se va a saber nada de los datos de la tarea hasta que aparezcan y sean registrados al problema.

En el presente proyecto se considerará *un modelo de planificación estocástico en línea*, puesto que en el escenario escogido de simulación no se tiene conocimiento de cuando aparecerá un incidente a la mesa de ayuda, ni cuando un técnico terminará de solucionar dicho problema.

2.1.1.4.1 Asignación de Tareas: Notación Estocástica

La notación para las tareas estocásticas es la siguiente:

- P_{ij} = el tiempo de procesamiento aleatorio de la tarea j a la máquina i.
- $E[P_{ij}]$ = el tiempo esperado de procesamiento de la variable P_{ij}
- R_j = el tiempo aleatorio de aparición de la tarea j.
- w_j = el peso (importancia) de la tarea j.

2.1.1.4.2 Asignación de Tareas: Máquinas en Paralelo sin Relación

El α de este problema será Máquinas en Paralelo sin Relaciones o Rm. Este entorno tiene como característica que hay *m* máquinas con diferentes velocidades de procesamiento. La velocidad de la máquina *i* para procesar la tarea *j* se denota por v_{ij}. El tiempo de procesamiento de la tarea *j*, p_{ij}, es igual a p_j/v_{ij}. (Pinedo 2010: 14)

Esto se puede simplificar si se tiene que las velocidades de las máquinas no dependen de las tareas ($v_{ij} = v_i$), llamado Máquinas en Paralelo con Diferentes Velocidades; y aún más si las velocidades de las máquinas son idénticas ($v_i = 1$ y $p_{ij} = p_j$ para todo *j*), pasando a que sea un problema de Máquinas en Paralelo.

Para este problema se está considerando Máquinas en Paralelo sin Relaciones, debido a que tendremos trabajos con tiempos variables de procesamiento; y tenemos técnicos que, debido a su experiencia o falta de ella, pueden procesar la tarea más rápida o lenta respectivamente.

2.1.1.4.3 Asignación de Tareas: Restricción en Elección de Máquinas

El β de este problema será Restricción en Elección de Máquinas o M_j. Esta restricción se encuentra en problemas de Máquinas en Paralelo. Cuando está presente, no todas las *m* máquinas pueden procesar una tarea *j*. M_j denota el set de máquinas que puede procesar dicha tarea (Pinedo 2010: 17).

Esto va a estar presente en nuestro problema de la manera que ciertos incidentes no se le van a asignar a ciertas personas por no tener la suficiente experiencia para resolverlo de una manera rápida (por ejemplo, un técnico junior o practicante no se le van a asignar incidentes de alta urgencia o de alta complejidad).

2.1.1.4.4 Asignación de Tareas: Tiempo de Cumplimento Total con Pesos

El γ de este problema será Tiempo de Cumplimiento Total con Pesos ($\Sigma w_j C_j$). La suma de los tiempos de cumplimiento de las *n* tareas nos da una indicación del inventario total causado por la asignación. Comúnmente se le conoce como flujo de tiempo con pesos (Pinedo 2010: 19).

Esto va para el proyecto en la manera de que el objetivo es la sumatoria de los tiempos de cumplimiento de las tareas con el peso debido a que nos es relevante el peso para la prioridad de las tareas. Es necesario priorizar los trabajos con mayores pesos debido a que tienen un alto nivel de urgencia.

2.1.2 Conceptos relacionados a la propuesta de solución

2.1.2.1 Tiempo de Procesamiento con Peso Esperado Más Corto Primero - WSEPT

Weighted Shortest Expected Processing Time First es un tipo de asignación en situaciones estocásticas. Es una versión estocástica del WSPT o Weighted Shortest Processing Time First que utiliza los tiempos esperados para hacer el cálculo de cuál tarea se asignará primero. Esta asignación es suficiente como política de asignación para máquinas simples. La fórmula de asignación es la siguiente:

$$w_j / E[P_j]$$

Esto permite minimizar la suma esperada de los tiempos de cumplimiento con peso, priorizando los trabajos que tengan un peso importante, pero a la vez, considerando el tiempo esperado que se demoran en terminar (Pinedo 2010: 268).

2.1.2.2 Algoritmo MinIncrease

El algoritmo MinIncrease es un algoritmo propuesto por Nicole Megow, Marc Uetz y Tjark Vredeveld. Este algoritmo es una política de asignación estocástica y en línea que minimiza el valor esperado del tiempo de cumplimiento con pesos.

Este algoritmo se presenta en la siguiente fórmula:

Se tiene una tarea *j* y máquinas *i*. Se escoge a la máquina *i* que nos dé el menor valor de la siguiente fórmula:

$$z(j,i) = w_j \cdot \sum_{k \in H(j),\, k<j,\, k \to i} \mathbb{E}[P_k] + \mathbb{E}[P_j] \cdot \sum_{k \in L(j),\, k<j,\, k \to i} w_k + w_j \mathbb{E}[P_j] .$$

Figura 1.3: Fórmula del Algoritmo Minincrease

w_j y $E[P_j]$ se refieren al peso y al tiempo esperado de la tarea *j* respectivamente.

H(*j*) se refiere a un set de tareas que tengan una prioridad mayor o igual a la tarea j. Asimismo, se refiere a L(*j*) como las tareas que tengan una prioridad menor a j. Esta prioridad se calcula por medio de WSEPT señalado anteriormente.

Aquí, se está tomando dos sumatorias:

- Sumatoria de los tiempos esperados de las tareas ya asignadas a la máquina i que cumplan con el requisito de pertenecer a H(*j*).
- Sumatoria de los pesos de las tareas ya asignadas a la máquina i que cumplan con el requisito de pertenecer a L(*j*).

Megow, Uetz y Vredeveld prueban que su algoritmo es una ρ-aproximación (el resultado se aproxima al óptimo), por lo tanto se espera que su desempeño esté garantizado (Megow, Uetz, Vredeveld 2006).

2.1.2.3 Algoritmos Metaheurísticos

Los algoritmos metaheurísticos son métodos de solución que guían a un proceso de búsqueda heurístico, el cual su meta es explorar el espacio de búsqueda de manera eficiente para encontrar soluciones óptimas. Usualmente estos algoritmos son aproximados y no determinísticos, y pueden aplicarse a una gama de problemas.

Existen diversos algoritmos metaheurísticos que se pueden clasificar de diversas maneras (Blum, Roli 2003: 272).

- Por ser o no inspirados en la naturaleza.
- Basado en población o en un solo punto.
- Objetivo estático o dinámico.
- Una o varias estructuras de vecindad.
- Con o sin memoria.

Con esto, se permite que un algoritmo heurístico muy general sea optimizado para el contexto del problema, obteniendo un mejor rendimiento.

2.1.2.4 Algoritmo GRASP

Una de las principales técnicas meta heurísticas es el algoritmo GRASP o “Greedy Randomized Adaptative Search Procedure”, desarrollada por Thomas Feo y Mauricio Resende en 1989.

Sus siglas describen lo que el algoritmo realiza:

- “Greedy” (Voraz) – similar al algoritmo voraz, escoge el mejor candidato más próximo
- “Randomized” (Aleatorio) – después de tener la lista de candidatos escoge uno de ellos al azar.
- “Adaptative” (Adaptativo) – por su capacidad de modificarse al contexto del problema
- “Search” (Búsqueda) – por que realiza una búsqueda dentro de un espacio, realizando la evaluación en los candidatos de ese espacio.

Los algoritmos GRASP tienen una ventaja sobre los algoritmos voraces, que en vez de seleccionar solamente el mejor valor de la función objetivo, se amplía o "relaja" este criterio de tal manera de que en vez de seleccionar un único elemento, se pueda seleccionar un conjunto de elementos candidatos a ser parte del conjunto solución y que cumplen ciertas condiciones, y de estos, se realizará una selección aleatoria de alguno de los elementos.

Según Feo y Resende, "hay dos fases dentro de cada iteración GRASP: la primera construye inteligentemente una solución inicial vía una función adaptativa aleatoria golosa; la segunda aplica un procedimiento de búsqueda local a la solución construida esperando encontrar una mejora" (Feo, Resende 1995: 109).

El procedimiento es el siguiente:

- Se tiene un set de candidatos a ser conjunto solución.
- Se escoge una lista restringida de candidatos o RCL (Restricted Candidates List), también llamado Paso de Construcción.
- Se elije un elemento de la RCL.
- Se inicia un proceso de mejora, que ya sea por medio de repeticiones u optimización, se tiene un resultado mejorado.
- En caso de que la condición de parada (se llegó al óptimo, o condición de número de iteraciones o tiempo) no esté completa, volver al punto 2.
- Se muestra la mejor solución.

En el paso de construcción, la lista restringida de candidatos es obtenida mediante el mejoramiento del criterio goloso (tomar siempre el próximo mejor), como podemos ver aquí.

Sean:

$$\beta \coloneqq Mejor\,\{f(x): x\ \in\ N\}$$

$$\gamma \coloneqq Peor\{f(x): x\ \in\ N\}$$

El RCL estará definido por:

$$RCL = \{x\ \in N: \beta \leq f(x) \leq\ \beta + \alpha(\gamma - \beta)\}$$

El parámetro α se llama parámetro de relajación, y es lo que nos permite construir una lista de soluciones "aleatorias". Este parámetro puede modificarse de acuerdo al criterio más adecuado que se requiera.

Se observa que:

- $\alpha = 0 \rightarrow$ Criterio totalmente goloso.
- $\alpha = 1 \rightarrow$ Criterio totalmente aleatorio.
- $0 < \alpha < 1 \rightarrow$ Criterio goloso y aleatorio

Esto es lo que permite tener un componente adaptativo, puesto que el parámetro puede variar en cada iteración.

En el paso de mejora, se toma el resultado entregado en la fase de construcción, y se mejora a través de funciones que permitan mejorar el conjunto solución por medio de un conjunto de "vecindad". Este permitirá intercambiar uno a uno los elementos de la solución con aquellos que no pertenecen a la solución.

2.1.3 Otros conceptos

2.1.3.1 Gestión de Incidentes

Un evento que surge en una infraestructura de TI puede desembocar en un incidente, el cual reduce o interrumpe la calidad del valor del servicio. Los problemas son la causa del incidente, los cuales deben de ser averiguados para la resolución completa. (Tupia 2013: 134)

La gestión de incidentes debe de tener una el objetivo de:

- Restablecer la operación normal del servicio lo más rápido posible ante la ocurrencia de algún incidente.
- Minimizar el impacto negativo en la operación de negocio y dentro de niveles de calidad y disponibilidad acordados previamente.

La gestión de incidentes debe de estar ubicado en un ámbito, que permite registrar directamente los eventos comunicados por los usuarios y por el personal técnico.

Los siguientes conceptos deben de ser tomados en cuenta para el manejo de incidentes:

- Modelo de atención de incidentes: posee pasos predefinidos para escalas de tiempo, listas de actividades, procedimientos de escalamiento y asignación de responsabilidades.
- Escalas de tiempo: se requiere escalas de tiempo de resolución, notados en los documentos de Acuerdos de Nivel de Servicio (SLA) y Acuerdos de Nivel de Operaciones (OLA).
- Incidentes graves: que requieren procedimientos particulares por ser de alto impacto negativo a la continuidad del servicio o del negocio.
- Métricas: para el monitoreo e informe de la eficacia y eficiencia de las acciones correctivas.

Las actividades de la gestión de incidentes (según ITIL) son las siguientes:

- Identificación y Registro de incidentes.
- Clasificación de los incidentes.
- Priorización.
- Diagnóstico (resolución en caso de que sea incidentes menores).
- Escalado (en caso de que el incidente necesite un nivel superior de asistencia).
- Investigación (razones del por qué ocurrió el incidente).
- Resolución y Recuperación.
- Cierre del Incidente.

La tesis no se enfocará a un marco regulatorio de ITIL, ni a cubrir todo el aspecto de la gestión de incidentes, pero sí otorga un en el proceso de asignación en el ámbito de identificación, clasificación y priorización.

3 Estado del arte

En este apartado veremos el estado del arte para otorgar la solución del problema de asignación de tareas.

3.1 Formas exactas de resolver el problema

3.1.1 Programación Lineal

La programación lineal es una técnica de modelado usada en el proceso de toma de decisiones, en donde dado un conjunto de m ecuaciones o inecuaciones lineales, y n número de variables de decisión, se requiere hallar valores no negativos de las variables que satisfagan las restricciones y maximicen o minimicen la función objetivo (Mejía y otros 2012).

Gráficamente, se puede representar las restricciones como ecuaciones lineales en un plano. Estas líneas formarán un borde en un polígono en donde se establecerá la región factible que la solución puede tomar, siendo las del más extremo la solución más efectiva posible.

La programación lineal es comúnmente utilizada para negocios y economía, pero también puede servir para problemas de ingeniería, como asignaciones. De esta manera, se puede tener una solución exacta al problema de asignación, debido a que puede acoplarse lo que se tiene como incidentes: órdenes de prioridad, objetivos, etc.

Sin embargo, la programación lineal tiene limitaciones en cuanto a tiempos de procesamiento y problemas NP, que no poseen representación en polinomios o datos inciertos. Para este caso de estudio, la cantidad de agentes e incidentes, más el factor impredecible de la asignación en línea hace prácticamente imposible resolverlo de manera completamente exacta.

3.2 Formas aproximadas de resolver el problema

3.2.1 Algoritmo MinIncrease

El algoritmo MinIncrease (Megow, Uetz, Vredeveld 2006) descrito anteriormente responde al problema de asignación de máquinas en paralelo idénticas. Sin embargo, se necesita de consideraciones adicionales en este problema debido a las restricciones de las máquinas (M_j), así como buscar de que todos estén debidamente “ocupados”.

3.3 Productos comerciales para resolver el problema

Se ofrecen una variedad de productos para apoyar en el problema de asignación de tareas en una mesa de ayuda. Estos productos se encargan de apoyar a la gestión de servicios en tecnologías de información o ITSM. Esta lista certificada por ITIL según la página Pink Elephant (Pink Elephant 2014):

3.3.1 CA Service Desk Manager Suite r12.9

CA Service Desk Manager Suite es una plataforma de soporte a la mesa de ayuda creada por CA Technologies en donde permite, tanto en versión desktop como móvil, automatización extensiva, “Software as a Service”, análisis y gráficas con el objetivo

de reducir riesgos y costos; e incrementar la productividad del equipo en la mesa de ayuda. (CA Technologies 2014).

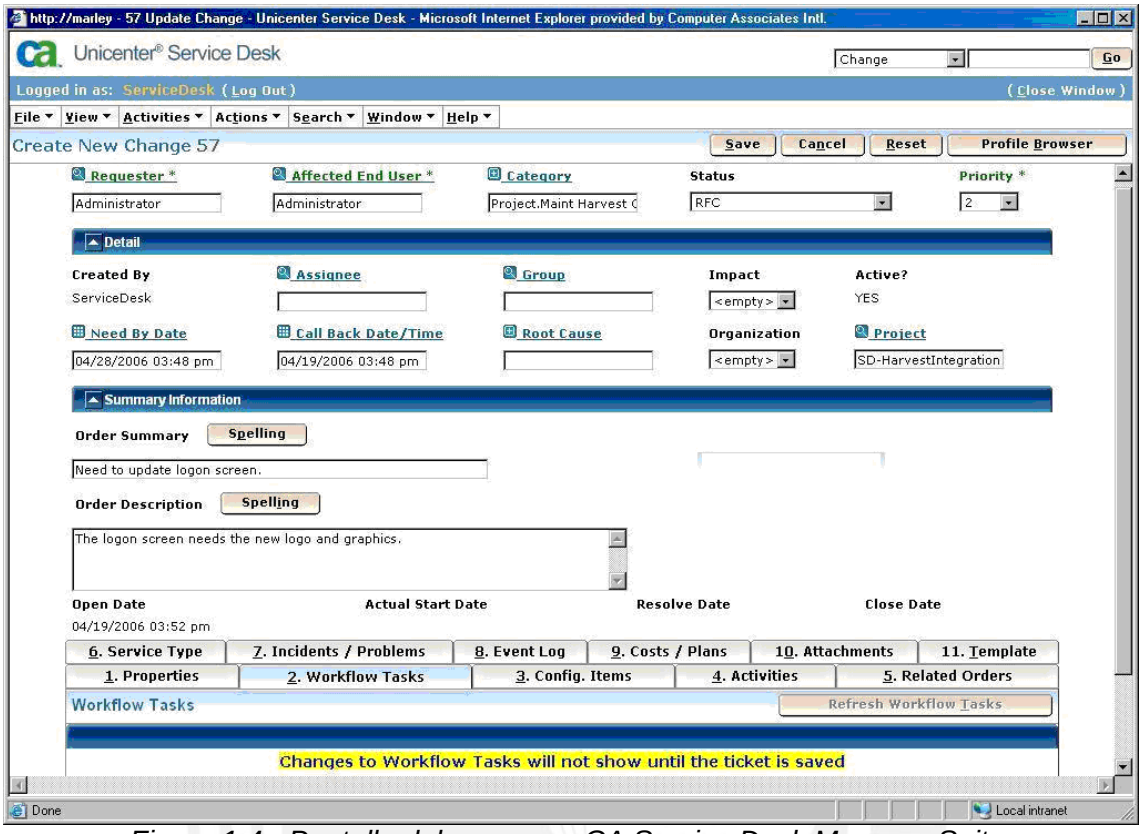


Figura 1.4 - Pantalla del programa CA Service Desk Manager Suite

3.3.2 HEAT Service Management 2014

HEAT Service Management, creado por FrontRange Solutions USA Inc., es una solución robusta y flexible para la gestión de servicios. Tiene facilidades para requerir, planear, aprobar, cambiar, auditar y controlar los cambios necesarios al servicio. Permite elaborar flujos de trabajo de complejidades variadas (FrontRange Solutions 2014).

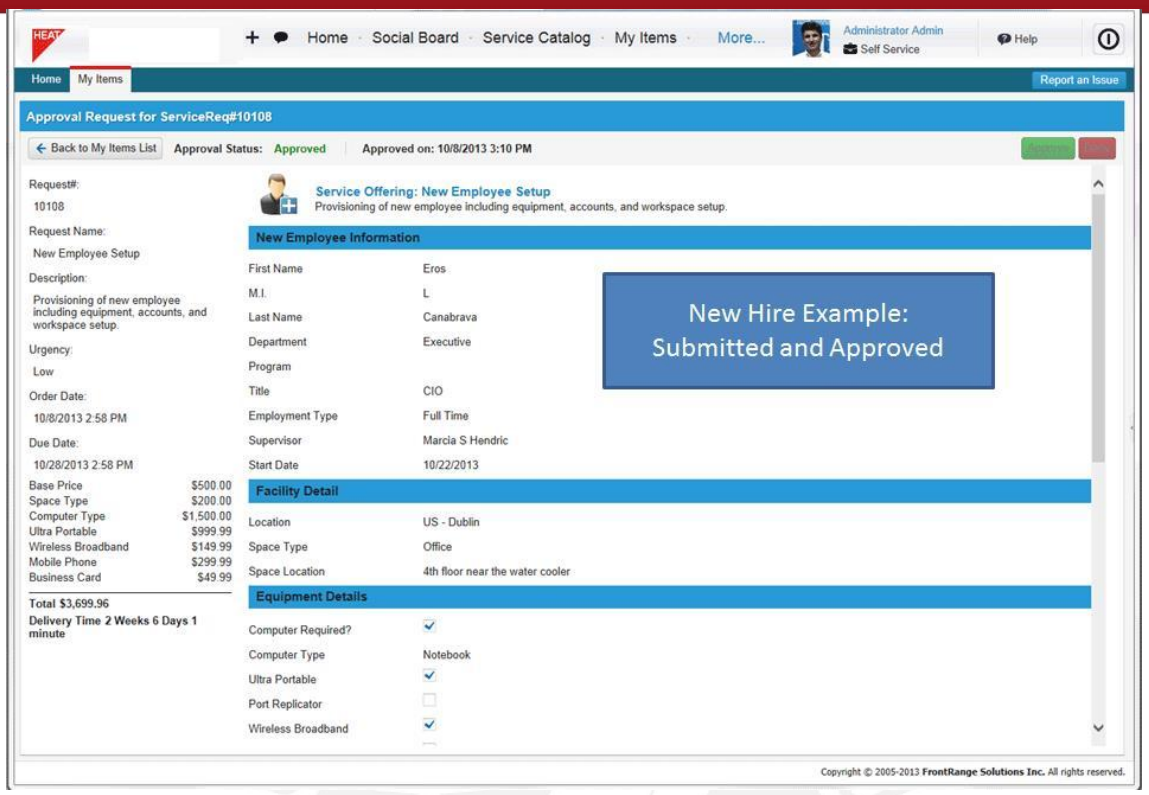


Figura 1.5 – Pantalla del sistema HEAT Service Management

3.3.3 MSM v12 – Marval Service Management

MSM, desarrollado por Marval Software Ltd. provee soluciones para mesas de ayuda. Es compatible con los marcos ITIL e ISO/IEC 20000. Ofrece mejorías de costo-eficiencia, acceso rápido a ítems de conocimiento, notificaciones y gráficas en tiempo real (Marval Software 2014).

Figura 1.6: Pantalla del sistema Marval Service Management

3.3.4 OMNITRACKER ITSM v5

OMNITRACKER ITSM v5 es un sistema que ofrece soluciones de servicios modelado a ITIL con plantillas de proceso respectivos, incluido la de gestión de incidentes; así como una rápida implementación y escalabilidad garantizada (OMNINET 2014).

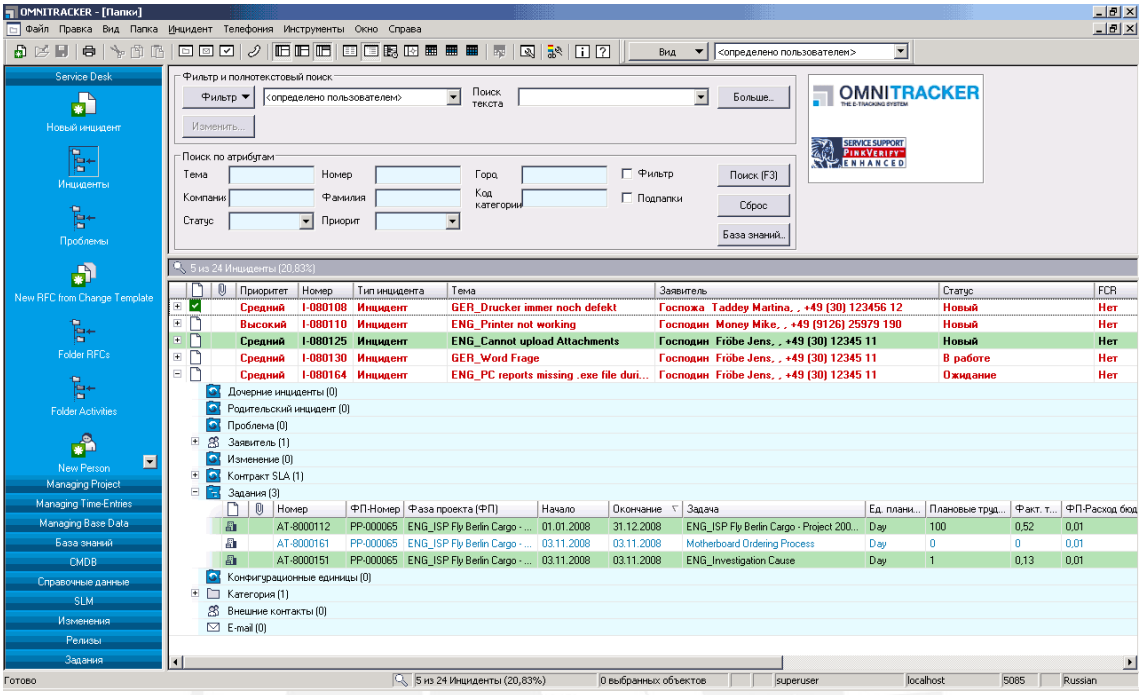


Figura 1.7 – Pantalla del sistema OMNITRACKER ITSM v5

3.3.5 ProactivaNET v8

ProactivaNET es un sistema de ITSM construido en España. El software incluye los procesos modelados de ITIL de gestión de problemas e incidentes, gestión de problemas y gestión de cambios y entregas. Incluye vista por móvil. Puede integrarse con otros productos de Proactiva NET (Espacio Tecnológico Molinon 2014).

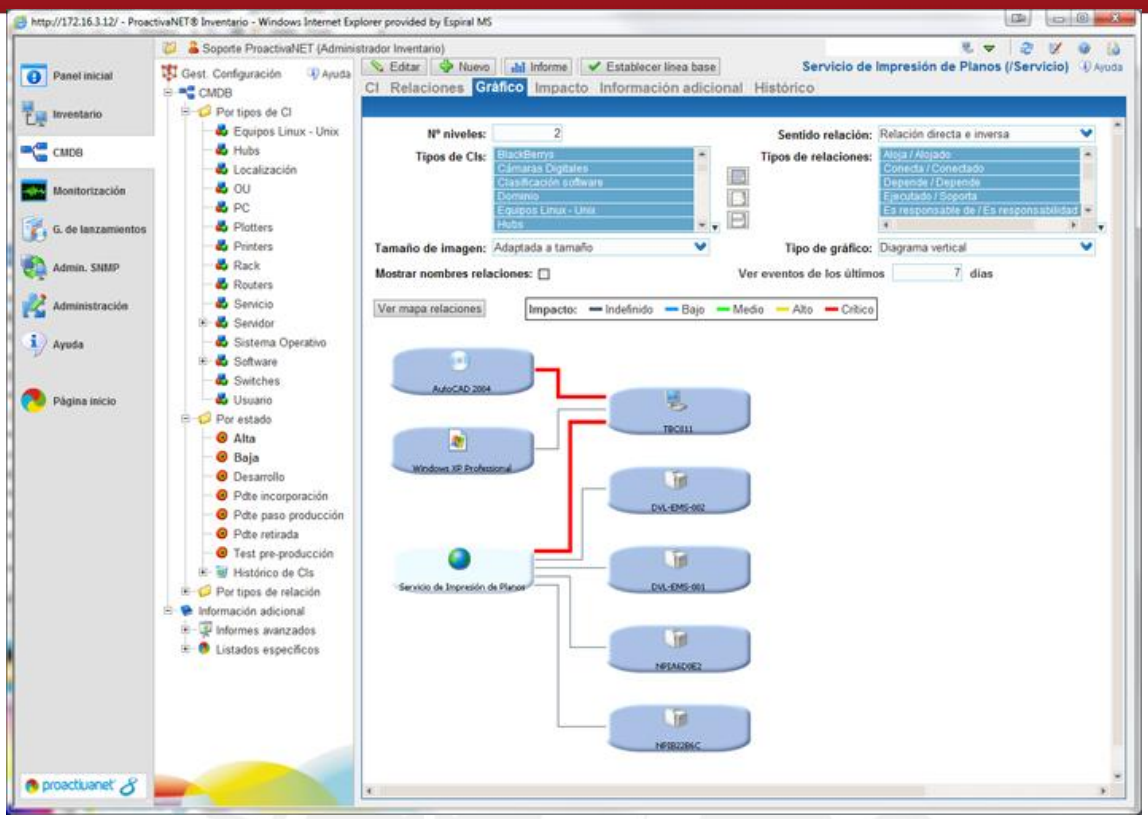


Figura 1.8 – Pantalla del sistema ProactivaNET v8

3.3.6 HP Service Manager v9.30

HP Service Management es el software desarrollado por Hewlett-Packard Development Company. Ofrece varios módulos de servicio para TI, entre ellos un apoyo automatizado para las mesas de ayuda. Puede dar seguimiento a incidentes de TI, apoyo en auto-servicios de TI y otorgar manejo de incidentes y problemas (Hewlett-Packard Development Company 2014).

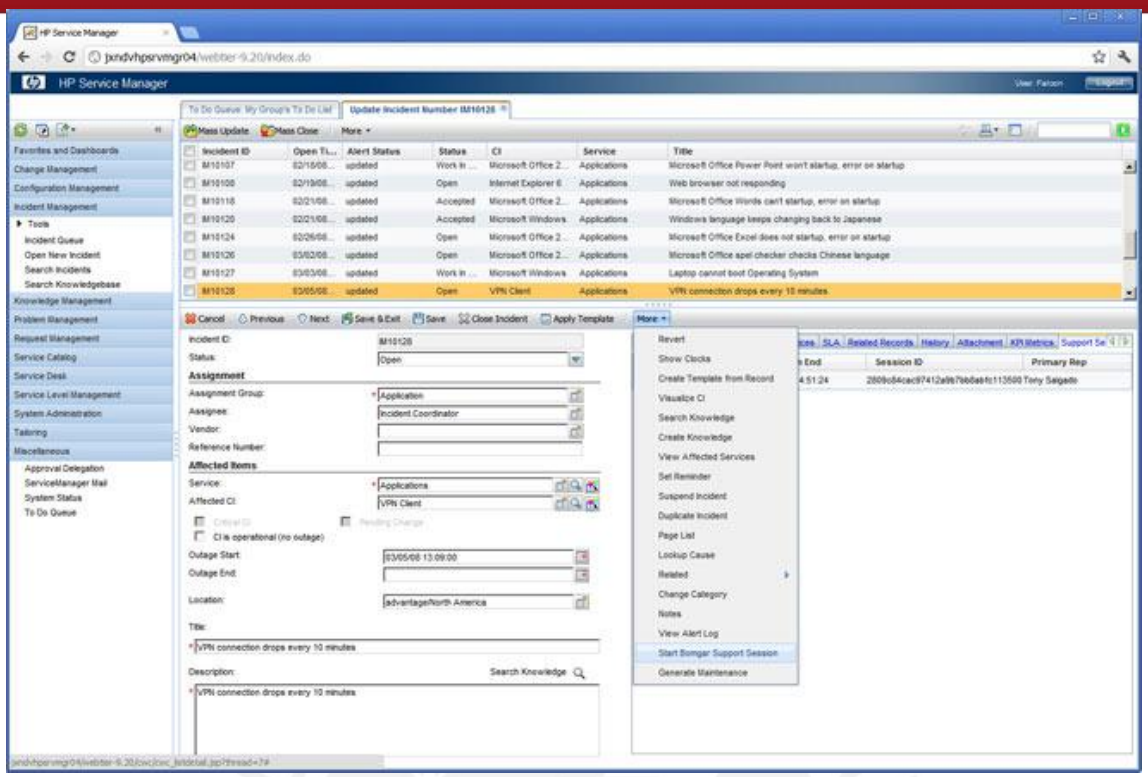


Figura 1.9 – Pantalla del sistema HP Service Manager v9.30

3.3.7 Comparación de los ejemplos por Pink Elephant

Glosario:

- Nivel de cumplimiento:
  - Bronce: demuestra el cumplimiento con el modelo de ITIL con funcionalidad, automatización y documentación.
  - Plata: Lo mencionado en el nivel bronce y ha sido aplicado siguiendo ITIL en al menos tres clientes.
  - Oro: Lo mencionado en el nivel plata, y ha presentado evidencia de su funcionamiento por medio de reportes y capturas de pantalla.
- Procesos:
  - AVM – Gestión de disponibilidad.
  - CAP – Gestión de capacidad.
  - CHG – Gestión de cambios.
  - EV – Gestión de eventos.
  - FM – Gestión de finanzas.
  - IM – Gestión de incidentes.
  - ITSCM – Gestión en la continuidad de servicio de TI.
  - KM – Gestión del conocimiento.
  - PM – Gestión de problemas.
  - REL – Gestión de lanzamiento y aplicación.
  - RF – Cumplimiento de pedidos.
  - SACM – Gestión de activos de servicio y configuración.
  - SCM – Gestión del catálogo de servicios.
  - SLM – Gestión de nivel de servicio.
  - SPM – Gestión de portafolio de servicios.

Sistema	Nivel de Cumplimiento	Procesos
CA Service Desk Manager Suite r12.9	Bronce	AVM, CAP, CHG, EV, FM, IM, ITSCM, KM, PM, REL, RF, SACM, SCM, SLM, SPM
HEAT Service Management 2014	Bronce	CHG, IM, KM, PM, REL, RF, SACM, SCM, SLM, SPM
MSM v12	Plata	AVM, CAP, CHG, EV, FM, ITSCM, KM, PM, REL, SACM, SPM
OMNITRACKER ITSM v5	Plata	CHG, EV, IM, KM, PM, RF, SACM, SCM, SLM
ProactivaNET v8	Oro	CHG, IM, KM, PM, REL, RF, SACM, SCM, SLM, SPM
HP Service Manager v9.30	Oro	CHG, IM, KM, PM, REL, RF, SACM, SCM, SLM, SPM

Figura 1.10 – Tabla de comparación de los ejemplos anteriores

3.4 Problemas relacionados

3.4.1 Asignación de Tareas Estocásticas

- Schulz propone algoritmos provee soluciones algorítmicas para asignación estocástica (Schulz 2005).
- Uetz escribe sobre algoritmos tanto para asignación determinística y estocástica (Uetz 2002).

3.4.2 Asignación de Tareas en Estocásticas o en Línea

- Afrati y otros utilizan esquemas para la solución de minimizar el tiempo de cumplimiento con pesos con tiempos de lanzamiento (Aftati y otros 1999).
- Anderson y Potts utiliza asignación de tareas para minimizar el tiempo de cumplimientos con pesos en una sola máquina (Anderson y Potts 2004).
- También lo hacen Fiat y Woeginger para minimizar el objetivo mencionado anteriormente (Fiat y Woeginger 1999).
- Heydenrich, Muller y Uetz se inspiran en MinIncrease para hacer un mecanismo genérico y descentralizado con una visión miope para otorgar solución a tareas en línea (Heydenrich, Muller y Uetz 2006).

3.4.3 Asignación de Tareas en Máquinas sin Relación

- Schulz y Skutella tienen publicaciones sobre asignación de tareas en máquinas sin relación (Schulz y Skutella 2005).

3.4.4 Flow Shop Scheduling

Flow Shop Scheduling contiene un problema en la variante de asignación de tareas en donde hay m máquinas en serie y cada tarea debe de pasar por cada una de estas m máquinas. Se puede generalizar haciendo que haya varias máquinas en cada etapa (Pinedo 2010: 15). Es un tipo de problema diferente a esta situación de la mesa de ayuda, sin embargo puede ser que una mesa de ayuda esté envuelta en varios procedimientos de la gestión de incidentes como análisis del problema, equipos de técnicos para procedimientos de problemas urgentes, etc.

- Ramirez provee una solución al problema de Flow Shop Scheduling utilizando GRASP con doble relajación; es decir, utilizando dos alfas de relajación para la elección de máquinas y tareas (Ramirez 2006).
- Bard y Rios proveen un GRASP para la minimización del makespan en un Flow Shop de m máquinas (Bard y Rios 2000).
- Tupia utiliza GRASP contra un caso de máquinas diferentes y tareas independientes (Tupia, 2004).

3.5 Conclusiones sobre el estado del arte

Evaluando los productos comerciales presentados anteriormente, se puede ver que resultan útiles para el seguimiento de errores y su módulo de gestión de incidentes tiene automatizaciones. Se tendría que ver una comparación y simulación de cómo es el algoritmo de asignación contra el propuesto en esta tesis. Algunos servicios utilizan Round Robin como método de asignación. Sin embargo, esta asignación es para máquinas en paralelo (Pinedo 2010: 342) y no para una situación en donde las máquinas varían. Y como se ha optado, Megow, Uetz y Vredeveld optan por crear el algoritmo MinIncrease como una opción para las asignaciones estocásticas y en línea.

Según lo leído en las investigaciones, se obtienen soluciones para varias variantes de la asignación de tareas. Las respuestas ciertamente se complican si se pasa a un entorno estocástico y si se pasa a un entorno en línea.

El problema radica en la asignación en diferentes velocidades y con restricciones de elección de máquina; es decir, en la notación de Graham “Problema-Condiciones-Objetivo” sería $Rm \mid M_j, R_j \mid \Sigma w_j * C_j$ lo cual no es considerado en el algoritmo que se está basando esta tesis. Para resolverlo, se necesitaría que se consideren las máquinas que puedan resolver las tareas asignadas, así como facilitar la distribución de tareas de manera de que los mejores sean los únicos con tareas para resolver, así como asignar tareas de peso (urgentes) que sean resueltos lo más rápido posible.

Es por ello de que a dicho algoritmo se le asignará una metaheurística que permita la elección aleatoria de una lista restringida de candidatos a resolver el problema, a fin de distribuir las tareas de manera distribuida.

CAPÍTULO 2

1 Objetivo general

Aplicar el algoritmo GRASP al algoritmo MinIncrease GRASP para obtener una solución eficiente al problema de asignación de incidentes a una mesa de ayuda.

2 Objetivos específicos

1. Definir el algoritmo a usar que permita una asignación de tareas a máquinas en paralelo en un entorno estocástico y en línea.

2. Definir la función de mérito para la evaluación de técnicos e incidentes, utilizado para determinar la asignación entre una lista de candidatos para asignar la tarea a una máquina.

3. Definir los grados de relajación (constantes α) a utilizar en el algoritmo GRASP tanto para el ordenamiento de técnicos como el ordenamiento de incidentes.

4. Implementar las estructuras de datos de soporte para el algoritmo, los incidentes y los técnicos.

5. Desarrollar una experimentación numérica para demostrar la mejora con el algoritmo GRASP con respecto a un algoritmo voraz.

6. Construir el programa simulador de la solución.

3 Resultados esperados

- Resultado esperado para el objetivo 1: El algoritmo escogido para la asignación de los incidentes a técnicos y adaptado para el problema.

- Resultado esperado para el objetivo 2: La definición de la función de mérito para escoger al técnico a resolver el incidente.

- Resultado esperado para el objetivo 3: La definición de los grados de relajación a utilizar en el algoritmo GRASP.

- Resultado esperado para el objetivo 4: La arquitectura mostrando la construcción de las estructuras de soporte.

- Resultado esperado para el objetivo 5: El documento de experimentación y comparación numérica evidenciando la superioridad del algoritmo GRASP en la asignación de personal versus un algoritmo voraz.

- Resultado esperado para el objetivo 6: El programa prototipo de asignación de personal con los algoritmos implementados.

4 Herramientas, métodos y procedimientos

4.1 Mapeo

Resultados esperados	Herramientas a usarse
RE1: El algoritmo escogido para la asignación de los incidentes a técnicos y adaptado para el problema.	**Algoritmo MinIncrease:** será utilizado para establecer un método de evaluación de una tarea a las máquinas
RE2: La definición de la	**Algoritmo GRASP:** que especifica una

función de mérito para escoger al técnico a resolver el incidente.	función de mérito necesario para escoger el técnico de una lista restringida de candidatos.
RE3: La definición de los grados de relajación a utilizar en el algoritmo GRASP.	**Algoritmo GRASP:** que utiliza un grado de relajación correcto para hacer la selección lo suficientemente aleatorio.
RE4: La arquitectura mostrando la construcción de las estructuras de soporte.	**Pseudocódigo:** Utilizado para controlar el proceso de desarrollo del código. **Programación Orientada a Objetos - Java,** el lenguaje será utilizado para la construcción de código fuente de las estructuras.
RE5: El documento de experimentación y comparación numérica evidenciando la superioridad del algoritmo GRASP en la asignación de personal versus un algoritmo voraz.	**Prueba de Hipótesis:** Utilizada para sostener la hipótesis que un algoritmo es más fuerte que el otro. **Prueba de distribución normal:** Utilizada para evaluar si ambos algoritmos arrojan datos en una distribución normal. **Prueba varianzas homogéneas:** Utilizada para comparar varianzas si son o no homogéneas. **Prueba de comparación:** Utilizada para comparar los valores y validar la superioridad de un algoritmo. **SPSS:** Programa utilizado para el análisis estadístico.
RE6: El programa prototipo de asignación de personal con los algoritmos implementados.	**Programación Orientada a Objetos - Java,** el lenguaje será utilizado para la construcción de código fuente de todo el programa.

4.2 Herramientas

4.2.1 Algoritmo MinIncrease

Este algoritmo fue propuesto por Megow, Uetz y Vredeveld para la selección de un técnico a una tarea por medio de una evaluación matemática con respecto a los tiempos de procesamiento y el peso de las tareas.

Esta herramienta será utilizada para obtener una lista de cifras respecto de una tarea a todas las máquinas posibles, con la cual se obtendrá en orden las máquinas más ideales a asignar.

4.2.2 Algoritmo GRASP

El algoritmo GRASP, plantea un algoritmo voraz, aleatorio, adaptativo y de búsqueda. Se tiene una ventaja sobre los algoritmos voraces comunes, que en vez de seleccionar solamente el mejor valor de la función objetivo, se relaja el criterio de tal manera que se seleccione un conjunto de elementos que cumplen ciertas condiciones.

El algoritmo GRASP, tal como se ha explicado en el marco teórico, establecía que la fórmula de establecer la lista restringida de candidatos tiene un parámetro α, el cual interpreta si la búsqueda tiende más a ser aleatoria o ser voraz. El objetivo será ubicar

el parámetro α entre 0 y 1, el cual nos dé el resultado suficientemente aleatorio que necesitemos.

4.2.3 Programación Orientada a Objetos - Java

La programación orientada a objetos es un paradigma de programación que representa conceptos de datos como “objetos” que tienen procedimientos asociados llamados “métodos”. Estos objetos pueden ser agrupados en “clases” distintas. Este paradigma permite un orden modular del código de programación y puede ser fácilmente modelado.

Java es un lenguaje de programación orientada a objetos, desarrollado por James Gosing en Sun Microsystems, ahora parte de Oracle Corporation. Java tiene la ventaja de ser portable y compatible independiente del sistema operativo, ayudado por la Java Virtual Machine. El lenguaje es uno de los más populares en existencia.

El código fuente del proyecto será desarrollado utilizando este paradigma de programación con este lenguaje.

4.2.4 SPSS

SPSS es un programa especializado para el análisis estadístico creado por IBM Corporation. Su nombre significa “Paquete Estadístico para las Ciencias Sociales”. Esta herramienta nos servirá para facilitar los cálculos respectivos en la experimentación numérica.

4.3 Metodologías

4.3.1 Pseudocódigo

El pseudocódigo es un método de descripción de alto nivel de un código fuente para un programa o un algoritmo. Éste puede ser leído y adaptado a varios lenguajes de programación debido a que sigue una estructura similar estándar.

El pseudocódigo será utilizado como una manera preliminar para mostrar el flujo del código del algoritmo y las estructuras de datos.

4.3.2 Prueba de Hipótesis

Es el procedimiento para verificar si una propiedad en una población estadística es compatible con la muestra de la población. Se tiene una hipótesis determinada y una hipótesis alternativa; las cuales final del ejercicio se determinará cuál debe de ser rechazada.

Se utilizará para verificar los resultados de la experimentación numérica que certifiquen que el algoritmo apoyado es más rápido.

4.3.3 Prueba de distribución normal

Las pruebas de distribución normal nos permitirán saber si los datos que hemos generado a través de los dos algoritmos son de distribución normal o no.

Estas pruebas consistirán en:

- Prueba de asimetría y curtosis.
- Prueba de Kolmogorov-Smirnov.
- Prueba de Shapiro-Wilk.
- Prueba de histograma y normal esperada.

4.3.4 Prueba de varianzas homogéneas

La prueba de varianzas homogéneas nos permitirá saber si los datos generados tienen una varianza homogénea o no. Para este caso de estudio utilizaremos la prueba F de Fisher.

4.3.5 Prueba de comparación

La prueba Z permite comparar si dos o más medias muestrales pueden haberse obtenido de poblaciones con la misma media paramétrica respecto de una variable dada o son diferentes. Para esto, los datos requieren que sean de distribución normal y de varianza homogénea.

La prueba U de Mann-Whitney permite comparar las medias muestrales en caso de que la distribución no sea normal y la varianza no sea homogénea.

Un diseño adecuado permitirá que los datos obtenidos cumplan con las premisas paramétricas y permite la aplicación de pruebas estadísticas para la interpretación de resultados.

5 Alcance

El objetivo del proyecto es la asignación de tareas estocásticas en línea a máquinas con diferentes velocidades por medio de dos algoritmos. El tema de mesa de ayuda, usando asignación de técnicos a incidentes será el modelo del programa simulador.

El programa no proveerá apoyo a la solución de los incidentes en sí.

El programa tampoco hará reasignación de tareas o pre-procesamiento (“preemption” en asignación de tareas), ni considerará fechas de entrega o “release dates” cuando las tareas aparezcan, ya que serán asignadas de inmediato para resolverlas a la brevedad posible.

La comparación en la experimentación numérica será contra una posición completamente voraz en la asignación (el primero mejor en la lista de MinIncrease).

5.1 Limitaciones

El problema de asignación es un problema de tipo NP-Hard, por lo que no habrá una solución “exacta” o “definitiva”. Por la aleatoriedad del algoritmo GRASP y de las

circunstancias del problema, es posible que el resultado no sea el óptimo, pero será uno satisfactorio.

La optimización es más visible en casos de mesas de ayuda grandes con flujos de incidentes frecuentes y en una cantidad razonable. La optimización no se percibirá en su totalidad para sets de incidentes que sean pocos o mesas de ayuda de muy pocas personas.

5.2 Riesgos

Riesgo identificado	Impacto en el proyecto	Medidas correctivas para mitigar
Error en la fórmula de cuantificación de las tareas y máquinas en el algoritmo MinIncrease.	Mediano - Alto: Es posible que perjudique el desarrollo del proyecto si el error se encuentra en un estado avanzado del proyecto.	Se tiene que realizar pruebas de manera que el algoritmo esté correctamente implementado.
Error de formulación en la función de mérito en el algoritmo GRASP.	Mediano - Alto: Es posible que perjudique el desarrollo del proyecto si el error se encuentra en un estado avanzado del proyecto.	Se tienen que realizar formulaciones que estén alineados con la teoría de modelos de asignación, así como los del mismo algoritmo GRASP.
Error de selección de grados de relajación en el algoritmo GRASP	Alto: Es posible que perjudique el desarrollo del proyecto si el error se encuentra en un estado avanzado del proyecto, o conlleve al fracaso del proyecto si sucede en una presentación final.	La investigación y pruebas sobre la selección del grado de relajación deben de ser exhaustiva a fin de mitigar dicho riesgo.
Error en la generación de tareas para la prueba de asignación.	Mediano: Posibles cálculos no suficientes para la demostración del funcionamiento del algoritmo.	Realizar pruebas a fin de que la lista aleatoria de tareas sea la correcta.
El documento de experimentación numérica demuestra que el algoritmo voraz resulta más eficaz que el algoritmo GRASP con doble relajación.	Severo: Posible fracaso del proyecto.	Realizar pruebas a revisar si la diferencia es debido a errores de grados de relajación o formulación de la función de mérito.

6 Justificación y viabilidad

6.1 Justificativa del proyecto de tesis

Uno de los beneficios significativos es la asignación automática efectiva de recursos. De esta manera, se ahorra tiempo en deliberar quién es el más apropiado para

encargarse de la tarea. En el caso del ejemplo, se delibera quién es el técnico más apropiado para resolver el incidente, a la vez de que los más experimentados están libres para ser asignados donde su experiencia sea la más efectiva, así como estar atento a los diversos incidentes y que el tiempo de respuesta sea dentro de los parámetros establecidos en la prioridad y urgencia del incidente.

Para los problemas de asignación de tareas, con esta tesis se ofrece una solución al caso de máquinas en paralelo con diferentes velocidades y restricciones de procesamiento, lo cual el algoritmo MinIncrease no consideraba originalmente, apoyándose con un algoritmo que provea aleatoriedad suficiente.

En el ejemplo, el beneficio que trae el proyecto es una mejora a las mesas de ayuda o centros de servicio que restablezcan el valor adicional que otorgan los servicios, así como restablecer la continuidad de negocio si es que el incidente es crítico, otorgando una mayor resilencia a la empresa.

6.2 Análisis de viabilidad del proyecto de tesis

- Viabilidad técnica:

El proyecto se construirá a partir del lenguaje de programación Java, orientado a objetos. Esto es por la facilidad de manejo con los documentos de diseño y por el conocimiento que se tiene del lenguaje. El lenguaje se puede descargar libremente de internet, utilizando como interfaz de desarrollo Netbeans, el cual también se puede descargar libremente.

La base de datos que se utilizará para el soporte del sistema de información será MySQL. Esta base de datos puede ser descargado gratis, el cual puede instalar una base de datos localmente en la computadora. En caso de que se requiera una base de datos externa, se tendrá que consultar la viabilidad con el área de redes de la universidad, el cual podrá proveer de una base de datos, con su administrador y contraseña.

También se necesita estudiar a profundidad el marco y la teoría de asignación y algoritmo GRASP y MinIncrease. La teoría del algoritmo GRASP y MinIncrease debe de ser aprendida a conciencia para la formulación correcta de las funciones de mérito.

Para este proyecto se disponen de recursos bibliográficos como:

- Biblioteca de la Facultad de Ciencias e Ingeniería de la Pontificia Universidad Católica del Perú.
- Biblioteca de tesis en la Facultad de Ciencias e Ingeniería.
- Revistas de investigación online, como por ejemplo AICIT-AISS.

Además de contar con el apoyo teórico en las bibliografías, se puede contar con el apoyo del asesor de tesis y profesores con conocimiento en ciencias de la computación para asesorías y consulta de problemas.

- Viabilidad temporal:

Para este proyecto, se dispone de 13 semanas para realizar completamente el proyecto de fin de carrera, con entregables semanales.

Se podrían encontrar dificultades de tiempo con laborales profesionales, por lo que el avance estaría supeditado a las horas de la tarde y noche.

Para mayor detalle, revisar el diagrama de Gantt adjunto con el apéndice.

- Viabilidad económica:

El proyecto de fin de carrera es sin fines de lucro y el proyecto no necesita de recursos monetarios para su propósito.

- Análisis de necesidades:

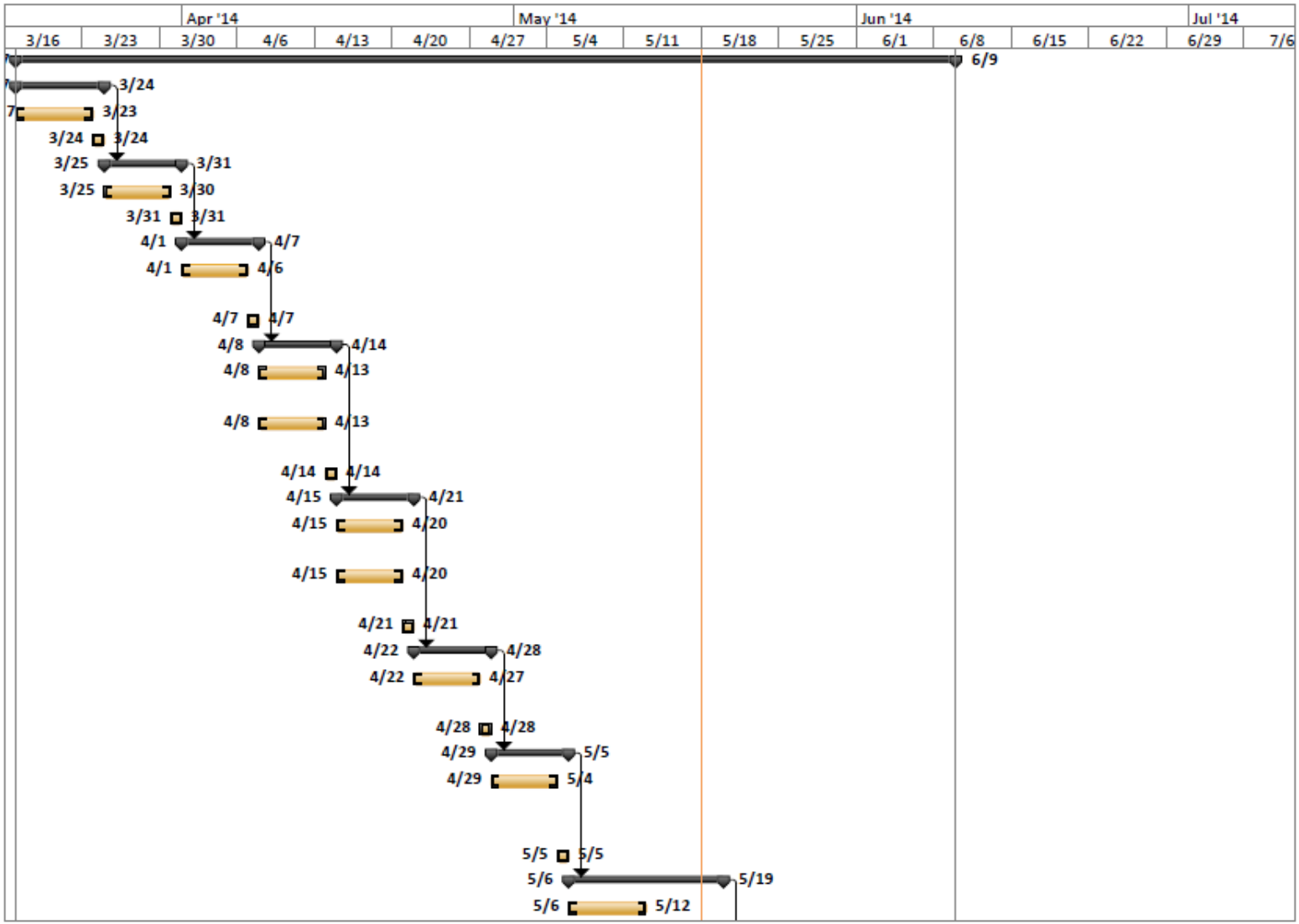
Necesidades	Comentario
Conocimiento	A disposición: • Biblioteca de Facultad de Ciencias e Ingeniería • Asesorías de profesores relevantes a Ciencias de la Computación • Acceso en línea a bases de datos de artículos científicos de AICIT o Mathematics of Operations Research.
Tiempo	Se realiza un avance cada siete días (revisar diagrama de Gantt).
Monetario	No es necesario en este proyecto, es un proyecto sin fines de lucro ni busca invertir en un producto.
Humano	No es necesario en este proyecto, es enteramente un proyecto de una sola persona. No se necesita manejo de personal ni establecer políticas de comunicación.

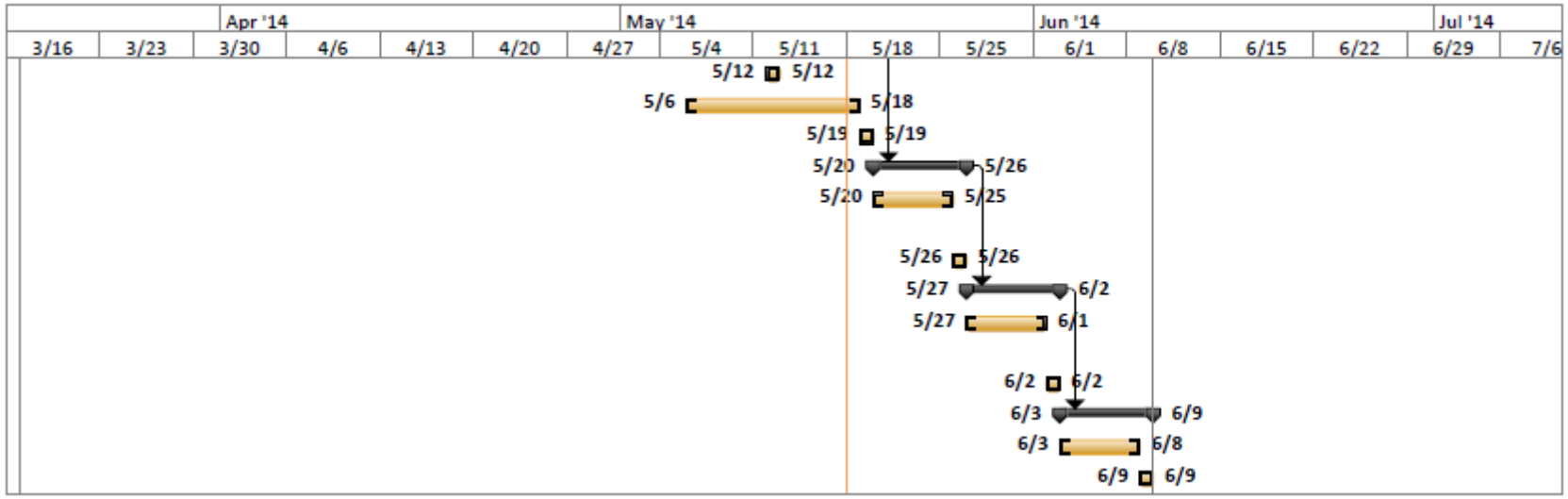
7 Plan de actividades

El plan de actividades se encontrará a continuación en un diagrama de Gantt.

ID	Task Name	Duration	Start	Finish
1	**Desarrollo del Curso de Proyecto de Tesis 2**	**61 days**	**Mon 3/17/14**	**Mon 6/9/14**
2	**Desarrollo Previo (Tesis 1)**	**6 days**	**Mon 3/17/14**	**Mon 3/24/14**
3	Ajustes al desarrollo previo. Revisión de contenido, gramática.	6 days	Mon 3/17/14	Sun 3/23/14
4	Exposición de desarrollo previo (Tesis 1)	1 day	Mon 3/24/14	Mon 3/24/14
5	**Adaptación del algoritmo MinIncrease al problema**	**5 days**	**Tue 3/25/14**	**Mon 3/31/14**
6	Investigación y pseudocódigo del algoritmo MinIncrease	5 days	Tue 3/25/14	Sun 3/30/14
7	Exposición del algoritmo MinIncrease adaptado	1 day	Mon 3/31/14	Mon 3/31/14
8	**Definición de la función de mérito para la selección de técnicos**	**5 days**	**Tue 4/1/14**	**Mon 4/7/14**
9	Investigación y pseudocódigo de la función de mérito para la evaluación de técnicos, utilizado para determinar al técnico asignado.	5 days	Tue 4/1/14	Sun 4/6/14
10	Exposición de la función de mérito	1 day	Mon 4/7/14	Mon 4/7/14
11	**Construcción del programa simulador (1)**	**5 days**	**Tue 4/8/14**	**Mon 4/14/14**
12	Construcción e implementación de la base de datos para el soporte del programa.	5 days	Tue 4/8/14	Sun 4/13/14
13	Construcción e implementación de los algoritmos usados para la selección de técnicos.	5 days	Tue 4/8/14	Sun 4/13/14
14	Exposición del programa simulador	1 day	Mon 4/14/14	Mon 4/14/14
15	**Construcción del programa simulador (2)**	**5 days**	**Tue 4/15/14**	**Mon 4/21/14**
16	Implementación de las estructuras de datos de los técnicos, incidentes y estructuras auxiliares.	5 days	Tue 4/15/14	Sun 4/20/14
17	Implementación de comunicación entre base de datos y los algoritmos para la asignación de incidentes.	5 days	Tue 4/15/14	Sun 4/20/14
18	Exposición del programa simulador	1 day	Mon 4/21/14	Mon 4/21/14
19	**Implementación de registros de resumen y log**	**5 days**	**Tue 4/22/14**	**Mon 4/28/14**
20	Implementar la impresión de registros de resumen y log de la asignación de los técnicos.	5 days	Tue 4/22/14	Sun 4/27/14
21	Exposición de los registros	1 day	Mon 4/28/14	Mon 4/28/14
22	**Definición de los grados de relajación**	**5 days**	**Tue 4/29/14**	**Mon 5/5/14**
23	Definir los grados de relajación (constantes α) a utilizar en el algoritmo GRASP tanto para el ordenamiento de técnicos como el ordenamiento de incidentes.	5 days	Tue 4/29/14	Sun 5/4/14
24	Exposición de los grados de relajación	1 day	Mon 5/5/14	Mon 5/5/14
25	**Exposición Parcial**	**10 days**	**Tue 5/6/14**	**Mon 5/19/14**
26	Redacción del documento parcial de Tesis 2	5 days	Tue 5/6/14	Mon 5/12/14

ID	Task Name	Duration	Start	Finish
27	Entrega del documento parcial	1 day	Mon 5/12/14	Mon 5/12/14
28	Ajustes al código fuente para la presentación	10 days	Tue 5/6/14	Sun 5/18/14
29	Exposición Parcial	1 day	Mon 5/19/14	Mon 5/19/14
30	**Experimentacion Numérica**	**5 days**	**Tue 5/20/14**	**Mon 5/26/14**
31	Desarrollar una experimentación numérica para demostrar la superioridad del algoritmo GRASP con doble relajación con respecto a un algoritmo	5 days	Tue 5/20/14	Sun 5/25/14
32	Exposición de Experimentación Numérica	1 day	Mon 5/26/14	Mon 5/26/14
33	**Carga de Datos**	**5 days**	**Tue 5/27/14**	**Mon 6/2/14**
34	Preparar un método para la carga de datos a partir de una lista de incidentes para realizar la simulación.	5 days	Tue 5/27/14	Sun 6/1/14
35	Exposición de Carga de Datos	1 day	Mon 6/2/14	Mon 6/2/14
36	**Entrega Final**	**5 days**	**Tue 6/3/14**	**Mon 6/9/14**
37	Ajustes de redacción al documento final.	5 days	Tue 6/3/14	Sun 6/8/14
38	Entrega Final	1 day	Mon 6/9/14	Mon 6/9/14





CAPÍTULO 3

1 Estructuras de Datos

En este apartado se verán las estructuras de datos a utilizar en la asignación de incidentes a técnicos en una mesa de ayuda. Para ello, definiremos cuáles son las estructuras que pertenecen a los incidentes y a los técnicos, así como estructuras auxiliares para el funcionamiento del programa.

1.1 Incidente

1.1.1 Definición

Un incidente es la representación de este problema de una tarea *j* que tiene que ser asignada a una máquina *m*. Este incidente tiene que ser asignado y eventualmente resuelto por la máquina escogida o mejor dicho, en el caso de este problema, a un técnico.

Estructura básica del incidente:

- Identificación – IDI
- Tipo – TINC
- Peso – W
- Tiempos de Procesamiento– TIME
- Técnico Asignado – ASIG
- Tiempo de Procesamiento Real – TPR

Figura 3.1.1 – Estructura básica de un incidente

- Identificación (IDI) es una serie de números, letras o símbolos de identificación al incidente. Permitirá el nombramiento del incidente para uso de bases de datos y/o reportes.
- Tipo (TINC) es un número que identificará el tipo de incidente que es según la política de catálogo de incidentes que se quiera modelar (leve, moderado, urgente, etc.). Se utilizará para la identificación y restricción del incidente para saber qué técnicos pueden resolver o están autorizados de resolver dicho problema.
- Peso (W) denota la importancia de dicho incidente (*w* en la notación de asignación de tareas). Este sirve para la priorización de tareas por medio de WSEPT.
- Tiempos de Procesamiento (TIME) es una lista que incluye los tiempos de procesamiento esperados para cada uno de los tipos de técnicos.
- Técnico Asignado (ASIG) es una identificación del mismo tipo de dato del incidente que se mostrará después.
- Tiempo de Procesamiento Real (TPR) es el tiempo real que el incidente ha tomado en ser resuelta en un tiempo t.

1.1.2 Representación en Pseudocódigo

La representación en código, el cual se aprovechará que el programa sea orientado a objetos, será el siguiente:

```
Clase Técnico {
        Entero IDI
        Entero TINC
        Entero W
        Entero[] TIME
        Entero ASIG
        Entero TPR
}
```

Figura 3.1.2 – Representación en pseudocódigo de un incidente

Se utilizarán enteros para la IDI y TINC para la facilidad de construcción e identificación de los tipos. Similarmente, para el peso (W), la lista tiempos de procesamiento TIME, y el tiempo de procesamiento real (TPR) utilizaremos enteros para representar las unidades de tiempo que vayan a pasar. ASIG será entero al igual que la identificación de los técnicos (se describirá más adelante).

1.2 **Técnico**

1.2.1 Definición

Un técnico es la representación en este particular problema de una máquina *m* en un problema de asignación. El trabajo del técnico será de resolver los problemas que se le haya sido asignado. Estos técnicos tendrán la siguiente estructura:

Estructura básica del técnico:

- Identificación – IDT
- Tipo – TTEC
- Tarea en Proceso – TP
- Tareas en Espera – TE
- Tareas Cumplidas – TC

Figura 3.1.3 – Estructura básica de un técnico

- Identificación (IDT) es una serie de números, letras o símbolos de identificación al técnico. Permitirá el nombramiento del técnico para uso de bases de datos y/o reportes.
- Tipo (TTEC) es un número de identificación que permitirá saber qué tipo de técnico es. Como se menciono en capítulos anteriores, es necesario de que se sepa qué tipo de técnico es (practicante, junior, sénior, jefe, etc.) para establecer las limitaciones de procesamiento de incidentes que el problema requiera. Por ejemplo, podemos establecer que los practicantes sólo pueden resolver incidentes de impacto leve o de fácil resolución.
- Tarea en Proceso (TP) indica cuál es la tarea que se está resolviendo en un momento *t* determinado. Esta tarea será resuelta en un momento aleatorio determinado por el tiempo de procesamiento esperado del incidente.
- Tareas en Espera (TE) es un listado de tareas las cuales han sido asignadas a este técnico, y faltan ser resueltas. Estas tareas en espera estarán ordenados por WSEPT, priorizando que la nueva tarea a procesar tenga un peso y tiempo de procesamiento esperado aceptables. Dicho ordenamiento ocurrirá cada vez

que se produzca una nueva asignación (inserción de la tarea *j* a TE) a este técnico.

- Tareas Cumplidas (TC) es un listado de tareas las cuales han sido asignado a este técnico, han terminado de procesar y se encuentran en el registro como tareas resueltas.

1.2.2 Representación en Pseudocódigo

La representación en código, el cual se aprovechará que el programa sea orientado a objetos, será el siguiente:

```
Clase Técnico {
        Entero IDT
        Entero TTEC
        Incidente TP
        Incidente[] TE
        Incidente[] TC
}
```

Figura 3.1.4 – Representación en pseudocódigo de un técnico

Similar a los incidentes, se utilizarán enteros para IDT y TTEC para la facilidad de la construcción e identificación del técnico. TP tendrá como tipo de dato a un incidente descrito anteriormente. TE y TC tendrán como tipo de dato una lista de incidentes.

Un técnico que no tenga tareas en TP en un tiempo *t*, asignará si es factible, en TP al primero de la lista en TE, el cual será removida de dicha lista.

Un técnico que en un tiempo *t* haya completado el incidente TP, será removido de TP y pasará a unirse al listado TC.

1.3 **Lista Técnico**

Se tendrá almacenado una lista de técnicos en el programa, los cuales serán generados de acuerdo a la estructura mencionada arriba utilizando parámetros por medio de base de datos, e inmediatamente juntados en la lista de técnicos, la cual será un arreglo dinámico del tipo técnico.

Ejemplo Lista de técnicos:

TEC 0: Junior	TEC 1: Junior	TEC 2: Técnico	TEC 3: Técnico	TEC 4: Técnico	TEC 5: Técnico	TEC 6: Sénior	TEC 7: Sénior

Figura 3.1.5 – Imagen ejemplo de la lista de técnicos

1.4 **Lista Incidentes**

Se tendrá almacenado una lista de incidentes en el programa, los cuales serán generados de dos maneras:

- Generación de datos aleatoria. Por medio de datos de entrada, se generan un nuevo juego de datos en cada ejecución.
- Juego de datos existente: se carga un juego de datos existente y se procede generar incidentes, almacenándolos en la lista de incidentes.

Este arreglo será de tipo dinámico.

Ejemplo Lista de incidentes:

INC 0: Leve	INC 1: Moderado	INC 2: Moderado	INC 3: Leve	INC 4: Severo	INC 5: Leve	INC 6: Trivial	INC 7: Urgente

Figura 3.1.6 – Imagen ejemplo de la lista de incidentes

1.5 Matriz de Tiempos Esperados

Se tendrá almacenado una matriz de tiempos en el programa, los cuales serán generados por medio de la carga de la base de datos que se explicará más adelante. En esta matriz dinámica estarán guardados los tiempos esperados divididos por el tipo de técnico.

Los índices de la matriz serán los tipos de técnicos. Este apuntará a guardar un arreglo que contenga la lista de tiempos esperados para ese tipo de técnico.

		Tipo de Incidentes		
Tipo Técnico		Leve	Moderado	Severo
Junior	→	3	4	N/A
Técnico	→	2	3	6
Sénior	→	1	3	4

Figura 3.1.7 – Imagen ejemplo de la matriz de tiempos esperados

1.6 Matriz de Ocurrencias

La matriz de ocurrencias tiene como fin el apoyo a la generación de data aleatoria en cada ejecución. Estos datos servirán para cargar la tabla de la base de datos al programa y tener el apoyo para la generación de datos. Con ello, podremos crear con la estructura "incidente" y llenar la estructura de lista de incidentes.

Tiene como características la frecuencia de que ocurran esos incidentes en un tiempo *t*, la probabilidad de que este incidente se materialice y registre, y su peso o severidad.

Tipo Incidente		Frecuencia	Probabilidad	Peso
Leve	→	5	90	1
Moderado	→	4	50	4
Severo	→	2	30	7

Figura 3.1.8 – Imagen ejemplo de la matriz de ocurrencias

2 Base de Datos

En este apartado veremos las tablas de la base de datos que se van a utilizar en este trabajo, los cuales servirán de apoyo como carga de datos y parámetros iniciales.

2.1 Tabla de Cantidad de Técnicos

Esta tabla apoya a la generación de la lista de técnicos mencionada anteriormente. Esta lista provee el número de técnicos a crear por cada tipo que se desee.

CANTIDAD_TECNICO	
ID_TIPO_TECNICO	ENTERO
TIPO_TECNICO	CADENA
CANTIDAD	ENTERO

Figura 3.2.1 – Tabla de Cantidad de Técnicos

2.2 Tabla de Ocurrencias de Incidentes

Esta tabla posee las probabilidades, frecuencias y peso para la matriz de ocurrencias mencionadas anteriormente. Este poseerá el tipo de incidente, la frecuencia de aparición en un momento *t*, la probabilidad de que se materialice y registre, y el peso o importancia de los incidentes. Como dijimos anteriormente, esta tabla se utilizará en caso de que se desee generar una lista de incidentes aleatoria.

OCURENCIA_INCIDENTE	
ID_TIPO_INCIDENTE	ENTERO
TIPO_INCIDENTE	CADENA
FRECUENCIA	ENTERO
PROBABILIDAD	ENTERO
PESO	ENTERO

Figura 3.2.2 – Tabla de ocurrencia de incidentes

2.3 **Tabla Tiempos Esperados**

La tabla de tiempos esperados tiene como uso el llenado de la matriz de tiempos esperados visto anteriormente. Esta tabla tiene un número de columnas variadas además del tipo de técnico. Se tendrán N columnas en donde N será el número de tipos de incidentes a establecer en el problema. Esta tabla será generada cada vez que se modifique las tablas anteriores de Técnico e Incidentes.

TIEMPO ESPERADO	
TIPO_TECNICO	ENTERO
TIPO_INCIDENTE_1	ENTERO
TIPO_INCIDENTE_2	ENTERO
…	
TIPO_INCIDENTE_N	ENTERO

Figura 3.2.3 – Tabla de tiempos esperados

2.4 **Tabla de Incidentes**

Esta tabla tiene como fin tener un registro de los incidentes, bien sean generados o insertados de una batería de incidentes a simular. Contiene datos de la identificación del incidente, el tipo de incidente, el peso o importancia del incidente, el momento *t* de la aparición y la identificación del técnico que haya sido asignado dicho incidente.

LISTA_INCIDENTES	
ID_INCIDENTE	ENTERO
ID_TIPO_INCIDENTE	ENTERO
PESO	ENTERO
APARICION	ENTERO
ID_TECNICO_ASIGNADO	ENTERO

Figura 3.2.4 – Tabla de incidentes

2.5 **Tabla de Carga de Incidentes**

Esta tabla tiene como fin tener un registro de los incidentes que hayan sido cargados al problema (lo cual, por lo tanto, no usará generación de datos aleatorios). Dicha carga de incidentes tiene una estructura similar a la tabla de incidentes, con la excepción de que no hay un campo de asignación de los técnicos puesto que no se ha asignado ningún incidente al inicio del problema.

CARGA_INCIDENTES	
ID_INCIDENTE	ENTERO
ID_TIPO_INCIDENTE	ENTERO
PESO	ENTERO
APARICION	ENTERO

Figura 3.2.5 – Tabla de carga de incidentes

CAPÍTULO 4

1 Adaptación del Algoritmo MinIncrease al problema:

Como se estipuló en el capítulo 1, el algoritmo MinIncrease arroja un número comparando un trabajo con todas las máquinas. El algoritmo será responsable de entregar una lista de números al algoritmo GRASP, el cual será responsable de seleccionar un técnico.

El algoritmo MinIncrease en sí se ejecutará para todas las máquinas que puedan ejecutar el problema, es decir, primero se asegurará de que la máquina puede resolver el problema. Caso contrario, no incluye a la máquina en dicho cálculo.

Para ello, primero tenemos una función llamada ListaMinIncrease que toma como datos de ingreso el trabajo j y la lista de técnicos, y devolverá el cálculo hecho a cada una de las máquinas posibles.

1.1 ListaMinIncrease

```
Función ListaMinIncrease(Arreglo Incidente, Arreglo ListaTecnicos):
1.      Mapa  ListaRatiosMI <- []
2.      Para i <- 1 Hasta ListaTecnicos.Tamaño Hacer:
2.1.            Si Trabajo.Tiempo(ListaTecnicos[i].tipo) != 0 Entonces:
2.1.1
.               ListaRatiosMI.Agregar(ListaTecnicos[i].ID,CalculoMinIncrease(Trabajo
        , ListaTecnicos[i]))
            Fin Si
        Fin Para
3.      Retornar ListaRatiosMI
Fin Función
```

Figura 4.1 – Pseudocódigo de la función ListaMinIncrease

Explicación:

- Parámetros de entrada: un incidente, la lista de técnicos.
- Parámetros de salida: mapa de tuplas [ID de técnico, números de unidades MinIncrease].

1. Creación de la lista de retorno de cálculos Minincrease
2. Bucle: desde 1 hasta el tamaño de la lista de técnicos

2.1 Condición: Se averigua el tiempo esperado del incidente para el técnico en la posición i de la lista de técnicos (i siendo controlado por el bucle). En caso de que no sea 0 (es decir, que esté autorizado para resolver el incidente), proseguir.

2.1.1 Se procede a llamar a la función CalculoMinIncrease para obtener el ratio de MinIncrease respecto al incidente con dicho técnico en la posición i. Una vez obtenido el ratio, se agrega a la lista de retorno de cálculos de ratio junto con el ID del técnico.

3. Retorna el mapa de tuplas de ratios Minincrease.

1.2 **CalculoMinIncrease**

```
Función CalculoMinIncrease(Arreglo Trabajo, Arreglo Tecnico):
1.      Real  ValorExpresion <- 0
2.      Real SumatoriaAltaProcesamiento <- 0
3.      Real SumatoriaBajaPeso <- 0
4.      Real RatioComparacion <- Trabajo.Peso/Trabajo.Tiempo(Tecnico.Tipo)
        //PRIMERA SUMATORIA
5.      Para i <- 1 Hasta Tecnico.ListaTrabajosAProcesar.Tamaño Hacer:
5.1.            Si RatioComparacion <=
        Tecnico.ListaTrabajosAProcesar[i].Peso/Tecnico.ListaTrabajosAProcesar[i].Tie
        mpo(Tecnico.Tipo) Entonces:
5.1.1                   SumatoriaAltaProcesamiento = SumatoriaAltaProcesamiento +
        Tecnico.ListaTrabajosAProcesar[i].Tiempo(Tecnico.Tipo)
            Fin Si
        Fin Para
        //SEGUNDA SUMATORIA
6.      Para i <- 1 Hasta Tecnico.ListaTrabajosAProcesar.Tamaño Hacer:
6.1.            Si RatioComparacion > Tecnico.ListaTrabajosAProcesar[i].Peso/
        Tecnico.ListaTrabajosAProcesar[i].Tiempo(Tecnico.Tipo) Entonces:
6.1.1                   SumatoriaBajaPeso = SumatoriaBajaPeso +
.       Tecnico.ListaTrabajosAProcesar[i].Peso
            Fin Si
        Fin Para
        //JUNTAR
7.      ValorExpresion <- Trabajo.Peso * SumatoriaAltaProcesamiento +
        Trabajo.Tiempo(Tecnico.Tipo) * SumatoriaBajaPeso +
        Trabajo.Peso*Trabajo.Tiempo(Tecnico.Tipo)
8.      Retornar ValorExpresion
Fin Función
```

Figura 4.2 – Pseudocódigo del algoritmo MinIncrease

Explicación:

- Parámetros de entrada: un incidente, un técnico
- Parámetros de salida: un número real con el valor del ratio MinIncrease para ese incidente y ese técnico

1. Declaración de la variable del ratio MinIncrease a retornar.
2. Declaración de la variable de la sumatoria de tiempo de procesamiento de los incidentes previamente asignados a este técnico cuyo WSEPT sea mayor o igual al WSEPT del incidente a asignar.
3. Declaración de la variable de la sumatoria de pesos de los incidentes previamente asignados a este técnico cuyo WSEPT sea menor al WSEPT del incidente a asignar.
4. Declaración y asignación del ratio de comparación contra los incidentes previamente asignados (el WSEPT del incidente a asignar).
5. Bucle de la primera sumatoria: desde 1 hasta el tamaño de la lista de incidentes a resolver asignados al técnico.

5.1 Condicional: Se comparan los WSEPT de un incidente i en la lista de tareas por resolver contra el WSEPT del incidente declarado en el punto 4. En caso de que sea mayor o igual, procede.

5.1.1 Agregar el tiempo de procesamiento a la variable declarada en el punto 2.

6. Bucle de la segunda sumatoria: desde 1 hasta el tamaño de la lista de incidentes a resolver asignados al técnico.

6.1 Condicional: Se comparan los WSEPT de un incidente i en la lista de tareas por resolver contra el WSEPT del incidente declarado en el punto 4. En caso de que sea menor, procede.

6.1.1 Agregar el peso a la variable declarada en el punto 3.

7. El valor del algoritmo MinIncrease con ese problema será asignado por la sumatoria de tres partes: Multiplicar el peso del incidente contra la variable vista en el punto 2, multiplicar el tiempo de procesamiento con la variable vista en el punto 3, multiplicar peso e incidente del problema.

8. Regresa el valor de MinIncrease.

CAPÍTULO 5

1. Adaptación del algoritmo GRASP al problema:

El algoritmo GRASP provee de una selección aleatoria y golosa en vez de solamente golosa en la selección del mejor técnico respecto a la lista MinIncrease.

```
Función GRASP (Mapa ListaRatiosMI, Real Alfa):
1.      Arreglo RCL <- []
2.      OrdenarMenorAMayor(ListaRatiosMI)
3.      Real Beta <- ListaRatiosMI.Primero //MEJOR VALOR
4.      Real Tau <- ListaRatiosMI.Ultimo //PEOR VALOR
5.      Real LimiteRCL <- Beta – Alfa * (Beta – Tau)
6.      Para i = 1 Hasta ListaRatiosMI.Tamaño Hacer:
6.1.        Si LimiteRCL <= ListaRatiosMI[i] Entonces:
6.1.1.          RCL.Agregar(ListaRatiosMI[i])
            Fin Si
        Fin Para
7.      S  <- SeleccionAleatoria(RCL)
8.      Retornar S
Fin Función
```

Figura 5.1 – Pseudocódigo del algoritmo GRASP

Explicación:

- Entrada: Mapa de Tuplas de Ratios MinIncrease, Alfa (α) de relajación.
- Salida: ID del técnico seleccionado.

1. Se crea un arreglo de Lista Restringida de Candidatos o RCL.
2. Se ordena de menor a mayor el mapa de ratios MinIncrease respecto a sus valores dados por el algoritmo MinIncrease.
3. Se obtiene un real Beta (β), que será el primero del mapa con el ratio MinIncrease más bajo, o sea el mejor valor posible.
4. Se obtiene un real Tau (τ), que será el último del mapa con el ratio MinIncrease más alto, o sea el peor valor posible.
5. Se crea el límite para que un dato ingrese al RCL, el cuál será por medio del cálculo de GRASP mencionado en capítulos anteriores. La fórmula es: β – α * (β - τ).
6. Bucle: se recorre desde 1 hasta el tamaño del mapa de ratios.

6.1 Condición: si el valor de MinIncrease del mapa en la posición *i* es menor o igual que el límite de ingreso al RCL definido en el punto 5, procede Agregar el tiempo de procesamiento a la variable declarada en el punto 2.

6.1.1 Agrega el ID del técnico en la lista RCL si pasó la condición previa.

7. Se escoge aleatoriamente un ID de técnico de la lista RCL. Éste será el técnico a asignar en el incidente.
8. Retornar el valor del técnico.

Luego se tiene una fase de mejora al final de la ejecución en donde se ejecuta repetidas veces la simulación entera dependiendo de un número de iteraciones que se especificará en el capítulo siguiente. Se intercambiará los resultados de las eficiencias totales para que siempre se tenga el mejor resultado posible; es decir, la mejor simulación posible.

Para ello, especificamos “eficiencia total” con la siguiente operación:

$$Eficiencia\ Total = \frac{\sum_{i\to 1, i\ \in Cumplidos}^{n} w_i * E[CA_i]}{\sum_{i\to 1, i\ \notin Cumplidos}^{n} w_i * E[CA_i]}$$

Figura 5.2 – Fórmula de la Eficiencia Total de una simulación

Debido a que el objetivo del ejercicio de asignación (léase la parte γ de la notación de Graham en el capítulo 1) es el tiempo esperado de cumplimiento con pesos, podemos dividir las tareas en la simulación entre cumplidas y no cumplidas al final de la ejecución. Por ello, aplicamos una división de:

- La sumatoria de los tiempos cumplidos esperados (dependiendo del técnico asignado en cada incidente) multiplicado por su respectivo peso para todas las tareas *que hayan sido cumplidas*.
- La sumatoria de los tiempos cumplidos esperados (dependiendo del técnico asignado en cada incidente) multiplicado por su respectivo peso para todas las tareas *que **no** hayan sido cumplidas*.

Luego de esto, podemos simular varias veces la ejecución y obtener un mejor resultado. Para ello, la función de GRASP mejora es el siguiente:

Función GRASP Mejora (Real MejorEficiencia, Real Eficiencia):
1. Si MejorEficiencia < Eficiencia Entonces:
1.1. Real Eficiencia = Mejor Eficiencia
Fin Función

Figura 5.3 – Pseudocódigo de la mejora de resultado GRASP

Explicación:

- Entrada: Valor del mejor ratio de eficiencia posible en todas las iteraciones hasta el momento y el ratio de eficiencia obtenido en una iteración.
- Salida: Ninguna.

1. Si la eficiencia de la iteración es mayor que el mejor eficiencia, entra a 1.1.
1.1. Cambia el valor de la mejor eficiencia a de la iteración actual.

2. WSEPT para el reordenamiento de incidentes asignados.

El Tiempo Esperado de Procesamiento con Peso Más Corto o *Weighted Shortest Expected Processing Time*, como se mencionó en el capítulo 1, es una forma de selección en situaciones de scheduling estocástico, principalmente para problemas de una máquina. Sin embargo, aquí lo necesitamos para un ordenamiento importante: una vez que se hayan asignado un incidente a un técnico, cualquier trabajo asignado que esté en espera debe de ser reordenado de acuerdo a WSEPT para priorizar los trabajos que nos den mayor "puntaje": los más importantes o urgentes.

$$WSEPT = \frac{w_i}{E[C_i]}$$

Figura 5.4 – Fórmula del WSEPT para el reordenamiento.

Con este reordenamiento, podemos tener cada incidente en espera o *"backlog"* del técnico debidamente ordenado por urgencia y también por cuan provechoso es utilizar dicho tiempo en resolver el incidente de tal nivel, ya que juega parte en la división.

Función WSEPT (Tecnico tecnico):
1. Si técnico.ListaDeEspera.tamaño > 1 Entonces:
1.1 ReoredenarWSEPT(técnico.ListaDeEspera)
Fin Si
Fin Función

Figura 5.5 – Pseudocódigo de la función WSEPT

Explicación:

- Entrada: Técnico seleccionado por GRASP al que se insertó el incidente nuevo.
- Salida: Ninguno.

1. Condición: si el tamaño de la lista de espera del técnico es mayor a 1, entonces procede (NOTA: Previamente ya se insertó el incidente en cuestión a la lista de espera del técnico).

1.1 Reordenar por mayor a menor la lista de espera del técnico por medio del valor entregado por la fórmula del WSEPT.

CAPÍTULO 6

1. Cálculo del alfa del GRASP óptimo

Para el cálculo de GRASP, se utilizó una repetición de 32000 veces para analizar los mejores valores, peores valores, promedio de los valores retornados y su desviación estándar. Para ello, se utilizó una batería de ejercicios idénticos y condiciones idénticas para las pruebas.

En la primera tabla a continuación, se hizo los cálculos entre 0 y 1, sumando 0.1 cada vez para tener una idea de donde estimar la posición de alfa general.

Alfa	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Mejor	1.063	1.093	1.212	1.548	1.597	1.628	1.607	1.509	1.529	1.437	1.427
Promedio	0.814	0.672	0.682	0.678	0.728	0.702	0.717	0.801	0.818	0.805	0.786
Desviación Estándar	0.097	0.118	0.126	0.140	0.173	0.146	0.159	0.161	0.153	0.147	0.140

Figura 6.1 – Tabla de cálculo del valor del coeficiente alfa #1

Se puede observar que 0.5 nos arrojó un promedio alto con respecto a los vecinos. Con esto, tenemos una idea general de donde se puede encontrar el alfa ideal. Realizaremos una siguiente búsqueda para el centesimal entre 0.45 y 0.55.

Alfa	0.45	0.46	0.47	0.48	0.49	0.5	0.51	0.52	0.53	0.54	0.55
Mejor	1.660	1.633	1.592	1.583	1.586	1.687	1.586	1.581	1.691	1.534	1.572
Promedio	0.718	0.719	0.715	0.714	0.720	0.703	0.703	0.713	0.713	0.712	0.718
Desviación Estándar	0.165	0.167	0.165	0.166	0.168	0.146	0.144	0.146	0.145	0.146	0.153

Figura 6.2 – Tabla de cálculo del valor del coeficiente alfa #2

En este caso, el programa arrojó un promedio más alto y un mejor valor en 0.53. Por lo tanto, se le escogerá como el alfa a participar en el proyecto.

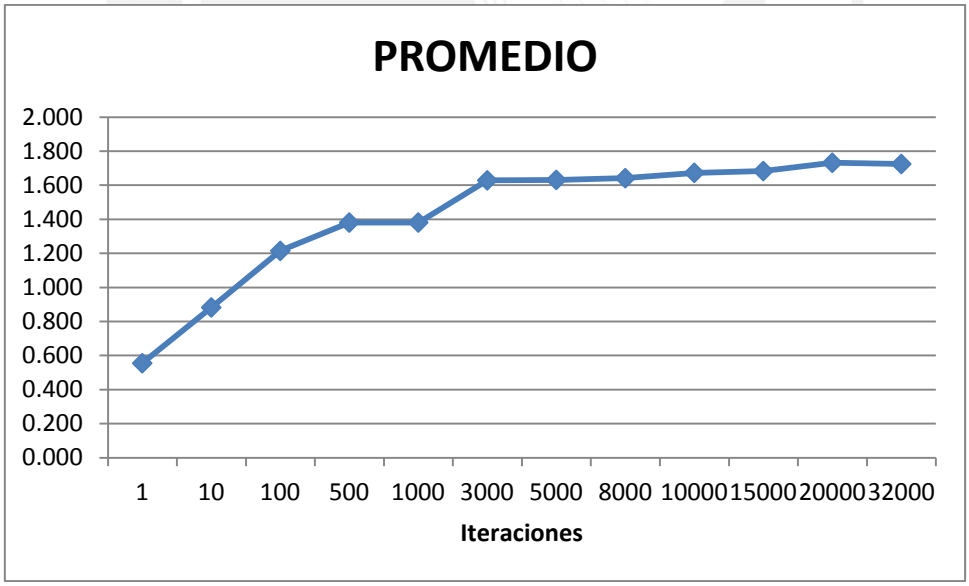
2. Cálculo de repeticiones óptimas para GRASP

El siguiente procedimiento es averiguar una cantidad de repeticiones que sea asequible para tener tanto un promedio satisfactorio como un tiempo de ejecución razonable.

	PROMEDIO	TIEMPO
1	0.555	0.06
10	0.882	0.41
100	1.214	2.33
500	1.381	6.81
1000	1.381	15.03
3000	1.629	46.19
5000	1.631	76.88
8000	1.641	123.47
10000	1.673	146.48
15000	1.683	230.70
20000	1.732	299.34
32000	1.725	354.94

Figura 6.3 – Tabla de ejecuciones del promedio y el tiempo de ejecución

Vemos que a 3000 ejecuciones, se tiene un resultado bueno que se encuentra por encima del promedio (1.427), y se tiene un tiempo de ejecución que puede ser aceptable en muchos casos. En simulaciones más grandes, es probable de que se incremente el tiempo de ejecución dependiendo del número de técnicos, número de incidentes a asignar y el tiempo *t* de la ejecución.



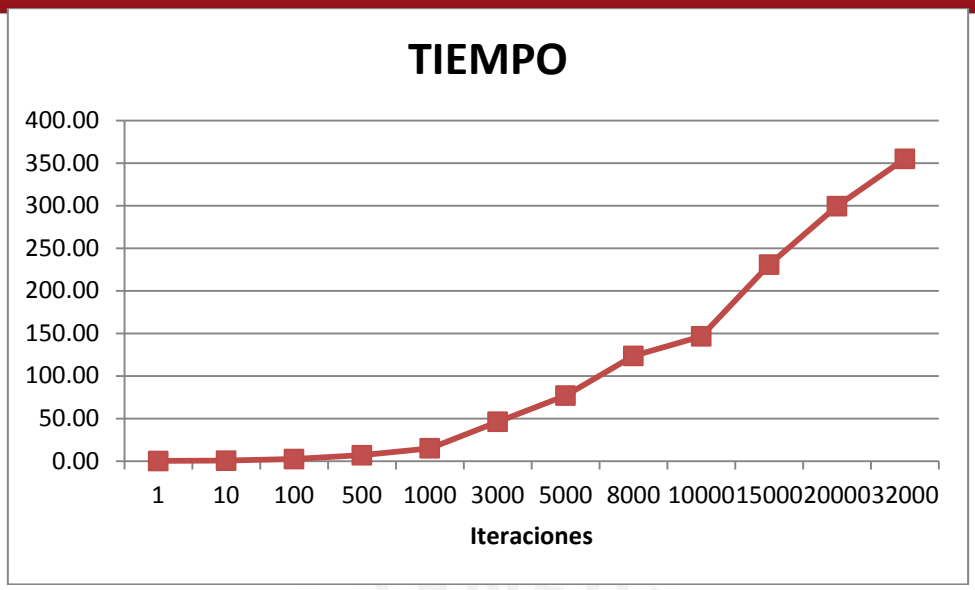


Figura 6.4 – Gráficos de las ejecuciones del promedio y el tiempo de ejecución

CAPÍTULO 7

1. Experimentación numérica

1.1 Introducción

En este apartado comprobaremos por cómo el algoritmo MinIncrease es mejorado por el algoritmo GRASP por medio de la experimentación númerica.

Para ello, estaremos utilizando el algoritmo MinIncrease sólo. Recordemos que el algoritmo MinIncrease originalmente toma el mejor resultado posible para la asignación (la asignación que reduzca lo mejor posible el valor de la fórmula antes mostrada). Por lo tanto, se le puede considerar que la elección del técnico es "el primero mejor" o "algoritmo voraz". Por medio del algoritmo GRASP estamos introduciendo un grado de aleatoriedad que permitirá cumplir con los requisitos de hacer un mejor uso de los empleados, haciendo que todos tengan menos tiempos muertos, que todos los empleados laboren en la medida de lo posible y que los mejores técnicos les sea evitado trabajos triviales.

1.2 Objetivo

El objetivo de esta experimentación es la comparación del algoritmo MinIncrease con MinIncrease + GRASP en la asignación de tareas, evidenciando la superioridad de este último.

1.3 Diseño

La unidad de comparación será lo que llamaremos la eficiencia total de la ejecución. Este será dado por la siguiente fórmula ya mencionada anteriormente:

$$Eficiencia\ Total = \frac{\sum_{i\to 1, i\ \in Cumplidos}^{n} w_i * C_i}{\sum_{i\to 1, i\ \notin Cumplidos}^{n} w_i * C_i}$$

La eficiencia total es la división entre las sumatorias de los n incidentes cumplidos en un período de tiempo y los n incidentes no cumplidos, los cuales tienen un valor de la multiplicación entre el peso del incidente y el tiempo esperado de cumplimiento (que dependerá del técnico a quien se le haya asignado).

Para esto, las condiciones de la simulación han sido las siguientes:

- Se ha ejecutado la simulación en un tiempo t de 480. Esto será para representar una jornada de trabajo promedio en un día (8 horas).
- La tabla de la cantidad de técnicos fue de la siguiente manera:

Tabla Cantidad Técnicos	
Tipo Técnico	Cantidad
1	5
2	8
3	2

Figura 7.1 – Tabla de cantidad de técnicos para la experimentación numérica

- La tabla de incidentes y sus frecuencias y pesos ha sido de la siguiente manera, la cual sirvió para crear una batería de ejercicios para la simulación:

Tabla Incidente Generado			
Tipo Incidente	Frecuencia	Probabilidad	Peso
1	2	30	1
2	2	20	5
3	1	15	10
4	1	10	15
5	1	5	20

Figura 7.2 – Tabla de incidentes para la experimentación numérica

- La tabla de tiempos esperados fue de la siguiente manera:

Tabla Tiempos Esperados					
Tipo Técnico	TRIVIAL	LEVE	MODERADO	SEVERO	URGENTE
1	5	10	X	X	X
2	4	8	15	20	X
3	3	6	10	15	30

Figura 7.3 – Tabla de tiempos esperados para la experimentación numérica

1.4 Ejecución

Al ejecutar 30 veces el problema, se obtienen los siguientes resultados:

Ejecución	MinIncrease	MI + GRASP
1	0.752	1.603
2	0.876	1.625
3	0.803	1.646
4	0.925	1.476
5	0.742	1.527
6	0.972	1.544
7	0.703	1.675

8	0.792	1.602
9	0.865	1.451
10	0.724	1.460
11	0.836	1.528
12	0.865	1.529
13	0.954	1.484
14	0.791	1.472
15	0.692	1.527
16	0.649	1.534
17	0.860	1.535
18	0.875	1.591
19	0.750	1.493
20	0.853	1.440
21	0.689	1.420
22	0.815	1.437
23	0.763	1.460
24	0.918	1.576
25	0.986	1.437
26	0.949	1.549
27	0.789	1.647
28	0.674	1.515
29	0.869	1.437
30	0.924	1.465

Figura 7.4 – Tabla de eficiencias para la experimentación numérica

A simple vista se puede evidenciar la superioridad numérica que se obtiene aplicando GRASP contra el "primero el mejor".
A continuación, analizamos por medio de pruebas los datos para rechazar o aceptar la hipótesis de que GRASP mejora MinIncrease.

1.4.1 Prueba de distribución normal

Se desea probar que los resultados obtenidos siguen una distribución normal:
H_0 = Los valores siguen una distribución normal.
H_1 = Los valores no siguen una distribución normal.
Utilizando la herramienta SPSS (detalle completo en el anexo), podemos ver los resultados de 4 maneras:

- Prueba de asimetría y curtosis.
- Prueba de Kolmogorov-Smirnov.
- Prueba de Shapiro-Wilk.
- Prueba de histograma y normal esperada.

En estos casos estaremos usando un nivel de significancia del 95%

Resultados:

Prueba	MinIncrease + Primero Mejor	MinIncrease + GRASP
Asimetría y curtosis	Asimetría: -0.217 Curtosis: ‘0.814 Ambos dentro de [+1.96, -1.96] ACEPTA H_0	Asimetría: 0.069 Curtosis: -0.6 Ambos dentro de [+1.96, -1.96] ACEPTA H_0
Kolmogorov-Smirnov	Valor de significancia: 0.2 Mayor que 0.05 ACEPTA H_0	Valor de significancia: 0.2 Mayor que 0.05 ACEPTA H_0
Shapiro-Wilk	Valor de significancia: 0.497 Menor que 0.05 ACEPTA H_0	Valor de significancia: 0.098 Mayor que 0.05 ACEPTA H_0
Histograma y normal esperada	Histograma curva: Sí Cercano a lo esperado: Sí ACEPTA H_0	Histograma curva: Sí Cercano a lo esperado: Sí ACEPTA H_0

Figura 7.5 – Tabla de resultados del análisis de la distribución normal

Vemos que en esta prueba, ambos pasan todas las pruebas. Por lo tanto, se acepta la hipótesis nula: ambos son distribuciones normales.

1.4.2 Prueba F de Fisher

La prueba F de Fisher o F-test nos permite saber si hay un nivel de varianza similar en las pruebas.

Se tiene la hipótesis que:

H_0 = Las varianzas son significativamente homogéneas.
H_1 = Las varianzas son significativamente distintas.

Utilizando SPSS (detalle en el anexo), con nivel de significancia es 95%, vemos que:

Test of Homogeneity of Variances

Eficiencia

Levene Statistic	df1	df2	Sig.
3.543	1	58	.065

Figura 7.6 – Tabla de resultados del análisis de la homogeneidad de varianzas

Si vemos Sig, 0.65 es mayor a 0.05, por lo tanto se acepta la hipótesis nula de que ambas varianzas son homogéneas.

1.4.3 Prueba Z

La prueba Z nos permitirá saber cuál de los algoritmos es el más efectivo.
Se plantean las siguientes hipótesis:

$$H_0: X_1 = X_2$$

$$H_1: X_1 \neq X_2$$

Donde:
H_0 = Se acepta que MinIncrease con GRASP es igual a MinIncrease con primero mejor.
H_1 = Se acepta que MinIncrease con GRASP son diferentes a MinIncrease con primero mejor.

```
z_statistic  p_value cohens_d

  40.50143    .00000  7.39451

Number of cases read:  1    Number of cases listed:  1
```

Figura 7.7 – Resultados del análisis de la prueba Z

Con SPSS vemos que:

- El valor de significancia es menor que 0.5 (dado por el grado de confiabilidad). Por lo tanto, **se rechaza** la hipótesis nula.

Con esto comprobamos que MinIncrease con GRASP es superior a MinIncrease solo.

CAPÍTULO 8

1 Funcionamiento básico del programa

El funcionamiento básico del programa es el siguiente:

Programa Asignación Mesa de Ayuda:
1. Lectura de cantidad de técnicos
2. Lectura de tipos de incidentes
3. Lectura de matriz de tiempos esperados
4. Repetición GRASP:
4.1 Repetir hasta tiempo límite:
4.1.1 Lectura o Creación de incidentes
4.1.2 Asignación por algoritmos
5. Escribir Reportes

Fin Programa

Figura 8.1 – Esquema del funcionamiento del programa

Explicación:

1. Se leen la tabla de técnicos y se carga con el tipo Técnico en la lista de técnicos. La cantidad y los tipos dependerán de los datos en la tabla de la base de datos.
2. Se cargan los tipos de incidentes para identificarlos que parten de la tabla de ocurrencia de incidentes, y, en caso de generarlos, saber la frecuencia y posibilidad de aparición, además de su peso o priorización.
3. Se lee la matriz de tiempos esperados y se carga al programa para saber los tiempos esperados por técnico e incidente.
4. Se repite varias veces la simulación debido al algoritmo GRASP. Esto nos permitirá tener mejores valores de eficiencia debido a la repetición de simulaciones.

4.1 Se repite hasta un tiempo límite. Este tiempo puede ser variado, pero para estos ejemplos el tiempo *t* lo estaremos considerando como minutos.

4.1.1 Se leen los incidentes cargados para un tiempo t o se generan aleatoriamente, dependiendo del modo escogido.

4.1.2 Se asignan los incidentes con los algoritmos mencionados en capítulos anteriores.

5. Se escriben los reportes de eficiencia de los técnicos y el log de procedimiento por cada tiempo t que haya pasado.

La ejecución de este programa va a ser apoyado mediante una interfaz gráfica, la cual nos permitirá interactuar con la base de datos, modificando el valor de los parámetros de número de técnicos, pesos de incidentes y tiempos esperados. Así como facilitar la carga de datos y la visualización de las eficiencias y los reportes del desempeño de cada técnico.

2 Reportes

El programa simulador arrojará dos archivos de reporte de desempeño: Log de asignaciones y resumen de desempeño.

2.1 Log de Asignaciones

El log de asignaciones tendrá la siguiente forma:

```
TIEMPO T = 7
--------
Problema: 9 Dificultad: 4 Peso: 15
Lista MinIncrease: {5=300.0, 6=300.0, 7=340.0, 8=300.0, 9=400.0, 10=300.0, 11=300.0, 12=300.0, 13=540.0, 14=300.0}
GRASP RCL: [5, 6, 7, 8, 9, 10, 11, 12, 14]
Tecnico Seleccionado: 11
--------
Problema: 10 Dificultad: 2 Peso: 5
Lista MinIncrease: {0=60.0, 1=50.0, 2=50.0, 3=50.0, 4=60.0, 5=40.0, 6=40.0, 7=56.0, 8=40.0, 9=80.0, 10=40.0, 11=140.0, 12=40.0, 13=156.0, 14=60.0}
GRASP RCL: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14]
Tecnico Seleccionado: 0
--------
ESTADO DE LOS TECNICOS:
El tecnico con ID 9 esta empezando a resolver el incidente #4
El tecnico con ID 11 esta empezando a resolver el incidente #9
El tecnico con ID 13 termino el incidente #3
```

Figura 8.2 – Ejemplo de sección del log de asignaciones

El ejemplo muestra las siguientes partes:

- Tiempo *t* que está ocurriendo en esa asignación.
- Problema: su identificación, dificultad y peso.
- Lista MinIncrease y RCL del GRASP para saber la asignación.
- Técnico seleccionado a resolver el incidente.
- Estado de los técnicos: que técnicos están resolviendo que incidentes y cuales acaban de resolver un incidente en el tiempo *t*.

2.2 Resumen de Desempeño

El resumen de desempeño tendrá la siguiente forma:

```
Tecnico ID: 3 (Tipo: 1)
-Ratio Completo: 105
-Ratio Por Completar: 5
-Efectividad: 21.0
-Trabajos realizados:
--Incidente #11 (Tipo: 2 Prioridad: 5 Tiempo Esperado: 10 Tiempo Real: 12 Diferencia: 2)
--Incidente #21 (Tipo: 1 Prioridad: 1 Tiempo Esperado: 5 Tiempo Real: 6 Diferencia: 1)
--Incidente #39 (Tipo: 2 Prioridad: 5 Tiempo Esperado: 10 Tiempo Real: 10 Diferencia: 0)
-Trabajo en proceso:
--Incidente #38 (Tipo: 1 Peso: 1 Tiempo Esperado: 5)
-Trabajos asignados por realizar:
--Incidente #86 (Tipo: 1 Prioridad: 1 Tiempo Esperado: 5)
```

Figura 8.3 – Ejemplo de sección del resumen de desempeño

Las partes del resumen de desempeño son las siguientes:

- Identificación del técnico y su tipo.
- Ratio de incidentes resueltos.
- Ratio de incidentes por resolver.
- Efectividad personal.
- Incidentes resueltos (hasta el tiempo *t*).
- Incidentes en resolución (hasta el tiempo *t*).
- Incidentes por resolver (hasta el tiempo *t*).

CAPÍTULO 9

1 Conclusiones y Recomendaciones

En este apartado analizaremos las conclusiones que otorga el proyecto, así como recomendaciones para su uso y trabajos futuros que pueden salir a partir de este proyecto.

1.1 Conclusión del Proyecto

Este proyecto tiene como meta la aplicación de dos algoritmos (algoritmo de asignación MinIncrease y algoritmo metaheurístico GRASP) para la solución de un caso particular de la asignación de tareas que es la situación de una mesa de ayuda, con sus respectivas particularidades y limitaciones.

El proyecto logra ofrecer una opción de algoritmos de asignación de tareas estocásticas en línea con las características y limitaciones que tienen las mesas de ayuda. Asimismo, se producen archivos de texto que ofrecen un resumen de los incidentes solucionados y por solucionar en la mesa de ayuda por cada técnico y su respectivo ratio de efectividad, así como un archivo detallando paso por paso la asignación de los incidentes y los tiempos que se asignaron y resolvieron los incidentes.

Vemos también que la combinación de dichos algoritmos resulta productiva que el mismo algoritmo de asignación MinIncrease independiente. Debido a las condiciones del entorno en las mesas de ayuda mencionado en la problemática, era conveniente introducir un elemento de relajación para la toma de decisiones después de aplicar el algoritmo MinIncrease, y no tomar siempre el más efectivo. Esto se demostró en la experimentación numérica, de cómo en una situación con condiciones similares, la adición del algoritmo GRASP mejoró notablemente el desempeño del programa contra únicamente el algoritmo MinIncrease, justificando su uso.

Con ello, pudimos cumplir dos objetivos clave en este proyecto:

- Aplicación de un algoritmo para la asignación estocástica y en línea adaptado a la situación de una mesa de ayuda (MinIncrease).
- Uso del algoritmo GRASP para la mejora de las asignaciones dadas por MinIncrease debido a las condiciones del entorno del proyecto (la mesa de ayuda), otorgándole una mejora significativa.

Es de notar que la simulación para la asignación en la experimentación numérica se hizo con condiciones que fueran rápidas de calcular, pero a la vez significativas. Se puede elaborar una simulación más extensa, con una mesa de ayuda aún más grande, así como una simulación en donde se simule días y/o semanas de trabajo. Sin embargo, esto es limitado por el poder computacional y el tiempo de ejecución para la simulación completa.

1.2 Recomendaciones

Para la aplicación de este trabajo, es recomendable que se haga la simulación con una mesa de ayuda de varios miembros con rangos distintos, así como un alto número de incidentes que corresponda asignar con diferentes tipos de severidad. Una lista de técnicos “idénticos” y una batería baja de incidentes o una clasificación inapropiada de

ellos pueden llevar a simulaciones con resultados incorrectos o resultados irrelevantes para problemas de complejidad baja.

Para ello, es necesario que se tenga documentado apropiadamente en la empresa o situación que se desea aplicar, documentos que establezcan los rangos de los técnicos de la empresa y su cantidad, así como establecer la clasificación de un incidente con la severidad que se merezca.

Para los tiempos esperados de cumplimiento de tareas, es necesario también establecer el promedio aceptable o frecuente de resolución de incidentes de una severidad en particular. Esto puede establecerse por medio de documentos como los Acuerdos de Nivel de Servicio o SLA (Service Level Agreement).

1.3 **Trabajos Futuros**

Las aplicaciones de este trabajo pueden llevarse a varias aplicaciones:

- Aplicación directa: Se puede utilizar la simulación como algoritmo en “back end” para un programa de asignación de tareas o manejo de proyectos. La aplicación podría hacerse vía comunicación con la base de datos. Asimismo, es posible mejorar el reporte de los entregables con gráficos y estadísticas propias de un programa de hoja de cálculo o programas estadísticos.
- Cambio de entorno: Es posible cambiar el código y propiedades como el alfa del algoritmo GRASP para las características y limitaciones que tengan. Con ello, es posible que se adapte a problemas de asignación de tareas similares que tengan como propiedad que sean estocásticos y en línea, además de tener limitaciones en el entorno que hagan la elección voraz del algoritmo MinIncrease menos óptima que con una elección aleatoria y voraz.
- Caso de estudio: Este proyecto puede servir como caso de estudio para problemas de asignación estocásticos y en línea que tengan máquinas no relacionadas. Sería posible generalizarlo incluso para casos de que cada máquina sea distinta con tiempos de procesamiento esperados distintos, y proponer mejoras al proyecto.

Referencias bibliográficas

PINEDO, Michael
2010 *Scheduling: Theory, Algorithms and Systems*
4ta Edición. New York: Editorial Springer.

TUPIA, Manuel
2009 *Fundamentos de Inteligencia Artificial*
1ra Edición. Lima: Tupia Consultores y Auditores SAC.

CORMEN, Thomas; LEISERSON, Charles; RIVEST, Ronald; STEIN, Clifford
2001 *Introduction to Algorithms*
2da Edición. USA: MIT Press.

LEE, Jon
2004 *A First Course in Combinatorial Optimization.*
1ra Edición.USA: Cambridge University Press.

PAPADIMITRIOU, Christos; STEIGLITZ, Kenneth
1998 *Combinatorial Optimization: Algorithms and Complexity.*
1ra Edición, New York: Courier Dover Publications.

BRASSAR, G; BRATLEY, P.
1996 *Fundamental of Algorithmics*
1ra Edición, New Jersey: Editorial Prentice Hall.

TUPIA, Manuel
2013 *Gestión de servicios de tecnologías de información bajo la óptica de ITIL.V3*
1ra Edición. Lima: Tupia Consultores y Auditores SAC.

MEJÍA, Miguel; CHAVEZ-BEDOYA, Luis; VERA, Carlos; CORNEJO, Christian; ARAGÓN, Lucy; ROJAS, Jonatán
2012 *Investigación Operativa I*
Edición 2012. Lima: PUCP, Facultad de Ciencias e Ingeniería.

UETZ, Marc
2002 *Algorithms for deterministic and stochastic scheduling*
Cuvillier Verlag, Gottingen, Germany,

FEO, Thomas A. y RESENDE, Mauricio G.C.
1995 *Greedy Randomized Adaptive Search Procedures*
Journal of Global Optimization. 6, pp 109-133.

MEGOW, Nicole; UETZ, Marc; VREDEVELD, Tjark
2006 *Models and Algorithms for Stochastic Online Scheduling*
Mathematics of Operations Research. Vol 31, Issue 3, August 2006, pp 513-525

DRESS, Andreas; YINFENG, Xu; BINHAI Zhu
2007 *Combinatorial Optimization and Applications: First International Conference*.

COCOA 2007, Xi'an, China, August 14-16, 2007. USA: Springer.

BLUM, Christian; ROLI Andrea
2003 *Metaheuristics in combinatorial optimization: Overview and conceptual comparison.*
ACM Computing Surveys. New York, Vol. 35, No. 3, pp 268-308

RAMIREZ RODRÍGUEZ, César Oswaldo
2006 *Un algoritmo GRASP con doble relajación para resolver problema del flow shop scheduling.*
Título de Ingeniero Informático. Lima: PUCP, Facultad de Ciencias e Ingeniería.

TUPIA, Manuel
2004 *Un algoritmo GRASP para resolver el problema de la programación de tareas dependientes en máquinas diferentes.*
Conferencia Latino Americana de Informática CLEI (30, 2004, Perú), Editores Solar M; Fernández-Baca D; Cuadros-Vargas E. p.129-139.

BARD, J y RIOS, R
2000 *Heurísticas para el secuenciamiento de tareas en líneas de flujo*
Universidad Autónoma de Nuevo León (México), Universidad de Texas (USA)

SCHULZ. A. y SKUTELLA, M.
2002 *Scheduling unrelated machines by randomized rounding*
SIAM Journal on Discrete Mathematics 15 (2002), pp. 450–469

AFRATI F. y otros.
1999 *Approximation schemes for minimizing average weighted completion time with release dates*
Proc. 40th Ann. Symp. on Foundations of Computer Science, pp. 32–43.

ANDERSON, E. y POTTS, C.
2004 *On-line scheduling of a single machine to minimize total weighted completion time*
Mathematics of Operations Research 29, pp. 686–697

FIAT, A. y WOEGINGER, G.
1999 *On-line scheduling on a single machine: Minimizing the total completion time,*
Acta Informatica 36, pp. 287–293

HEYDENREICH, Birgit, Rudolf MULLER y Marc UETZ
2006 *Decentralization and Mechanism Design for Online Machine Scheduling*
Algorithm Theory – SWAT 2006, 10th Scandinavian Workshop on Algorithm Theory, Riga, Latvia, July 6-8, 2006.

SCHULZ, A.
2005 *New old algorithms for stochastic scheduling*
Dagstuhl Seminar Proceedings, no. 05031, Internationales Begegnungs- und Forschungszentrum (IBFI), Schloss Dagstuhl, Germany, 2005.

ITIL Technologies
2014 "Service Desk Objective in ITIL Foundation". Consulta: 06 de Junio del 2014. <http://www.itilfoundation.org/Service-Desk-Objectives-in-ITIL-Foundation_43.html>

THE ECONOMIST
2014 *"Cyber Incident Response: Are business leaders ready?"* Consulta: 06 de Junio del 2014. <http://www.economistinsights.com/technology-innovation/analysis/cyber-incident-response>

DIARIO TI
2014 *"ESET lanza curso de sobre Gestión y Respuesta de incidentes"* Consulta: 06 de Junio del 2014. <http://diarioti.com/eset-lanza-curso-sobre-gestion-y-respuesta-de-incidentes/65603>

CA TECHNOLOGIES
2014 *"CA Service Desk Manager"*, Consulta: 06 de Junio del 2014 < http://www.ca.com/us/intellicenter/ca-service-desk-manager.aspx>

FRONTRANGE SOLUTIONS
2014 *"HEAT Service Management"*, Consulta: 06 de Junio del 2014 < http://www.frontrange.com/heat/products/service-management>

MARVAL SOFTWARE
2014 *"MSM ITSM"*, Consulta: 06 de Junio del 2014 <http://www.frontrange.com/heat/products/service-management>

OMNINET
2014 *"OMNITRACKER ITSM"*, Consulta: 06 de Junio del 2014 < http://www.omnitracker.biz/index.php?id=138&L=2 >

ESPACIO TECNOLÓGICO MOLINÓN
2014 *"ProactivaNET"*, Consulta: 06 de Junio del 2014 <http://www.proactivanet.com/proactivanet-service-desk >

HEWLETT-PACKARD DEVELOPMENT COMPANY
2014 *"HP Service Anywhere"*, Consulta: 06 de Junio del 2014 <http://www8.hp.com/us/en/software-solutions/service-anywhere-ITSM-ITIL-service-desk/index.html?>