PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

IMPLEMENTACIÓN DE UN ALGORITMO GRASP CON DOBLE RELAJACIÓN QUE PERMITA RESOLVER EL PROBLEMA DE LA ASIGNACIÓN DE CITAS MÉDICAS EN HOSPITALES

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

César David García Casanova

ASESOR: Ing. Rony Cueva Moscoso

Lima, Febrero del 2015

RESUMEN

En la actualidad, obtener una cita médica dentro de las instituciones de salud pública en el Perú se ha convertido en una difícil tarea. Esto ocurre debido a la gran demanda de citas médicas, las deficiencias a nivel de infraestructura y el déficit de médicos que existe.

Como objetivo de este proyecto, se busca implementar un algoritmo meta heurístico GRASP con doble relajación que permita asignar citas médicas tomando en consideración factores de vulnerabilidad de cada paciente, tales como la edad, condición socioeconómica, carga familiar y carga por enfermedad; y factores que permitan asignarle a dicho paciente un buen médico, tales como días de espera para la próxima cita y experiencia del médico tratante.

En este proyecto se compara la solución obtenida por el modelo actual de entrega de citas médicas y el modelo propuesto, el cual hace uso del algoritmo GRASP implementado. Los resultados obtenidos comprueban que el algoritmo GRASP obtiene mejores soluciones, desde el punto de vista del objetivo de este proyecto, el cual es priorizar a aquellos pacientes con mayor necesidad de atención.

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: **IMPLEMENTACIÓN DE UN ALGORITMO GRASP CON DOBLE RELAJACION PARA RESOLVER EL PROBLEMA DE LA ASIGNACION DE CITAS MEDICAS EN HOSPITALES**

ÁREA: Ciencias de la computación

PROPONENTE: Rony Cueva Moscoso

ASESOR: Rony Cueva Moscoso

ALUMNO: César David García Casanova

CÓDIGO: 20077041

TEMA N°: 565

FECHA: San Miguel, 14 de diciembre de 2014

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
Facultad de Ciencias e Ingeniería
08 APR. 2015
MIGUEL MEJIA PUENTE
DECANO

DESCRIPCIÓN

En el Perú, el sector Salud maneja principalmente dos tipos de seguros públicos: EsSalud y el Seguro Integral de Salud (SIS). Éstos acogen al 58.8% de la población nacional, mientras que los seguros particulares a sólo un 7.5% de la misma, según un estudio realizado a nivel nacional por el Instituto Nacional de Estadística e Informática INEI (Enero – Marzo del 2013).

A pesar de la gran acogida que posee el servicio de salud pública, la insatisfacción del servicio es alta donde una de cada dos personas no está satisfecha con el servicio brindado. Algunas de las causas son: la falta de salas de operación, citas médicas demasiado prolongadas, falta de camas hospitalarias y el bajo stock que se maneja en las farmacias de los centros médicos; entre otros.

De los problemas mencionados, el 34.7% de los asegurados considera a la distribución de citas como el principal problema que aqueja al sector salud. Esta cifra es justificada, ya que, según la defensoría del asegurado, el tiempo promedio que demora un usuario en obtener atención médica es de 30 días calendarios, alcanzando en algunos casos los 2 meses. Asimismo, se pudo observar que las citas médicas son entregadas en orden de llegada, es decir, no existe una prioridad sobre aquellos pacientes con mayor necesidad de atención.

Al tener un número limitado de citas por día, cantidad de médicos y centros de salud disponibles, la solución que se desarrollará deberá considerar diversos factores como la edad, condición socioeconómica, historial del paciente, tipo de enfermedad diagnosticada al paciente (enfermedades de interés nacional: VIH y TBC, crónicas, contagio rápido, etc.) para establecer la prioridad de atención del paciente.

Av. Universitaria 1801
San Miguel, Lima – Perú

Apartado Postal 1761
Lima 100 – Perú

Teléfono:
(511) 626 2000 Anexo 4801

Finalmente, al observar que uno de los problemas del sector salud público es la mala distribución de citas médicas y el alto tiempo que toma conseguirlas, se propone crear un algoritmo GRASP, el cual permitirá distribuir las citas médicas disponibles de una forma más eficiente. De esta forma, se buscará minimizar los tiempos de espera del paciente al querer obtener una cita médica, mejorar la calidad de la atención, dar atención médica a quien más pronto lo necesite y optimizar los recursos que maneja el sector salud.

OBJETIVO GENERAL

Desarrollar un algoritmo GRASP con doble relajación que permita resolver el problema de la asignación de citas médicas entre pacientes y doctores en hospitales.

OBJETIVOS ESPECÍFICOS

Los objetivos específicos del presente proyecto son:

1. Definir las estructuras de datos necesarias que usará el algoritmo para obtener la solución al problema propuesto.
2. Definir las funciones objetivo que serán utilizadas en el algoritmo. Una de ellas permitirá seleccionar al mejor paciente y la otra, el mejor doctor disponible.
3. Diseñar el pseudocódigo correspondiente del algoritmo GRASP que se usará para la solución.
4. Desarrollar la experimentación numérica adecuada para el problema utilizando el algoritmo GRASP a implementar y su adaptación a un algoritmo voraz.
5. Implementar un prototipo funcional que permita el ingreso de datos iniciales del problema y la visualización de los resultados.

ALCANCE

Este proyecto se basará en el desarrollo de un algoritmo GRASP con doble relajación para la distribución de citas médicas entre pacientes y doctores en hospitales cuyo horario de atención médica es variable en el tiempo, tomando en consideración cada uno de los factores del paciente seleccionados inicialmente, tales como la edad, carga familiar, costo por enfermedad diagnosticada, ingresos económicos; y los de cada médico: experiencia en años, horas de trabajo asignada, horas de trabajo disponibles. Además, debido a que se desea obtener el mejor paciente (prioridad de atención) y al mejor doctor (según los factores antes mencionados), se usará el criterio de doble relajación para obtener cada uno de estos.

El algoritmo a desarrollar contará con una fase de construcción y una de mejoría. Para la fase de construcción, se seleccionará el valor alfa de una tabla de distribución de probabilidad uniforme y discreta Además, por teoría, este algoritmo contará con 10,000 iteraciones; sin embargo, esta cantidad puede verse modificada dependiendo de los resultados que se obtengan de la experimentación numérica.

Máximo: 100 páginas

A mi abuelo Pascual por enseñarme que la vida siempre hay que verla con una sonrisa en el rostro.

A mi madre Elsa por su sacrificio, valentía, dedicación, paciencia y profundo amor. Madre, eres el motor principal de este primer logro.

A mi padre David por su amor, gran amistad y buenos consejos.

A mis hermanos por su apoyo, cariño y comprensión.

A mis padrinos Armando y Susana por su paciencia y amor.

Y a mi amiga y compañera Geraldine por acompañarme en las buenas y en las malas durante este largo camino.

AGRADECIMIENTOS

A mi asesor, Ing. Rony Cueva Moscoso, por su total apoyo y paciencia durante este proyecto de fin de carrera.

INDICE GENERAL

Índice de figuras

Índice de Tablas

CAPÍTULO 1: GENERALIDADES

En el siguiente apartado, se presenta la problemática, los objetivos y los resultados esperados del presente proyecto.

1 Problemática

En el Perú, el sector Salud maneja principalmente dos tipos de seguros públicos: EsSalud y el Seguro Integral de Salud (SIS). Éstos acogen al 62% de la población nacional, mientras que los seguros particulares a sólo un 7.4% de la misma, según un estudio realizado a nivel nacional por el Instituto Nacional de Estadística e Informática INEI (Enero – Febrero – Marzo del 2014) [INEI 2014].

Tipo de seguro de salud/ Área de residencia	Ene-Feb-Mar 2013	Ene-Feb-Mar 2014	Variación (Puntos Porcentuales)
Nacional	**66,3**	**69,5**	**3,2 %**
Únicamente EsSalud	24,8	25,7	0,9 %
Únicamente SIS	34,0	36,3	2,3 %
Con otros seguros	7,5	7,4	-0,1%

Tabla 1.1 Población afiliada a seguro de salud, según tipo de seguro [INEI 2014]

A pesar de la gran cantidad de usuarios que posee el servicio de salud público, la insatisfacción del servicio es un problema resaltante: una de cada dos personas no está satisfecha con el servicio brindado, esto a causa de la falta de salas de operación, citas médicas demasiado prolongadas, falta de camas hospitalarias y el bajo stock que se maneja en las farmacias de los centros médicos [INEI 2014].

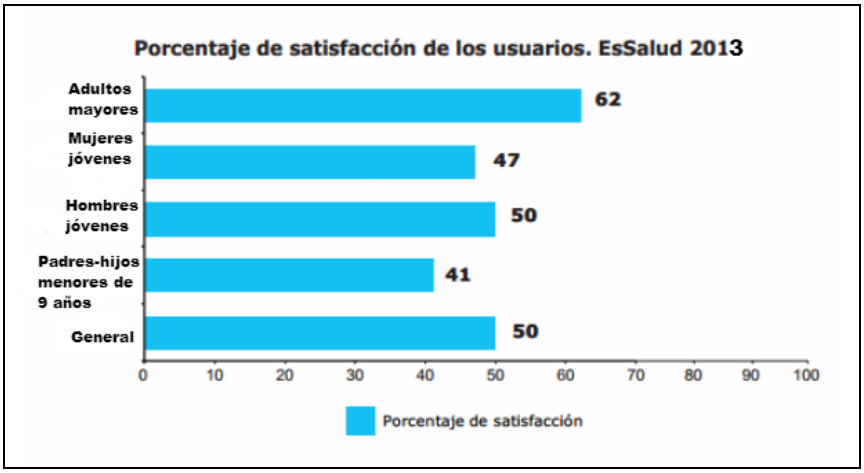


Figura 1.1 Niveles de satisfacción de usuarios en EsSalud [INEI 2014]

De los problemas mencionados anteriormente, el 34.7% de los asegurados considera a la distribución de citas como el principal problema que aqueja al sector salud. Esta cifra es justificada, ya que, según la defensoría del asegurado, el tiempo promedio que demora un usuario en obtener atención médica es de 30 días calendarios, alcanzando en algunos casos los 2 meses. [INEI 2014].

Asimismo, se pudo observar que las citas médicas son entregadas en orden de llegada, es decir, no existe una prioridad sobre aquellos pacientes con mayor necesidad de atención [INEI 2014].

4. ESTADISTICAS A NIVEL DE RED ASISTENCIAL
4.1 REDES ASISTENCIALES DE LIMA
4.1.1 RECLAMOS Y SOLICITUDES DE MEDIACION INGRESADOS AL SIAS

COD	MATERIA	RED REBAGLIATI	%	RED ALMENARA	%	RED SABOGAL	%	TOTAL	%
01	PROBLEMA DE CITAS	1060	39.07%	765	29.34%	955	35.42%	2780	34.68%
02	PROBLEMAS EN LA ENTREGA DE MEDICAMENTOS	169	6.23%	510	19.56%	177	6.57%	856	10.68%
08	PROBLEMAS CON LAS REFERENCIAS	252	9.29%	117	4.49%	343	12.72%	712	8.88%
16	FALTA DE INFORMACION DE LOS PROCESOS INSTITUCIONALES	261	9.62%	242	9.28%	170	6.31%	673	8.40%
22	RETRASO EN LA PRESTACION POR CAUSAS ADMINISTRATIVAS (INCLUYE CITT)	333	12.27%	149	5.72%	138	5.12%	620	7.73%
10	POSTERGACION EN LA ATENCION EN CONSULTORIOS	39	1.44%	198	7.59%	89	3.30%	326	4.07%
29	PROBLEMA DE ATENCION EN AYUDA AL DIAGNOSTICO	87	3.21%	52	1.99%	183	6.79%	322	4.02%
11	FALTA DE INFORMACION AL ASEGURADO SOBRE SU SALUD Y/O TRATAMIENTO	67	2.47%	32	1.23%	145	5.38%	244	3.04%
24	FALTA DE ATENCION EN MEDICINA GENERAL Y/O ESPECIALIZADA	86	3.17%	44	1.69%	103	3.82%	233	2.91%
33	IMPUNTUALIDAD	18	0.66%	139	5.33%	18	0.67%	175	2.18%
20	PROBLEMAS EN EL ACCESO AL SERVICIO DE EMERGENCIA	48	1.77%	37	1.42%	55	2.04%	140	1.75%
35	FALTA DE INSUMOS REACTIVOS	12	0.44%	114	4.37%	10	0.37%	136	1.70%
09	POSTERGACION DE INTERVENCION QUIRURGICA	64	2.36%	15	0.58%	48	1.78%	127	1.58%
27	ATENCION ASISTENCIAL INADECUADA	57	2.10%	37	1.42%	24	0.89%	118	1.47%
30	PROBLEMAS PARA LA OBTENCION DE CITAS POR TELEFONO/INTERNET Y OTROS QUE BRINDA ESSALUD EN LINEA	14	0.52%	23	0.88%	67	2.49%	104	1.30%
14	PROBLEMAS DE ACREDITACION	31	1.14%	16	0.61%	46	1.71%	93	1.16%
38	INCUMPLIMIENTO DE DEBERES DE LOS ASEGURADOS	16	0.59%	18	0.69%	50	1.85%	84	1.05%
26	MATERIAS QUE NO CORRESPONDEN A ESSALUD	25	0.92%	28	1.07%	13	0.48%	66	0.82%
03	MALTRATO EN LA ATENCION ASISTENCIAL Y/O	26	0.96%	11	0.42%	10	0.37%	47	0.59%

Figura 1.2 Razón de reclamos
[INEI 2014]

Además, los horarios de atención de cada médico pueden ser fijos o variar en el tiempo. Un médico contratado a tiempo completo posee un horario fijo establecido, el cual sólo podrá verse modificado en caso de enfermedad o licencia de algún tipo. Por otro lado, existe el caso de los médicos contratados por horas o medio tiempo, los cuales, por lo general, son la mayoría. Éstos poseen un horario de atención que puede verse modificado mes a mes, dependiendo de la disponibilidad del galeno y previa coordinación con el centro de salud. La programación de horarios de atención por médico se realiza mensualmente y está a cargo de la Dirección General de cada centro. [MINSA 2014]

Sumado al problema de la distribución de citas, el sector salud posee otras limitaciones con respecto a la política que maneja el sector salud en torno al proceso de atención, el cual tiene las siguientes características principales:

- Un médico sólo puede atender al día 20 citas de aproximadamente 15 minutos como máximo.

- Todo paciente debe ser atendido, en primer lugar, por un médico general, para luego ser derivado a un médico especialista (dependiendo de los síntomas del paciente).

- La entrega de citas debería efectuarse en el orden de llegada de los pacientes. Sin embargo, en la práctica se genera una "lista de espera" de pacientes debido a la escasez de turnos de atención.

Al tener un número limitado de citas por día y cantidad de médicos, la solución que se desarrollará deberá considerar diversos factores como la edad, condición socioeconómica, carga de enfermedad diagnosticada (enfermedades de interés nacional: VIH y TBC, crónicas, contagio rápido, etc.) y carga familiar para establecer la prioridad de atención del paciente.

Finalmente, al observar que uno de los problemas del sector salud es la mala asignación de citas médicas y el alto tiempo que toma conseguirlas, se propone crear un algoritmo GRASP, el cual permitirá distribuir las citas médicas disponibles de una forma más eficiente. De esta forma, se buscará minimizar los tiempos de espera del paciente al querer obtener una cita médica, mejorar la calidad de la atención, dar atención médica a quien más pronto lo necesite y optimizar los recursos que maneja el sector salud.

2. Objetivo general

Desarrollar un algoritmo GRASP con doble relajación que permita resolver el problema de la asignación de citas médicas entre pacientes y doctores en hospitales.

3. Objetivos específicos

Los objetivos específicos del proyecto son los siguientes:

1. Definir las estructuras de datos necesarias que usará el algoritmo para obtener la solución al problema propuesto.

2. Definir las funciones objetivo que serán utilizadas en el algoritmo. Una de ellas permitirá seleccionar al mejor paciente y la otra, el mejor doctor disponible.

3. Diseñar los pseudocódigos correspondientes al algoritmo Voraz y el algoritmo GRASP usados para la solución.

4. Desarrollar la experimentación numérica para comparar el algoritmo Voraz y el algoritmo GRASP a implementar, que confirme si el modelo GRASP propuesto es mejor que el modelo Voraz.

5. Diseñar un prototipo funcional que permita el ingreso de datos iniciales del problema y la visualización de los resultados.

4. Resultados esperados

Según los objetivos específicos señalados en el punto anterior, se esperan los siguientes resultados:

1. Estructuras de datos necesarias que soporten la solución al problema de la asignación de citas médicas. Estas estructuras serán usadas en el algoritmo para representar las características del médico y paciente. (Obj. Esp. 1)

2. Documento de especificación de las funciones objetivo que serán usadas en la solución. Éste permitirá comprender la funcionalidad de las funciones objetivo a desarrollar. (Obj. Esp. 2)

3. Documento que detalle el pseudocódigo del algoritmo Voraz y GRASP que se desarrollarán para la solución. Éste permitirá comprender el funcionamiento de los algoritmos de forma global. (Obj. Esp. 3)

4. Documento que muestre el detalle y las conclusiones de la experimentación numérica basada en la prueba de comparación de medias. (Obj. Esp. 4)

5. Prototipo funcional que permita el ingreso de datos iniciales y visualizar los resultados obtenidos al final de la ejecución. (Obj. Esp. 5)

5. Herramientas, Métodos, Metodologías y Procedimientos

5.1 Introducción

A continuación se presentan las herramientas, métodos, metodologías y procedimientos utilizados para poder obtener cada resultado esperado.

Resultados esperado	Herramientas a usarse
Estructuras de datos necesarias que soporten la solución al problema de la asignación de citas médicas. Estas estructuras serán usadas en el algoritmo para representar las características del médico y paciente.	**Extreme Programming** es una metodología de desarrollo que está más enfocada en la programación que en la documentación. **Netbeans** es un entorno de desarrollo pensado para elaborar programas. **Java** es un lenguaje de programación que permite la implementación de programas.
Documento de especificación de las funciones objetivo que serán usadas en la solución. Éste permitirá a comprender la funcionalidad de las funciones objetivo a desarrollar	**Microsoft Word 2010**
Documento de especificación del pseudocódigo del algoritmo GRASP a desarrollar para la solución.	**Microsoft Word 2010** **Metodología del algoritmo GRASP** es una metodología que permitirá guiar la implementación del algoritmo necesario para este proyecto.
Documento que muestre el detalle y las conclusiones de la experimentación numérica basada en la prueba de	**Microsoft Excel 2010** **Microsoft Word 2010**

Resultados esperado	Herramientas a usarse
comparación de medias.	**ANOVA** es un test estadístico usado para la comparación de dos o más medias entre dos o más grupos.
Prototipo funcional que permita el ingreso de datos iniciales y visualizar los resultados obtenidos al final de la ejecución.	**Extreme Programming** es una metodología de desarrollo que está más enfocada en la programación que en la documentación. **Netbeans** es un entorno de desarrollo pensado para elaborar programas. **Java** es un lenguaje de programación que permite la implementación de programas.

Tabla 1.2. Resultados esperados / Herramientas
Fuente: Elaboración propia

5.2 Herramientas y Metodologías del Producto

A continuación se describen las metodologías a utilizar con respecto al proyecto en general.

i. Extreme Programming (XP)

Para este proyecto se hizo una adaptación de la metodología XP (eXtreme Programming), ya que al ser una metodología ligera de desarrollo de software permite reducir la documentación en comparación a una metodología tradicional, aumentar la productividad en el proceso de desarrollo y la reutilización del código desarrollado. [BECK 1999]

Debido a que el principal objetivo de este proyecto es la implementación de un algoritmo, la metodología XP se ajusta muy bien dado que se centra más en la implementación que en la documentación.

Fases del XP

- Planificación :

En esta etapa se realizó la planificación de las entregas que se realizaron y se planteó los alcances del proyecto.

Se presentó el siguiente documento:
- Historias de usuario: Este documento contendrá los requerimientos del proyecto.

- Diseño

En esta etapa se planteó el diseño que tendrá la herramienta a desarrollar.

Se presentaron los siguientes documentos:

- Glosario de términos: Este documento contiene las especificaciones de los métodos y clases que se usarán en este proyecto.

- Prototipo de interfaz de usuario: Este documento contiene el diseño de interfaz que se utilizó en el proyecto.

- Prototipo de base de datos: Este documento contiene la descripción y detalle de la base de datos que se usó en el proyecto.

-Arquitectura de la herramienta: Este documento contiene el diseño de la arquitectura usado en el proyecto.

- Desarrollo

Esta es la etapa más importante tanto en la metodología como en el proyecto ya que se busca implementar el algoritmo GRASP y la optimización.

Se presentará el siguiente documento:

-Estándar de programación: Este documento contendrá los detalles de los estándares de programación que se usarán en el proyecto.

- Pruebas

En esta etapa se desarrollan las pruebas necesarias a la solución que se desarrollará, al ser un sistema inteligente las pruebas se van a enfocar en la eficiencia del algoritmo y la calibración de las constantes de relajación.
Se presentará el siguiente documento:

- Pruebas de optimización: en este documento se detallará las pruebas que se realizaron para calibrar la constante de relajación y las pruebas de rendimiento y tiempo de ejecución del algoritmo.

Todos los documentos descritos previamente se pueden visualizar en el anexo.

ii. Metodología del algoritmo GRASP

La metodología GRASP es un algoritmo meta heurístico, el cual permite generar varias soluciones diferentes y cercanas al óptimo global gracias al componente aleatorio que posee [GANOZA, SOLANO 2004].

Dentro de esta metodología se dará también la calibración de la constante de relajación del algoritmo GRASP construcción que será implementado (α).

Ya que se el presente proyecto de fin de carrera consiste en adaptar un algoritmo GRASP que resuelva el problema de la distribución de citas médicas en hospitales, será necesario usar la metodología correspondiente a la implementación de dicho algoritmo.

iii. Análisis de la varianza (ANOVA)

El método del Análisis de la Varianza (ANOVA) es una de las técnicas más utilizadas en el análisis de datos en diseños experimentales. Esta técnica permite contrastar más de dos medias, por lo que puede definirse como una generalización de la prueba t para diferencias de dos o más medias.

Para el uso de este método deben considerarse tres requisitos previos:

- Independencia de los datos: Se debe de asegurar que todas las muestras de datos son independientes entre sí.

- Normalidad de los datos: Ésta puede ser contrastada mediante un test apropiado, como por ejemplo el de Kolmogorov – Smirnov.

- Homocedasticidad: Ésta puede ser contrastada mediante un test apropiado, como por ejemplo el de la F o el de Levene.

Debido a que es necesaria una experimentación numérica que permita verificar que el modelo GRASP planteado es mejor que el modelo Voraz actualmente utilizado, se seleccionará este método ya que permite realizar una prueba de medias entre dos o más grupos de datos, además, permite una generalización de la prueba T-Student, lo cual reduce la probabilidad de error tipo I (Rechazar una hipótesis nula aún cuando es cierta).

iv. Método de Comparación de Múltiples Tratamientos

El método de comparación de múltiples tratamientos es utilizado para comparar las diferencias entre dos o más poblaciones. Con este método se podrán resolver preguntas como ¿Qué diferencia significativa existen entre las medias de los tratamientos? o ¿Qué poblaciones son diferentes al resto?

Para poder resolver estas preguntas, primero será necesario determinar si existen diferencias significativas entre las poblaciones, esto se podrá resolver haciendo uso de la técnica de Análisis de Varianza (ANOVA). Con estos resultados, se procede a identificar cuáles son los tratamientos diferentes y, finalmente, realizar la comparación de medias para determinar qué tratamiento posee una media mayor.

5.3 Herramientas y Metodologías del Proyecto

La metodología PMBOK se usará para la gestión de integración del proyecto, dado que esta metodología es ampliamente aceptada por el estándar de proyectos. Esta metodología cuenta con nueve áreas de conocimiento distribuidos en cinco grupos de proceso, sin embargo en el presente proyecto solo se utilizarán las siguientes áreas:

Gestión de la Integración del Proyecto

En esta área se incluyen los procesos y actividades necesarias para identificar, definir, combinar, unificar y coordinar los diversos procesos y actividades de la dirección del proyecto. Esta área implica tomar decisiones en cuanto a la

asignación de recursos, balancear objetivos y manejar las interdependencias entre las áreas de conocimiento de la dirección de proyectos. [PMBOK 2009]

Se tomará en cuenta los siguientes principios:

- Dirigir y Gestionar la Ejecución del Proyecto
- Monitorear y Controlar el Trabajo del Proyecto
- Realizar el Control Integrado de Cambios
- Cerrar Proyecto o Fase

Gestión del Alcance del Proyecto

Esta área incluye los procesos necesarios para garantizar que el proyecto incluya todo el trabajo requerido para completarlo con éxito. [PMBOK 2009]

Se tomará en cuenta los siguientes principios:

- Recopilar requisitos
- Definir el Alcance
- Verificar el Alcance

Gestión del Tiempo del Proyecto

Esta área inclúyelos procesos requeridos para administrar la finalización del proyecto a tiempo. [PMBOK 2009]

Se tomará en cuenta los siguientes principios:

- Definir las Actividades

- Estimar la Duración de las Actividades
- Desarrollar el Cronograma
- Controlar el Cronograma

Gestión de los Riesgos del Proyecto

Esta área incluye los procesos relacionados con llevar a cabo la planificación de la gestión, la identificación, el análisis, la planificación de respuesta a los riesgos, así como su monitoreo y control en un proyecto. [PMBOK, 2009]

Se tomará en cuenta los siguientes principios:

- Planificar la Gestión de Riesgos
- Identificar los Riesgos
- Planificar la Respuesta a los Riesgos

6. Alcance

Este proyecto se basó en el desarrollo de un algoritmo GRASP con doble relajación para la asignación de citas médicas entre pacientes y doctores en hospitales cuyo horario de atención médica es variable en el tiempo. Para el trabajo se tomó en consideración cada uno de los factores del paciente seleccionados inicialmente, tales como la edad, ingreso económico mensual, carga por enfermedad diagnosticada y carga familiar; y los de cada médico: experiencia, disponibilidad de turnos, cantidad de turnos de trabajo y experiencia.

Además, debido a que era necesario obtener el mejor paciente (prioridad de atención) y al mejor doctor (según los factores antes mencionados), se usó el criterio de doble relajación para obtener cada uno de estos.

El algoritmo desarrollado cuenta con una fase de construcción y una de mejoría. Para la fase de construcción, se seleccionaron los valores de las constantes de relajación en a una tabla de distribución de probabilidad uniforme y discreta. Además, por teoría, este algoritmo contará con 10,000 iteraciones; sin embargo, esta cantidad puede verse modificada dependiendo de los resultados que se obtengan de la experimentación numérica.

6.1 Limitaciones

Entre las limitaciones del proyecto se pueden indicar:

- La principal limitación que se puede identificar en el presente proyecto de fin carrera es el hecho que el resultado que brindará el algoritmo GRASP no necesariamente será un valor óptimo. Esto se debe que dicho algoritmo presenta cierto grado de aleatoriedad y por tanto no explora todas las posibles soluciones, sino solo un grupo de ellas.

- Los datos que serán utilizados como datos de entrada deberán seguir un formato y estructura específicos.

- El tiempo de ejecución puede verse modificado dependiendo de las características del hardware y software de la computadora en la que se ejecute el proyecto. Esto a causa de la complejidad de los algoritmos y el número de iteraciones que realizará para obtener resultados.

6.2 Riesgos

Riesgo identificado	Impacto en el proyecto	Medidas correctivas para mitigar
Mala planificación del proyecto	Rechazo de entregables. No realizar los entregables. Entregables realizados a destiempo.	Revisar y cumplir con los plazos de tiempo planteados para cada tarea del proyecto.
Pérdida parcial y/o total de avances del proyecto.	No presentar entregables. Retraso en el avance del proyecto.	Realizar backups periódicos de cada uno de los entregables y los avances respectivos.

Riesgo identificado	Impacto en el proyecto	Medidas correctivas para mitigar
	Fracaso del proyecto.	
Rechazo de trabajadores para brindar información sobre el proceso de entrega de citas médicas.	Retraso en el modelado de proceso.	Buscar información a través de la página web del Ministerio. Persistir en la obtención de información.

Tabla 1.3 Riesgos/Impacto/Medidas correctivas
Fuente: Elaboración propia

7. Justificativa y viabilidad del proyecto

A continuación se presenta la justificativa y el análisis de viabilidad del presente proyecto.

7.1 Justificativa

Como se comentó en la problemática del presente proyecto, uno de los principales problemas de los centros médicos públicos del sector salud en el Perú es la distribución de citas médicas, las cuales debido a su gran demanda generan altos tiempos de espera para los pacientes que las requieran.

Debido a la gran cantidad de pacientes que son atendidos a diario será necesaria la implementación de una solución informática. Asimismo, luego del análisis del problema, se llegó a la conclusión de que la complejidad del mismo es de tipo NP, esto quiere decir que el uso de un algoritmo exacto en la solución tendría un tiempo de respuesta alto, por lo tanto, el desarrollo tendrá que basarse en un algoritmo de aproximación que permita obtener una buena solución en un tiempo aceptable. Ésta permitirá distribuir las citas médicas a los pacientes tomando en cuenta diversos factores tales como edad, historial médico, enfermedad diagnosticada, entre otros.

Con esta implementación se buscará beneficiar a los asegurados, brindándoles una mejor atención médica en un tiempo acorde a sus necesidades; al Estado, priorizando a aquellos pacientes cuyas enfermedades, si no son atendidas a tiempo, podrían agravarse, y por ende, incrementar el costo de su tratamiento; y al

hospital, incrementando su nivel de confianza en los asegurados y reduciendo los conflictos internos que puedan generarse por la entrega de citas médicas.

Finalmente, este proyecto podría ser integrado a un sistema más amplio que permita una gestión completa de todas las demás áreas de un centro médico.

7.2 Viabilidad

Para analizar la viabilidad de este proyecto se tomarán en cuenta los siguientes aspectos:

- Viabilidad Técnica
- Restricciones de Tiempo

7.2.1 Viabilidad Técnica

Para el desarrollo del proyecto, luego de un análisis técnico de las diversas tecnologías que incluyeron Java, Python, C# y C++ se ha optado por utilizar Java como lenguaje de programación debido a las características que posee éste: multiplataforma, gran cantidad de documentación existente, buenas librerías gráficas y eficiencia en el manejo de memoria. Además, cabe resaltar el buen dominio sobre el lenguaje y cada una de las librerías que posee. [ORACLE 2014]

Basado en el lenguaje seleccionado, Java, se optó por utilizar el entorno de desarrollo integrado (IDE) NetBeans debido a la excelente compatibilidad y funcionalidades que posee, tales como autocompletado, debug de código, entre otros, las cuales permitirán que el desarrollo del proyecto se realice de forma rápida y sencilla. Como una ventaja adicional, Netbeans, por su condición de software libre, no tendrá gastos de licenciamiento. [NETBEANS 2014]

Para la experimentación numérica, se utilizará Microsoft Excel debido a la facilidad para mostrar los resultados obtenidos del experimento en modo gráfico. [MICROSOFT 2014]

Finalmente, el equipo de cómputo necesario para poder realizar las labores de análisis, diseño, construcción y pruebas, será un equipo portátil Dual Core de 2.8GHZ y 4GB de memoria RAM. Se considera que las cualidades de éste son las ideales para poder realizar las tareas ya mencionadas.

7.2.2 Restricciones de costo y tiempo

- **Restricciones de costo**

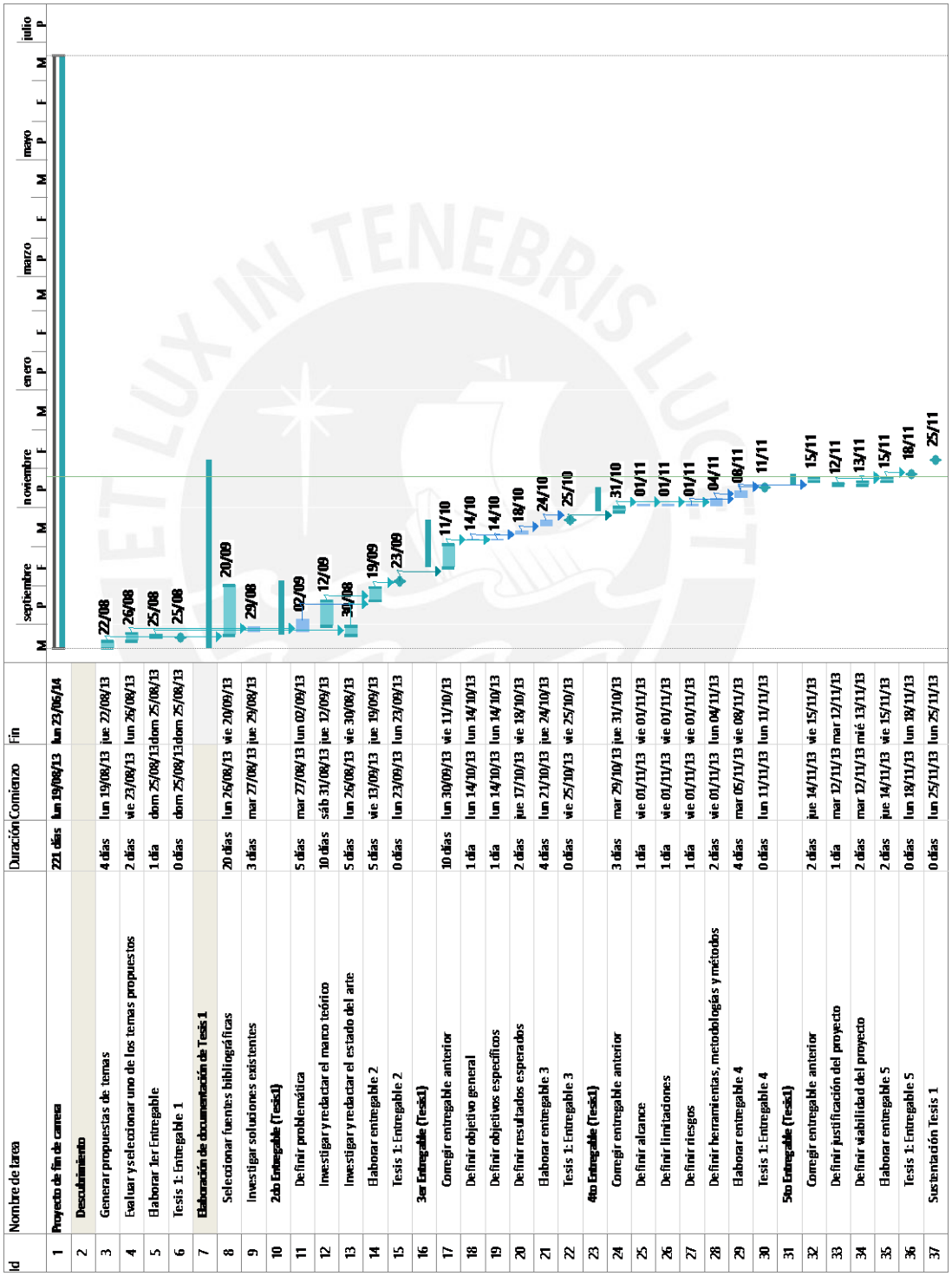
La restricción principal que existe para el desarrollo de este proyecto es utilizar herramientas de software libre para evitar algún costo relacionado adicional, ya que no se maneja un presupuesto para cubrir los costos de programas que requieran licencias. En aquellas herramientas que requieran una licencia de pago, se utilizarán aquellas disponibles en los laboratorios de la facultad.

- **Restricciones de tiempo**

Las restricciones de tiempo estarán dadas en base a la planificación inicial del proyecto, tal y como puede observarse en la Figura 1.3. Además, se considerarán márgenes de tiempo para cada tarea planificada de tal forma que de existir algún tipo de inconveniente, éste no afecte la planificación del proyecto.

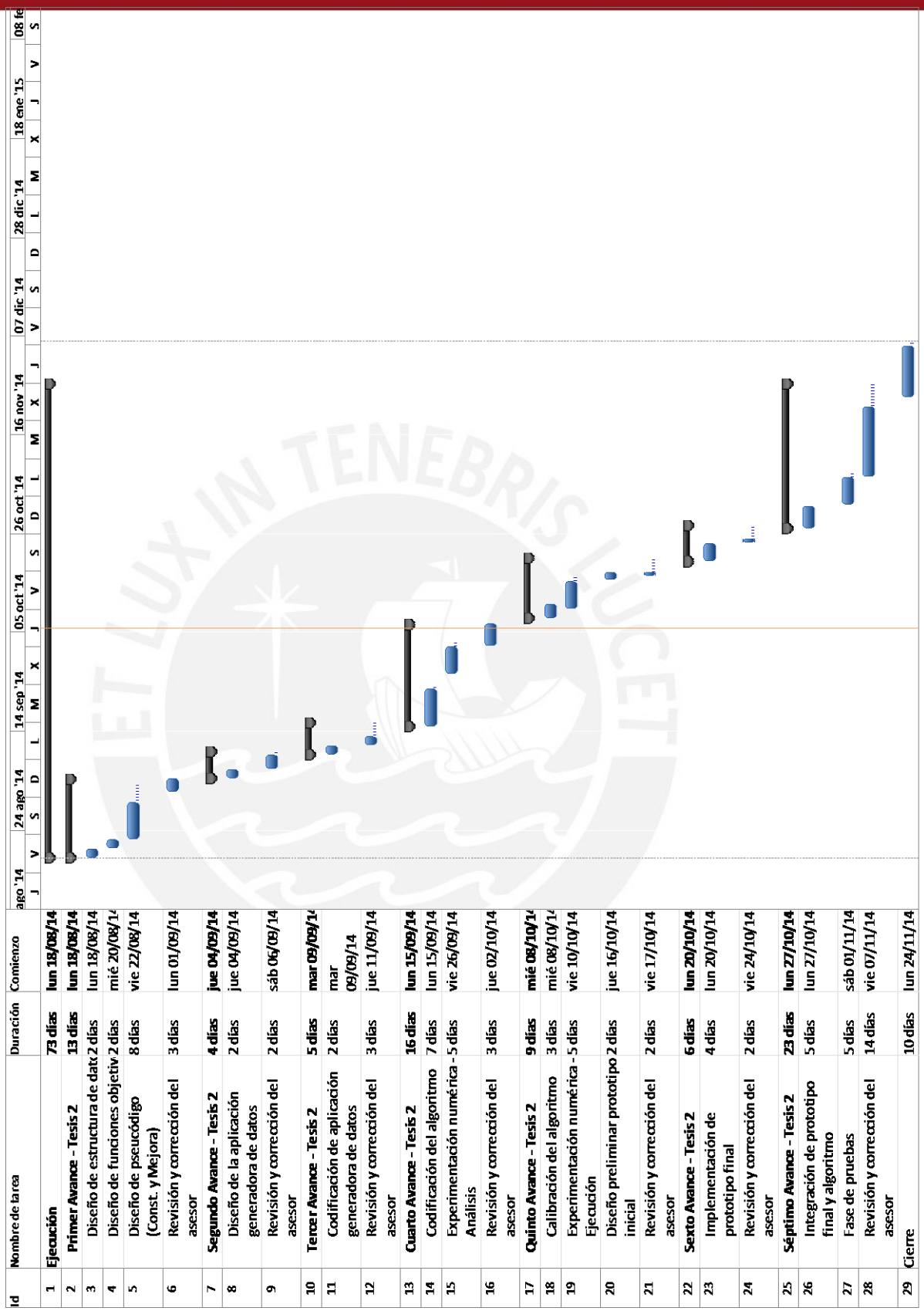
7.2.3 Gantt del proyecto

A continuación, se presenta la planificación del presente proyecto. Como se puede observar, la Figura 1.3 grafica el planeamiento realizado para la primera parte del proyecto, mientras que la Figura 1.4 hace referencia a la segunda parte del mismo.



Id	Nombre de tarea	Duración	Comienzo	Fin
1	**Proyecto de fin de carrera**	**221 días**	**lun 19/08/13**	**lun 23/06/14**
2	**Descubrimiento**			
3	Generar propuestas de temas	4 días	lun 19/08/13	jue 22/08/13
4	Evaluar y seleccionar uno de los temas propuestos	2 días	vie 23/08/13	lun 26/08/13
5	Elaborar 1er Entregable	1 día	dom 25/08/13	dom 25/08/13
6	Tesis 1: Entregable 1	0 días	dom 25/08/13	dom 25/08/13
7	**Elaboración de documentación de Tesis 1**			
8	Seleccionar fuentes bibliográficas	20 días	lun 26/08/13	vie 20/09/13
9	Investigar soluciones existentes	3 días	mar 27/08/13	jue 29/08/13
10	**2do Entregable (Tesis1)**			
11	Definir problemática	5 días	mar 27/08/13	lun 02/09/13
12	Investigar y redactar el marco teórico	10 días	sáb 31/08/13	jue 12/09/13
13	Investigar y redactar el estado del arte	5 días	lun 26/08/13	vie 30/08/13
14	Elaborar entregable 2	5 días	vie 13/09/13	jue 19/09/13
15	Tesis 1: Entregable 2	0 días	lun 23/09/13	lun 23/09/13
16	**3er Entregable (Tesis1)**			
17	Corregir entregable anterior	10 días	lun 30/09/13	vie 11/10/13
18	Definir objetivo general	1 día	lun 14/10/13	lun 14/10/13
19	Definir objetivos específicos	1 día	lun 14/10/13	lun 14/10/13
20	Definir resultados esperados	2 días	jue 17/10/13	vie 18/10/13
21	Elaborar entregable 3	4 días	lun 21/10/13	jue 24/10/13
22	Tesis 1: Entregable 3	0 días	vie 25/10/13	vie 25/10/13
23	**4to Entregable (Tesis1)**			
24	Corregir entregable anterior	3 días	mar 29/10/13	jue 31/10/13
25	Definir alcance	1 día	vie 01/11/13	vie 01/11/13
26	Definir limitaciones	1 día	vie 01/11/13	vie 01/11/13
27	Definir riesgos	1 día	vie 01/11/13	vie 01/11/13
28	Definir herramientas, metodologías y métodos	2 días	vie 01/11/13	lun 04/11/13
29	Elaborar entregable 4	4 días	mar 05/11/13	vie 08/11/13
30	Tesis 1: Entregable 4	0 días	lun 11/11/13	lun 11/11/13
31	**5to Entregable (Tesis1)**			
32	Corregir entregable anterior	2 días	jue 14/11/13	vie 15/11/13
33	Definir justificación del proyecto	1 día	mar 12/11/13	mar 12/11/13
34	Definir viabilidad del proyecto	2 días	mar 12/11/13	mié 13/11/13
35	Elaborar entregable 5	2 días	jue 14/11/13	vie 15/11/13
36	Tesis 1: Entregable 5	0 días	lun 18/11/13	lun 18/11/13
37	Sustentación Tesis 1	0 días	lun 25/11/13	lun 25/11/13

Figura 1.3 Gantt del proyecto – 1er Parte
Fuente: Elaboración propia



Id	Nombre de tarea	Duración	Comienzo
1	**Ejecución**	**73 días**	**lun 18/08/14**
2	**Primer Avance - Tesis 2**	**13 días**	**lun 18/08/14**
3	Diseño de estructura de datc	2 días	lun 18/08/14
4	Diseño de funciones objetiv	2 días	mié 20/08/14
5	Diseño de pseucódigo (Const. y Mejora)	8 días	vie 22/08/14
6	Revisión y corrección del asesor	3 días	lun 01/09/14
7	**Segundo Avance - Tesis 2**	**4 días**	**jue 04/09/14**
8	Diseño de la aplicación generadora de datos	2 días	jue 04/09/14
9	Revisión y corrección del asesor	2 días	sáb 06/09/14
10	**Tercer Avance - Tesis 2**	**5 días**	**mar 09/09/14**
11	Codificación de aplicación generadora de datos	2 días	mar 09/09/14
12	Revisión y corrección del asesor	3 días	jue 11/09/14
13	**Cuarto Avance - Tesis 2**	**16 días**	**lun 15/09/14**
14	Codificación del algoritmo	7 días	lun 15/09/14
15	Experimentación numérica - Análisis	5 días	vie 26/09/14
16	Revisión y corrección del asesor	3 días	jue 02/10/14
17	**Quinto Avance - Tesis 2**	**9 días**	**mié 08/10/14**
18	Calibración del algoritmo	3 días	mié 08/10/14
19	Experimentación numérica - Ejecución	5 días	vie 10/10/14
20	Diseño preliminar prototipo inicial	2 días	jue 16/10/14
21	Revisión y corrección del asesor	2 días	vie 17/10/14
22	**Sexto Avance - Tesis 2**	**6 días**	**lun 20/10/14**
23	Implementación de prototipo final	4 días	lun 20/10/14
24	Revisión y corrección del asesor	2 días	vie 24/10/14
25	**Séptimo Avance - Tesis 2**	**23 días**	**lun 27/10/14**
26	Integración de prototipo final y algoritmo	5 días	lun 27/10/14
27	Fase de pruebas	5 días	sáb 01/11/14
28	Revisión y corrección del asesor	14 días	vie 07/11/14
29	Cierre	10 días	lun 24/11/14

Figura 1.4 Gantt del proyecto – 2da Parte
Fuente: Elaboración propia

CAPÍTULO 2: MARCO CONCEPTUAL

En el presente apartado, se presentarán los conceptos necesarios para el completo entendimiento y análisis del problema, así como los conceptos necesarios para entender la solución planteada.

1. Conceptos aplicados a la problemática

A continuación, se presentan los conceptos relacionados al problema y su contexto.

1.1 Citas médicas

Una cita médica es un espacio de tiempo que reserva un paciente en un centro de salud, con la finalidad de tener una conversación con algún médico del establecimiento acerca de sus dolencias y recibir, con prontitud, un tratamiento. Una cita médica tiene como objetivo encontrar la razón de la dolencia del paciente [MINSA2014].

Existen dos tipos de citas médicas: cita médica a domicilio y cita médica interna. La cita médica a domicilio consiste en que el médico se apersona al lugar de residencia del paciente, quien por motivos de su dolencia no puede movilizarse a un centro médico. Por otro lado, existe la cita médica interna que es la cita común, en la que el paciente se acerca al hospital o clínica [MINSA 2014].

1.2 Carga por enfermedad

Según el Ministerio de Salud, la carga por enfermedad puede ser definida como la medida de pérdidas de salud que para una población representa las consecuencias mortales y no mortales de las diferentes enfermedades y lesiones. La carga de enfermedad atribuible a una enfermedad concreta se mide por un lado con su frecuencia y, por otro lado, a partir de las consecuencias mortales y efectos que generan discapacidad [MINSA 2014].

El estudio de la carga por enfermedad es importante ya que puede ser utilizado para los siguientes aspectos [OMS 2014]:

- Medir y comparar la salud de poblaciones o grupos sociales

- Conocer la evolución de la salud de una población o la magnitud de un problema de salud a través del tiempo

- Medir y comparar la importancia de los diferentes problemas de salud de una población de un momento dado.

- Utilizar estos resultados como un instrumento más para la definición de prioridades en salud y orientar la asignación de recursos.

Siguiendo el sistema de clasificación de la Organización Mundial de la Salud (OMS), se han creado 21 categorías para agrupar a todos los problemas de salud, de cualquier tipo como son enfermedades infecciosas, enfermedades no transmisibles y aquellas causadas por accidentes o lesiones [OMS 2014].

Para cada categoría se ha obtenido, mediante un estudio y análisis previo, una cantidad de años de vida saludable perdidos (AVISA). Esta cantidad toma en consideración cuántos años menos deja de vivir una persona en relación a la esperanza de vida humana, y también incluye cuántos años ha vivido esta persona en estado de discapacidad física a causa de alguna enfermedad o lesión. En resumen, AVISA es la suma de los años de vida en discapacidad (AVD) y los años de vida perdidos (AVP) [OMS 2014].

En la Figura 2.1, se muestra cada categoría con sus AVD y AVP respectivos.

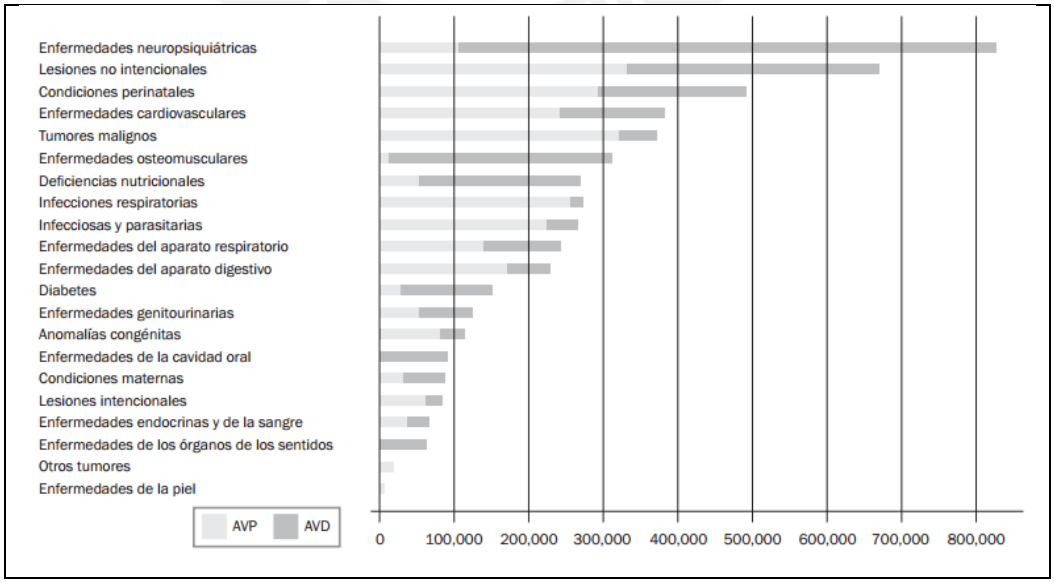


Figura 2.1 Carga por enfermedad / Categoría de enfermedad

[MINSA 2014]

2. Conceptos aplicados a la solución al problema

A continuación, se presentan los conceptos relacionados a la solución del problema.

2.1 Problemas NP

Los problemas NP son una clase de problemas de decisión que pueden ser resueltos por una máquina de Touring no determinista en tiempo polinómico. Ejemplos de este problema son el Timetabling, el problema de scheduling, el problema del agente viajero, etc. Los problemas NP fueron planteados inicialmente por Stephen Cook en un artículo titulado “The complexity of theorem-proving procedures” en el año 1971. [COOK 1971]

Según Cook, los problemas NP-completos son problemas que no pueden ser resueltos en tiempo polinomial, los cuales, desde el punto de vista computacional, no poseen una solución algorítmica eficiente. Por lo tanto, para resolver este tipo de problemas, es necesario el uso de aproximaciones, como los que usan los métodos heurísticos y metaheurísticos. Este tipo de problemas tiene las siguientes propiedades: [COOK 1971] [GAREY, JOHNSON 1979]

a) Ningún problema NP-completo puede ser resuelto por algún algoritmo o solución polinómica, en su defecto, se encontrarán soluciones algorítmicas con tiempo exponencial.

b) Si existe alguna solución polinómica para resolver algún problema NP-completo, entonces hay una solución polinómica para todos los problemas NP-completos.

2.2 El problema de asignación de citas médicas

El problema de asignación es uno de los problemas de optimización combinatoria fundamentales en la rama de la optimización de las operaciones y se puede definir como el proceso de asignar, de forma óptima, “n” agentes a “m” tareas, de tal forma que el costo de este emparejamiento sea mínimo.

Para el caso particular de la asignación de citas médicas, se deberá asignar un turno (cita médica) a uno de los pacientes tomando en cuenta la disponibilidad de horarios que posee cada médico, lo que es considerado como una limitación debido a regulaciones laborales. Dentro del problema también deberá considerarse el nivel de urgencia de atención para cada paciente, tomando en cuenta factores que permitan establecer un nivel de prioridad en la atención requerida. Por último, la solución final a la asignación de citas médicas deberá buscar minimizar el tiempo promedio de atención de cada paciente.

2.3 Heurística

Se puede definir a "heurística" como la metodología inteligente que se usa para poder llevar a cabo una tarea, que no es producto de un análisis sino del conocimiento y experiencia que se tiene del tema. La popularidad de la heurística se debe principalmente al matemático George Pólya y su libro "How to solve it". En una definición más simple, la heurística es un proceso de búsqueda para obtener resultados de problemas NP mediante la obtención de una solución no exacta. [POLYA 1965] [MOSCATO 1996]

- Buscar un problema similar previamente resuelto.

- Determinar la técnica que se aplicará para obtener una solución.
- En el caso en el que sea posible, utilizar la técnica y solución descrita en el punto anterior para resolver el problema planteado.

Existen dos interpretaciones posibles para el término heurística. La primera de ellas concibe las heurísticas como un procedimiento para resolver problemas. Para esta interpretación, las definiciones más interesantes que se extraen de la literatura son las siguientes: [POLYA 1965]

- Como dice H. Müller-Merbach en [MULLER 1991]: "En investigación operativa, el término heurístico normalmente se entiende en el sentido de un algoritmo iterativo que no converge hacia la solución más óptima del problema".

- Como dice Barr en [BAAR et al, 1999]: “Un método heurístico es un conjunto bien conocido de pasos para identificar rápidamente una solución de alta calidad para un problema dado”

La segunda interpretación de heurística entiende que éstas son una función que permiten evaluar las características de estado o solución. Algunas de las definiciones que se encuentran en la literatura son: [DUARTE 2007]

- Como dice Rich en [RICH et al, 1990]: “Una función heurística es una correspondencia entre las descripciones de los estados del problema hacia alguna medida de idoneidad, normalmente representada por números. Los aspectos del problema que se consideran, cómo se evalúan estos aspectos y los pesos que se dan a los aspectos individuales, se eligen de forma que el valor que la función da a un nodo del proceso de búsqueda sea una estimación tan buena como sea posible para ver si ese nodo pertenece a la ruta que conduce a la solución”.

- Como dice Russel en [RUSSELL et al, 2009]: “Actualmente, el término heurística se utiliza más bien como adjetivo para referirse a cualquier técnica que permita mejorar el desempeño del caso promedio en una tarea de resolución de problemas, aunque no necesariamente permita mejorar el desempeño del peor de los casos. Específicamente en el área de los algoritmos de búsqueda, se refiere a una función mediante la cual se obtiene una estimación del coste de una solución”.

Los algoritmos heurísticos no poseen ningún mecanismo que les permita escapar de los óptimos locales. Para solventar este problema, se introducen otros algoritmos de búsqueda más inteligentes que eviten, en la medida de lo posible, quedar atrapados en óptimos locales. Estos algoritmos de búsqueda más inteligentes, denominados metaheurísticas, son procedimientos de alto nivel que guían a algoritmos heurísticos conocidos evitando que éstos caigan en óptimos locales. [DUARTE 2007]

2.4 Meta heurísticas

El término metaheurística fue introducido por F. Glover en el año 1989, en su publicación “Taboo Search”. Con este término, pretendía definir un método de alto

nivel que guía y modifica otras heurísticas, y así poder explorar y hallar soluciones globales. [GLOVER 1989]

A partir de la definición original de F. Glover, en diferentes publicaciones se pueden encontrar diversas opiniones y puntos de vistas respecto a la metaheurística:

- Como dice Kelly en [KELLY,et al. 1996]: “Las metaheurísticas son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria, en los que los heurísticos clásicos no son efectivos. Las metaheurísticas proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y los procedimientos estadísticos”.

- Como dice Voss en [VOSS,et al 1999]: “Una metaheurística es un proceso iterativo maestro que guía y modifica las operaciones de una heurística subordinada para producir eficientemente soluciones de alta calidad. Las metaheurísticas pueden manipular una única solución completa (o incompleta) o una colección de soluciones en cada iteración. La heurística subordinada puede ser un procedimiento de alto (o bajo) nivel, una búsqueda local, o un método constructivo”.

La idea básica es enriquecer a los algoritmos heurísticos de forma que éstos no se queden atrapados en óptimos locales. La evolución de las meta heurísticas durante estos últimos 25 años ha tenido un comportamiento exponencial. Durante todo este tiempo de estudio e investigación de las metodologías heurística, se han encontrado soluciones de muy alta calidad a problemas que parecían difíciles de abordar. [GLOVER, LAGUNA 2007]

2.5 Algoritmo voraz

Los algoritmos golosos-miopes o voraces reciben esta denominación por las siguientes razones: [TUPIA 2004]

- Es goloso o voraz porque siempre escoge al mejor candidato para formar parte de la solución: aquel que tenga mejor valor de la función objetivo, lo que constituye el cumplimiento de cierto criterio goloso de selección.

- Es miope porque esta selección es única e inmodificable dado que no analiza más allá los efectos de haber seleccionado un elemento como parte de la solución. No deshacen una selección ya realizada: una vez incorporado un elemento a la solución, ésta permanece hasta el final y cada vez que un candidato es rechazado, lo es permanentemente.

El pseudocódigo del algoritmo voraz en su forma más general es el siguiente:

```
Algoritmo Voraz General(N,c,S,E,F)
Inicio
 0. Leer N, c, E, F
 1. Ordenar bajo algún criterio, los elementos de
E de acuerdo al valor de c
 2. S:= f
 3. Para i: 1 a N, hacer
 3.1.Si S∪ È es una solución viable entonces S = S ∪ E
Fin Voraz General
```

Figura 2.2 Pseudocódigo Algoritmo voraz
[TUPIA 2004]

2.6 Algoritmo GRASP

El algoritmo GRASP es una técnica metaheurística desarrollada por Thomas Feo y Mauricio Resende a mediados de 1989 con el propósito de resolver el problema del cubrimiento de conjuntos. El éxito de este algoritmo es debido a su gran acercamiento y rapidez de convergencia, al encontrar soluciones muy cercanas al óptimo global.

La siguiente figura muestra la estructura básica de un algoritmo GRASP: [FEO, RESENDE 1995]

```
Procedimiento GRASP (Instancia del Problema)
  1. Leer (Instancia)
  2. Mientras (no se cumpla condición de parada) hacer
       a. Procedimiento Construcción (S)
       b. Procedimiento Mejoría (S)
  3. Fin Mientras
  4. Retornar (Mejor S)
Fin GRASP
```

Figura 2.3 Pseudocódigo del algoritmo GRASP
[TUPIA 2004]

- **Fase de Construcción GRASP**

El algoritmo GRASP propone un conjunto de valores candidatos a ser los óptimos. Dicha lista se denomina RCL (Restricted Candidates List). Estos valores serán obtenidos mediante el uso de una función objetivo, la cual mide el aporte de cada elemento a la solución parcial.

El parámetro α empleado se denomina parámetro de relajación y describe el tamaño del segmento RCL. Asignar valores muy pequeños a α puede omitir soluciones potencialmente buenas y, si, por el contrario, el valor es demasiado alto, se pueden agregar soluciones que no contribuyan con una mejor solución.

A continuación, se procede a seleccionar al azar un elemento del RCL ("*a*"). Se determina si el conjunto solución es factible luego de agregarle dicho elemento, y si es afirmativo, entonces se añade "a" al conjunto S, eliminando "a" de los valores de entrada. Finalmente, se prosigue con las demás iteraciones hasta cumplir con la condición de parada del algoritmo [FEO, RESENDE 1995].

Procedimiento GRASP Construcción (N, c, E, F, S, a)
Inicio
Inicializar c, E, a
$S = \emptyset$, N = E
1. Mientras <no se cumpla condición de parada> hacer
Inicio
//Se crea la lista RCL
1.1 $RCL = \{\forall x \epsilon N: \beta \leq c(x) \leq \beta + \alpha(\tau - \beta)\}$
//Se halla el mejor y peor valor de N
1.2 $\beta = Mejor\{c(x): x \epsilon N\}$
1.3 $\tau = Peor\{c(x): x \epsilon N\}$
//Se selecciona un elemento de la lista RCL
1.4 $a = Aleatorio(RCL)$
//Se determina si el conjunto solución es factible luego de
//agregarle el elemento seleccionado anteriormente
1.5 $Si\ S \cup \{a\}\ \epsilon\ F \rightarrow S = S \cup \{a\}$
//Se eliminar el elemento seleccionado de N
1.6 $N = N - \{a\}$
//Se configura la función c para las siguientes iteraciones
1.7 $Adaptar\ c$
Fin Mientras
Fin GRASP Construcción.

Figura 2.4 Pseudocódigo de algoritmo GRASP de construcción
[TUPIA 2004]

- **Fase de Mejoría de la solución GRASP**

En la fase de mejoría de la solución GRASP, se explora de forma repetida los elementos preseleccionados en la fase de construcción en busca de una solución que mejora la actual.

Procedimiento GRASP Mejoría (X)
  Mientras (no es posible mejorar solución) hacer
    Inicio
      //Se forma una vecindad en base a la solución X, en la cual cada //elemento N_k tiene un grado de separación máxim igual a k.
      1. $N_k(X) = \{z_1, z_2, z_3, \dots z_N\}\ tq\ \sum_j^N |z_j - x_j| \leq k, z_j\ \epsilon\ \{0,1\}$
      //Se selecciona el mejor elemento de la vecindad.
      2. $Best_k(N_k) = \{C_i^{i+1} z_i\}$
      //Si se logra un mejor X, se actualiza la solución para que las siguientes //iteraciones se basen en esta nueva encontrada.
      3. $Si\ Best_k(N_k) es\ mejor\ que\ X \rightarrow X = Best_k(N_k)$
  Fin Mientras
  Retornar(X)
Fin GRASP Mejoría

Figura 2.5 Pseudocódigo de algoritmo GRASP de mejoría
[TUPIA 2004]

- **Criterio de doble relajación**

El criterio de la doble relajación es un mecanismo utilizado por el algoritmo GRASP con la finalidad de evaluar soluciones en base a dos funciones objetivos. Debido a ello, es necesario que, dentro de los parámetros del algoritmo, se definan, ya no una, sino dos constantes de relajación y dos lista de candidatos restringidas (RCL), las cuales permitirán evaluar cada una de las funciones objetivo planteadas [RAMIREZ 2006].

2.7 Conclusión

Los conceptos presentados anteriormente permitirán entender y ayudar a resolver el problema de la asignación de citas médicas entre pacientes y doctores. Asimismo, cómo es que el uso de las meta heurísticas y los algoritmos basados en ellas permitirán obtener una solución global al problema en tiempos aceptables.

3. Estado del arte

En el presente apartado, se presentan aquellas soluciones comerciales y académicas encontradas, que guardan cierto grado de relación con el problema propuesto en este proyecto, asimismo, se explicará la metodología utilizada en la revisión y las conclusiones de esta actividad.

3.1 Objetivos

El objetivo del estado del arte es tener un acercamiento a las distintas soluciones a problemas similares en la actualidad. Esto servirá para tener una idea general de los resultados que se han obtenido y los resultados que se esperaban. Además, se puede aprender de aquellas soluciones que funcionaron a la expectativa, y de las que no funcionaron, para estudiar y no cometer los motivos del fracaso.

3.2 Metodología usada en la revisión del estado del arte

La revisión del estado del arte se realizó en base a una revisión de los diversos artículos y revistas indexados en librerías digitales como Scopus, Springer, ACM y Springer.

En primer lugar se procedió a definir los conceptos a investigar: Algoritmo GRASP y el problema de asignación, para luego proceder a la recopilación del material de investigación. Una vez recopilado este material, se seleccionaron aquellas investigaciones cuyo objetivo fuera la optimización al problema de la asignación y que, además, aplicara diversas variedades de restricciones. Por otro lado, se encontraron investigaciones que resolvían el problema de la asignación a través de la investigación operativa, pero en un contexto mucho más sencillo al presente en las investigaciones sobre algoritmos inteligentes.

3.3 Soluciones comerciales

- **Galenus Pro**

Galenus Pro es un software altamente especializado, para la gestión integral avanzada de consultas y clínicas. Dentro de sus funcionalidades destacan: coordinación de citas, consultas, historiales del paciente, datos del paciente.

Además, al estar basado en tecnología Cloud Computing, posee un módulo de solicitud de citas y de agenda médica web. [GALENUS 2014]

GalenusPro está dirigido a facultativos e instituciones sanitarias, que quieren adaptar su gestión clínica y administrativa a las necesidades actuales de control y normativa existente.

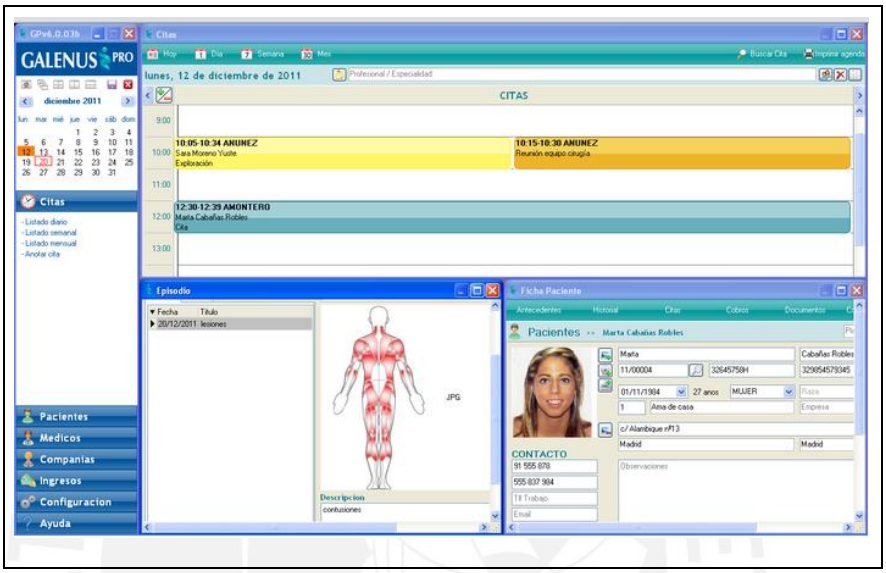


Figura 2.6 Interfaz gráfica principal – Galenus Pro
[GALENUS 2014]

- **DriCloud**

DriCloud es un software integral que permitirá gestionar historias clínicas, consultas, quirófanos, pruebas médicas, lista de espera, contabilidad, stock, vacaciones, estadísticas, gráficas, casos clínicos de interés científico, tiempos de espera de los pacientes fuera y dentro de la consulta, gestión de imágenes, documentos, facturas, etc.

- **Sistema de reserva de citas médicas MINSA**

El servicio de reservas de citas por Internet del Ministerio de Salud, permitirá al paciente separar su cita ingresando a su sitio web, disminuyendo así, los tiempos de espera y evitando las largas colas en los hospitales. Además, el servicio cuenta con un sistema de confirmación de cita a través de correo electrónico y/o vía mensaje de texto al celular del usuario. [MINSA 2014]

Por ahora el programa solo podrá ser usado en tres hospitales nacionales, en los Institutos de Enfermedades Neoplásicas, el Instituto Materno Perinatal y el Hospital San Bartolomé, aunque a lo largo del año más hospitales serán incluidos en el programa de reserva de citas médicas por Internet.

- **GPT**

Es una solución específica para la Gestión y Planificación de Turnos para Hospitales que asegura en todo momento la mejor adecuación entre la demanda de presencia de profesionales y el cumplimiento de las normas que regulan los tiempos de trabajo y descanso del personal. La solución se compone de dos módulos básicos, Planificación y Gestión. En el área de la planificación, GPT permite la definición del organigrama de Unidades Funcionales de planificación y de la normativa vigente mediante patrones de turnos. Algunas características: [GPT 2014]

- Planificación automática con potente capacidad de cálculo para la resolución de problemas complejos
- Rápido re-cálculo ante incidencias que pudieran alterar el plan de turnos previsto
- Interfaz gráfica intuitiva que facilita el registro de intercambios de turnos y la realización de desprogramaciones manuales
- Permite desprogramaciones manuales o automáticas en caso de incidencias.
- Permite obtener completos informes de programación de la jornada mensual, de la actividad laboral anual, notas simples e incidencias en múltiples formatos (MS Excel, PDF, XML).
- GPT es una aplicación basada en web y puede funcionar de modo independiente o integrada con otras aplicaciones informáticas de gestión de RR.HH y Nómina.

- **SP-Expert**

Este sistema libera a los gestores responsables de las tareas rutinarias tales como la verificación de las normas del convenio laboral o de la observancia de las reglas de los límites de tiempo de trabajo. Permite generar informes, Web Reporting, información actualizada (por ejemplo, detalles de las vacaciones pendientes), datos históricos (por ejemplo, los niveles históricos de dotación de personal según demanda).Las siguientes son algunas de las características:

- Diseñada para integrarse con los sistemas subyacentes de nómina, RRHH y control de presencia, así como con sistemas de análisis de la producción y de la demanda.

- Los empleados pueden recuperar, visualizar y registrar sus turnos, introducir sus preferencias, solicitar permisos y peticiones de intercambio de turno (shift swap) y actualizar los registros en la base de datos a través del Web Terminal de SP-EXPERT.

- Integración de los empleados en los procesos mediante workflow: solicitud de permisos, intercambio de turnos y necesidades de formación.

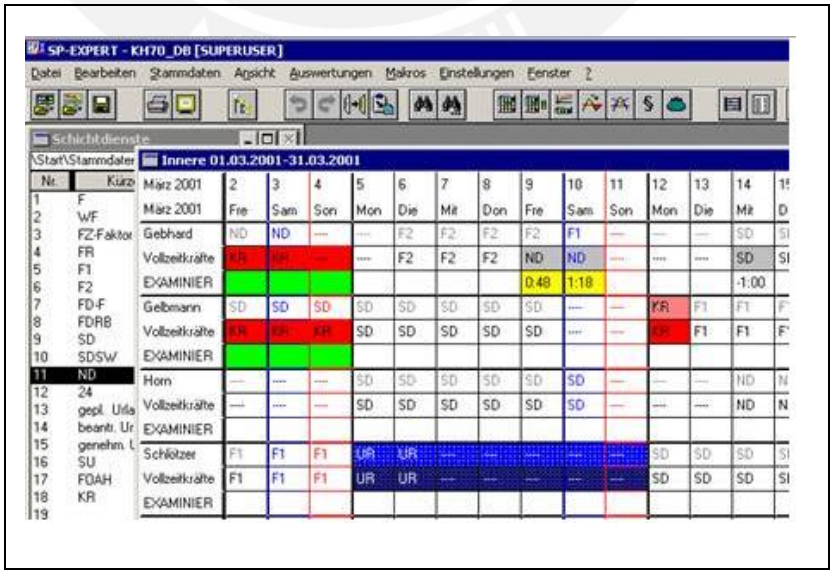


Figura 2.7 Interfaz de planificación de horarios – SP Expert

[SP-EXPERT 2014]

- **Untis**

Untis es un software diseñado para la generación y administración de horarios en centros educativos. Cuenta con diversos módulos como por ejemplo: Planificación de sustituciones, horarios por períodos, horarios por secciones, asignación de clases, etc.

	Valoración	Horas no planif	Huecos	Horas priorit.	Materias 2x/Día	Falta HH dobles
Mejor horario	380	0	1	2	2	8
Horario 2	387	0	1	2	2	10
Horario 3	414	0	1	4	2	9
Horario 4	417	0	1	2	2	9
Horario 5	420	0	1	2	2	10
Horario 6	463	0	1	2	3	9
Horario 7	471	0	1	1	4	10
Horario 8	473	0	1	2	3	12
Horario 9	497	0	1	3	2	14

Figura 2.8 Interfaz de generación de horarios - UNTIS
[UNTIS 2014]

- **Droster**

El sistema puede ser aplicado a cualquier negocio u organización – equipo de enfermería, equipo de médicos, hotelería y hospedaje, restaurantes, instituciones académicas, servicios de delivery, personal de limpieza y mantenimiento, entrenadores, empleados de mesa de ayuda, servicio de transporte, azafatas, agencia de viajes. Este sistema permite organizar asignaciones y reaccionar a cambios en los listados, aplicar reglas a los listados e imprimir cronogramas en una variedad de reportes. No hay límites en el número de turnos que se puede ingresar o en el número de empleados que se puede administrar con él.

Tiene un motor basado en reglas y maneja una cantidad ilimitada de lugares y personas. Los formatos en que puede exportar los reportes son: txt, htm, rtf, xls, tiff,

gif, bmp, jpeg, pdf. Permite también manejar un planificador con vistas Diarias y Horarias que ofrece la segmentación de tiempo necesaria.

3.4 Soluciones basadas en investigaciones

Existen diversos antecedentes con respecto al uso de algoritmos inteligentes para resolver problemas de asignación del personal en una empresa. A continuación se menciona algunos artículos que presentan metodologías basadas en algoritmos:

i. En [FEO, RESENDE 1995], el problema planteado consistía en asignar un costo a la ejecución de cada trabajo. Así el costo ij era el de ejecutar inmediatamente la tarea j después de la tarea i; luego agregaba el tiempo de ejecución d j para cada trabajo y el concepto de penalidad por retrasos en la línea j. De esta manera, se pretendía aplicar las técnicas GRASP para minimizar la suma de las penalidades reduciendo así el tiempo de procesamiento de lote.

ii. En [TUPIA 2004],para el problema del Job Scheduling con máquinas diferentes y tareas independientes, se amplió el criterio voraz de un algoritmo propuesto aplicando las fases convencionales de la técnica GRASP, lo que consiguió mejorar aproximadamente al 10% los resultados del algoritmo voraz para instancias incluso mayores a las 12500 variables, vale decir, 250 tareas por 50 máquinas.

iii. En [RIOS, BARD 2000], se planteó un GRASP con la finalidad de hallar una secuencia de N tareas en un ambiente de flow shop de M máquinas con tiempos de preparación dependientes de la secuencia, que minimice el makespan.

iv. En [ALVAREZ 2002], se aplica un algoritmo de búsqueda tabú para asignar a cada estudiante un grupo de estudio en una universidad de España. Esta investigación se divide en dos partes: el problema del timetabling y la construcción de la solución basada en la búsqueda tabú. El objetivo final de esta investigación fue el de generar horarios de alta calidad y equilibrar la cantidad de alumnos en un curso durante el ciclo universitario.

v. En [WARNER 1972], se propone una metodología, para realizar la asignación óptima del personal de enfermería extra, teniendo cierta prioridad sobre aquellas unidades que lo requieran. La prioridad de las unidades se basa en la percepción que tiene la persona responsable sobre la necesidad de personal adicional en las diferentes unidades de cuidado. Dicha prioridad puede estar basada en la cantidad de pacientes en la unidad, nivel de cuidado de los pacientes, capacidad de cada unidad, etc.

vi. En [LITCHFIELD 2003], se presenta un algoritmo heurístico que utiliza la búsqueda tabú para la organización del tiempo de trabajo semanal del personal de un restaurante. El algoritmo considera diferentes categorías de personal que han de ser asignadas a una serie de horarios de trabajo fijos, la disponibilidad del personal, así como sus preferencias, los requerimientos de la organización y la limitación de sus recursos.

vii. En [DEXTER,et al 2002], se trata el problema de centrar la atención de los servicios quirúrgicos en las áreas prioritarias, para ello estudian en qué medida un cambio en la asignación del tiempo de quirófano de los cirujanos, puede incrementar los costos variables peri operatorios. Los autores utilizan la programación lineal para resolver el problema.

viii. En [PUNNAKITIKASHEM,et al. 2008], trata el problema de la programación de los turnos de las enfermeras. Desarrollan un modelo de programación estocástica, que tiene como objetivo minimizar el exceso de carga de trabajo de las enfermeras. Debido a que la cantidad de atención requerida por un paciente es aleatoria, se consideran varios escenarios. Además, los autores proponen un algoritmo de tipo goloso para resolver el sub problema de los recursos. Los autores demuestran que pueden ahorrar hasta 273 horas de exceso de carga de trabajo por año en cada unidad médico-quirúrgica.

ix. En [LIM, RODRIGUEZ 2003] se aplica un hibrido de la búsqueda tabú e inteligencia artificial para el problema general de asignación de horarios a empleados. Este incorpora la disponibilidad de empleados para días específicos de la semana y horas del día, unión y manejo de la jerarquía de las reglas y las restricciones no homogéneas debido a los diferentes tipos de habilidades y diferentes niveles adquiridos.

x. En [BAAR, et al. 1999], se desarrolla dos versiones de la búsqueda tabú para el problema de gestión de un proyecto con recursos limitados. La primera está basada en una lista de actividades y el esquema serie como procedimiento de decodificación. El vecindario se genera utilizando una de las tres clases de movimientos basados en la ruta crítica. La segunda versión está basado en la así llamada representación de esquema de secuencia y utiliza el mecanismo de generación de vecinos y una regla de decodificación presentada por Brucker. [BRUCKER, et al. 1998]

xi. En [OGULATA,et al. 2008], se propone un modelo matemático jerárquico que permita realizar la programación adecuada de personal en un hospital, para una semana de programación. Los autores dividen el problema en tres etapas: selección de personal, asignación de los pacientes al personal y programación de los pacientes a lo largo de un día. Los objetivos planteados en el modelo jerárquico fueron: maximizar el número de pacientes considerando las limitaciones de capacidad; balance de cargas de trabajo de los fisioterapeutas y la asignación de pacientes entre 31 loa fisioterapeutas de manera equitativa; minimizar el tiempo de espera de los pacientes a través de la programación del tratamiento en un día.

xii. En [LAMIRI 2008], se formula un modelo estocástico para la programación de las salas de operación con dos tipos de pacientes: los pacientes electivos y los pacientes de emergencia, donde el problema de planificación consiste en asignar los casos electivos a diferentes periodos en un horizonte de planificación, a fin de minimizar el costo de los pacientes electivos y los costos de horas extras en los quirófanos

3.5 Conclusiones sobre el estado del arte

En este capítulo se ha hecho la revisión de las soluciones, tanto exactas como aquellas que guardan relación aproximada al problema de la entrega de citas médicas.

Dentro de las investigaciones realizadas sobre el problema del manejo de horarios, sea del rubro médico, comercial o académico, se emplean diversos algoritmos que permiten obtener soluciones óptimas globales. En muchos casos se crean

algoritmos híbridos, mejorados u optimizados para poder generar, en una primera fase una población inicial, y posteriormente una solución definitiva. Además, el uso de restricciones y factores que determinarán el comportamiento del algoritmo es también aplicado a las soluciones mencionadas anteriormente, y es considerado como un punto importante en la obtención de resultados. La gran mayoría de resultados obtenidos de estas investigaciones son favorables: reducción de costos, manejo óptimo de recursos limitados, reducción de tiempos de espera entre procesos, etc.

Sin embargo, se ha podido observar que muchas de las soluciones comerciales diseñadas para la reserva de citas son del tipo cola, esto quiere decir que la entrega de turno de cita médica se realizará en orden de llegada del paciente sin tomar en cuenta factor o prioridad alguna, por lo tanto, siguen en modelo utilizado dentro de las instituciones de salud pública del Perú.

En conclusión, se puede determinar que la solución planteada, a diferencia de las soluciones descritas anteriormente, realiza la entrega de citas tomando en cuenta factores críticos dentro del contexto de la salud, lo que permite una atención enfocada en el estado de salud de los pacientes. Asimismo, la gran mayoría de las soluciones encontradas forman parte de un sistema de información más grande, por lo que el costo de éstas, en comparación a la herramienta propuesta como solución y que, solo se enfoca en la entrega de citas mas no en otras actividades de la gestión de las instituciones de salud, es mucho más elevado.

CAPÍTULO 3: ESTRUCTURAS DE DATOS

Como parte de la solución del problema descrito en los capítulos anteriores, es necesario definir un conjunto de estructuras de datos que permitan dar soporte a la adaptación de los algoritmos seleccionados para este proyecto.

Dentro de estas estructuras podemos distinguir dos grupos, el primero hace referencia a una de las estructuras indispensables del algoritmo Grasp a implementar como es el RCL (Restricted Candidat List), el cual será utilizado para listar a los mejores pacientes y médicos candidatos. Por otro lado, tenemos el segundo grupo, el cual ayudará en operaciones auxiliares y de soporte necesarios para la resolución del problema.

Asimismo, se presentan las funciones objetivos utilizadas para evaluar la bondad de las soluciones generadas por los algoritmos utilizados.

1. Lista restringida de candidatos para pacientes

A continuación se explica la estructura de lista restringida para pacientes y la representación que tendrá a nivel de solución algorítmica.

1.1 Definición de la Estructura

Como se ha mencionado anteriormente, el RCL es una estructura fundamental utilizada por el algoritmo GRASP que permitirá agrupar un conjunto de posibles candidatos a ser óptimos. Para el problema planteado, se utilizará una lista restringida de pacientes candidatos que será utilizada en la fase de construcción del algoritmo.

Donde *pp*: Cantidad de pacientes; $\propto$: Constante de relajación de pacientes.

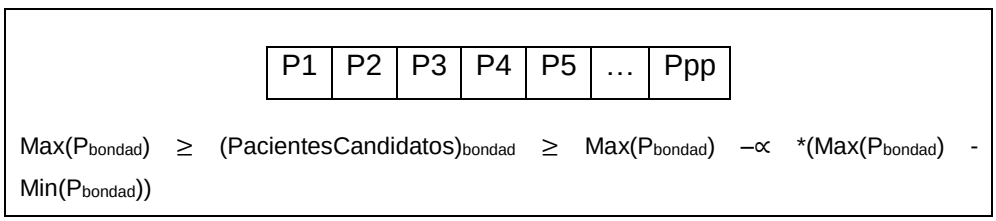


Figura 3.1 Lista restringida de candidatos – Pacientes
[Elaboración propia]

1.2 Representación

A continuación se muestra como se representará la lista restringida de candidatos para pacientes a nivel de código.

Lista<Paciente> pcl

2. Lista restringida de candidatos para médicos

A continuación se explica la estructura de lista restringida para médicos y la representación que tendrá a nivel de solución algorítmica.

2.1 Definición de la Estructura

Esta estructura permitirá almacenar a los médicos candidatos utilizados en la fase de construcción del algoritmo.

Donde *mm*: Cantidad de médicos; θ: Constante de relajación de médicos.

M1	M2	M3	M4	M5	…	Mmm

$Max(M_{bondad}) \geq (MédicosCandidatos)_{bondad} \geq Max(M_{bondad}) - \theta * (Max(M_{bondad}) Min(M_{bondad}))$

Figura 3.2. Lista restringida de candidatos - Médicos
[Elaboración propia]

2.2 Representación

A continuación se muestra como se representará la lista restringida de candidatos para pacientes a nivel de código.

Lista<Médico> mcl

3. Matriz de orden de atención al paciente

A continuación se explica la estructura de matriz de orden de atención para pacientes y la representación que tendrá a nivel de solución algorítmica.

3.1 Definición de la Estructura

Esta estructura consiste en una matriz de tamaño pp x mm, la cual será utilizada para almacenar el orden de atención de cada paciente asignado a un médico.

En la Figura 3.3, se puede observar un ejemplo del funcionamiento de esta matriz. En ella se puede observar que el médico M1 atenderá primero al paciente P1, seguido del paciente P3; para el caso del médico M2, éste atenderá primero al paciente P4, seguido del paciente P6, y así sucesivamente con los demás pacientes. De igual manera, sucederá lo mismo con cada médico.

Donde *pp*: Cantidad de pacientes; *mm*: Cantidad de médicos.

	M_1	**M_2**	**M_3**	**M_4**	**…**	**M_{mm}**
P_1	1	-	-	-	-	-
P_2	-	-	5	-	-	-
P_3	2	-	-	-	-	-
P_4	-	1	-	-	-	-
P_5	-	-	-	-	-	13
P_6	-	3	-	-	-	-
P_7	-	-	20	-	-	-
…	-	-	-	-	-	-
P_{pp}	-	-	-	30	-	-

Figura 3.3 Matriz de orden de atención al paciente
[Elaboración propia]

3.2 Representación

La matriz de orden de atención, al ser una estructura con datos enteros, será representada de la siguiente manera:

int[pp][mm] matrizOrdenAtencion

Esta matriz deberá inicializarse con valor 0.

4. Matriz de turnos disponibles de atención por médico

A continuación se explica la estructura de matriz de turnos disponibles de atención por cada médico y la representación que tendrá a nivel de solución algorítmica.

4.1 Definición de la Estructura

Esta estructura consiste en una matriz de tamaño 7 x mm, la cual será utilizada para almacenar la cantidad de turnos disponibles por día de cada médico. Dentro del algoritmo, será necesario crear una estructura copia en la cual se pueda ir reduciendo el número de turnos disponibles, y mantener los turnos originales para volver a establecer los turnos de atención cuando se cambia la semana de trabajo del médico. Asimismo, es necesario mencionar que los turnos de atención serán tomados de forma diaria, independientemente si son o no consecutivos, esto debido a que el tiempo de atención de cada paciente está basado en días.

En la Figura 3.4, se puede observar un ejemplo del funcionamiento de esta matriz. En ella se puede observar que la cantidad de turnos asignados al médico M1 es de 10 para los días Lunes, 6 para los días Martes, 8 para los días Miércoles, y así sucesivamente para los demás días. De igual manera, sucederá lo mismo con cada médico.

Donde *mm*: Cantidad de médicos.

	M 1	**M 2**	**M 3**	**…**	**Mm m**
L	10	6	6	18	0
M	6	4	14	16	18
M M	8	4	12	12	20
J	4	6	10	10	14
V	0	0	10	10	10
S	12	20	6	8	4
D	4	10	6	6	0

Figura 3.4. Matriz de turnos disponibles – Matriz original
[Elaboración propia]

Por ejemplo, para un ciclo Grasp de la fase de construcción, se tiene una copia de la matriz de turnos disponibles inicialmente que, conforme se van asignando pacientes y doctores, se va a modificando.

	M 1	M 2	M 3	…	Mm m
L	10	6	6	18	0
M	6	4	14	16	18
M M	8	4	12	12	20
J	4	6	10	10	14
V	0	0	10	10	10
S	12	20	6	8	4
D	4	10	6	6	0

Matriz copia (0 turnos asignados)

	M 1	M 2	M 3	…	Mm m
L	0	0	0	0	0
M	0	0	2	10	0
M M	0	0	12	12	0
J	0	4	10	10	0
V	0	0	10	10	0
S	10	20	6	8	4
D	4	10	6	6	0

Matriz copia (100 turnos asignados)

4.2 Representación

La matriz de turnos disponibles será representada de la siguiente manera:

int [7][mm] matrizTurnosDisponibles

5. Matriz de tiempo de atención del paciente

A continuación se explica la estructura de matriz de tiempo de atención que cada paciente deberá esperar para poder recibir atención médica, y la representación que tendrá a nivel de solución algorítmica.

5.1 Definición de la Estructura

Esta estructura consiste en una matriz de tamaño pp x mm, la cual será utilizada para almacenar los tiempos de atención (días) de cada paciente. Estos tiempos serán asignados en base al orden de atención y al turno disponible de cada médico.

En la Figura 3.5, se puede observar un ejemplo del funcionamiento de esta matriz. En ella se puede observar que el médico M1 atenderá en 9 días al paciente P1, en 6 días al paciente P2, y así sucesivamente con los demás pacientes. De igual manera, sucederá lo mismo con cada médico.

Donde *mm*: Cantidad de médicos; *pp:* Cantidad de pacientes

	M1	**M2**	**M3**	**M4**	**…**	**Mm m**
P1	1	-	-	-	-	-
P2	-	-	3	-	-	-
P3	3	-	-	-	-	-
P4	-	10	-	-	-	-
P5	-	-	-	-	-	8
P6	-	14	-	-	-	-
P7	-	-	10	-	-	-
…	-	-	-	-	-	-
Pp p	-	-	-	25	-	-

Figura 3.5 Matriz de tiempo de atención al paciente
[Elaboración propia]

5.2 Representación

A nivel de código la representación de la matriz de tiempos de atención es la siguiente.

int[pp][mm] matrizTiempoAtencion

Esta matriz deberá inicializarse con valor 0.

6. Lista de semana de trabajo por médico

A continuación se explica la estructura de lista de semana de trabajo para cada médico y la representación que tendrá a nivel de solución algorítmica.

6.1 Definición de la Estructura

Debido a que la programación de citas médicas es mensual, será necesario definir una estructura que permita conocer la semana de trabajo en la que se encuentra cada médico.

En la Figura 3.6, se puede observar un ejemplo del funcionamiento de esta lista. En ella se puede observar que el médico M1 se encuentra en la semana 1 de trabajo, es decir, que aún posee turnos de atención disponibles en la primera semana; el médico M2 se encuentra en la semana 2 de trabajo, lo que significa que los turnos disponibles para la primera semana ya han sido agotados. De igual manera, sucederá lo mismo con cada médico.

Donde *mm*: Cantidad de médicos.

	M1	M2	M3	M4	…	Mmm
Semana de trabajo	1	2	1	2		3

Figura 3.6 Lista de semana de trabajo
[Elaboración propia]

6.2 Representación

A nivel de código la representación de la lista de semana de trabajo es la siguiente:

int [mm] semanaTrabajo

Esta lista deberá inicializarse con el valor 1, debido a que se considera que todos los médicos inician con la primera semana de trabajo.

7. Resumen de Estructuras de Datos

A continuación se presentará un cuadro resumen conteniendo todas las estructuras presentadas anteriormente junto con la representación simplificada que será utilizada para el desarrollo de los pseudocódigos que serán presentados más adelante.

Nombre de la estructura	Descripción	Representación
Lista restringida de candidatos para pacientes	Permitirá almacenar a los pacientes candidatos.	PCL
Lista restringida de candidatos para médicos	Permitirá almacenar a los médicos candidatos.	MCL
Matriz de orden de atención al paciente	Matriz que almacenará el orden de atención de cada paciente, según cada médico asignado.	MO
Matriz de turnos disponibles de atención por médico	Matriz que almacenará los turnos disponibles de atención de cada médico.	MT
Matriz de tiempo de atención al paciente	Matriz que almacenará los tiempos de atención al cliente, en base al orden de atención asignado.	MA
Lista de semana de trabajo para médicos	Lista que permitirá conocer la semana de trabajo de cada médico.	S

Tabla 3.1 Resumen de estructura de datos
[Elaboración propia]

8. Función objetivo – Paciente

En este apartado se explicará la función objetivo definida para la selección de los mejores pacientes.

8.1 Definición

La siguiente función busca determinar el grado de bondad que posee un paciente.

Esta bondad será utilizada para establecer los niveles de prioridad de atención para cada uno de los pacientes a evaluar. A continuación se muestra dicha función:

$$FOP = \frac{\alpha f(E) + \beta CEDX + \gamma CF}{\omega IM}$$

Donde $f(E) = 0.015E^2 - E + 20$; $\alpha, \beta, \gamma, \omega$: Constantes configurables

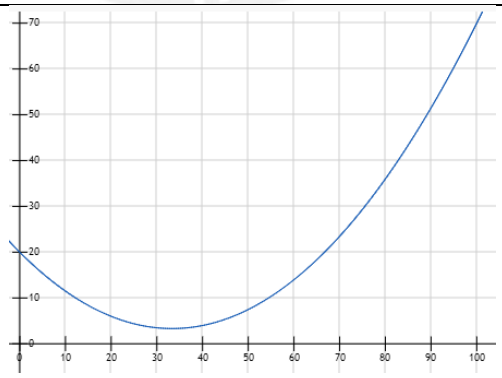
Figura 3.7 Función objetivo paciente
[Elaboración propia]

De la función presentada anteriormente, se hace referencia a diversas variables, tales como edad del paciente (E), costo de la enfermedad diagnosticada (CEDX), carga familiar (CF) e ingreso mensual del paciente (IM).

Para el presente caso, aquel paciente que posea un mayor valor de función objetivo, tendrá un mayor nivel de prioridad y por lo tanto más elegible a ingresar en la lista restringida de candidatos para pacientes.

8.2 Descripción de variables

E: Hace referencia a la edad del paciente. El sector salud posee políticas respecto a cierto rango de edades, como 0-3 años y 65 +años. Los pacientes cuya edad pertenezcan a esos rangos son considerados como población vulnerable. Con la finalidad de poder modelar este comportamiento, se hará uso de la siguiente función matemática:



$$f(E) = 0.015E^2 - E + 20$$

Figura 3.8. Función matemática para modelar el comportamiento de la edad del paciente.
[Elaboración propia]

Con esto, se desea que la prioridad de atención sea mayor en aquellos pacientes considerados como vulnerables.

CEDX: Hace referencia a la carga de enfermedad diagnosticada. Estos valores dependerán del tipo de enfermedad diagnosticada, las cuales, y como se explicó en capítulos anteriores, serán agrupados en 21 categorías de diagnóstico. En caso el paciente aún no posea un diagnóstico, es decir, sea su primera cita, se le asignará el valor de uno (1) a esta variable.

IM: Hace referencia al ingreso promedio mensual del paciente. Esto se realiza con la finalidad de priorizar a la población con menos recursos económicos. Los rangos de valores para esta variable dependerán de la edad que posee el paciente, se asume que el ingreso mensual para los pacientes menores de edad (<18 años) es igual a S/. 0, el de los adultos mayores (>65 años) variará entre S/.200 y S/. 700 (Un nuevo sol), mientras que para la población restante el ingreso mensual variará entre S/750.00 (Salario mínimo vital) y S/. 5000.00.

CF: Hace referencia a la carga familiar que posee el paciente. Lo que se desea, es que se priorice a aquellos pacientes que posean una mayor cantidad de hijos, debido a que, al ser el soporte principal familiar, un desgaste considerable de salud en el paciente puede conllevar a un mayor costo social hacia el Estado, como por ejemplo, posible desnutrición de los dependientes, acceso limitado a educación, entre otros. Los rangos de valores para esta variable son entre 0 – 5 hijos.

9. Función objetivo – Médico

En este apartado se explicará la función objetivo definida para la selección de los mejores médicos.

9.1 Definición

La siguiente función busca determinar el grado de bondad que posee un médico. Esta bondad será utilizada para establecer qué médicos están mejor calificados y/o tienen menor carga de trabajo para poder atender al paciente.

$$FOM = \frac{\propto EX}{\beta TC} * \theta FactorTurnos$$

Donde FactorTurnos=$(\frac{TD}{TT})$; $\propto, \beta, \theta$: Constantes configurables

Figura 3.9 Función objetivo médico
[Elaboración propia]

De la función presentada anteriormente, se hace referencia a diversas variables, tales como la experiencia del médico (EX), tiempo en días de la próxima cita disponible (TC), turnos disponibles de trabajo (TD) y turnos de trabajo de asignados (TT).

Para el presente caso, aquel médico que posea un mayor valor de función objetivo, será considerado como el más adecuado para una próxima asignación y por lo tanto más elegible a ingresar en la lista restringida de candidatos para médicos.

9.2 Descripción de variables

EX: Hace referencia a la experiencia, en años, del médico. Éste es mejor considerado si posee una mayor cantidad de años ejerciendo la carrera médica.

FactorTurnos: Hace referencia a la relación que existe entre los turnos disponibles (TD) y los turnos totales (TT) asignados a cada médico. Por lo tanto, a menor cantidad de turnos disponibles, el médico es menos considerado, debido a que se desea distribuir, uniformemente, la carga de trabajo para cada médico.

TC: Hace referencia al tiempo, en días, de la próxima cita disponible. Mientras más pronto sea ésta, mayor consideración se tendrá sobre el médico referido.

10. Función para la evaluación de soluciones

En este apartado se explicará la función objetivo definida para la selección de la mejor solución.

10.1 Definición

La siguiente función busca determinar el grado de bondad de las posibles soluciones de cada ciclo GRASP, de manera que se pueda discernir entre qué solución es mejor que otra. En el presente caso, aquella solución que tenga un mayor valor de función objetivo, será más óptima y por lo tanto más elegible a ser la solución final. Por ello se busca maximizar el valor de la función objetivo.

$$FS = Max\left(\sum_{i=0}^{NumPac} \frac{\propto BP_i * \beta BM_i}{\theta T_i}\right)$$

Donde FactorTurnos=$(\frac{TD}{TT})$; $\propto, \beta, \theta$: Constantes configurables

Figura 3.10 Función objetivo médico
Fuente: Elaboración propia

De la función presentada anteriormente, se hace referencia a diversas variables, tales como la bondad del paciente *"i"* (BP), la bondad del médico asignado al paciente *"i"* (BM) y el tiempo de espera necesario para la atención del paciente *"i"*(T)

10.2 Descripción de variables

BP_i: Hace referencia a la bondad del paciente *"i"*. Debido a que se desea dar atención a aquellos pacientes con mayor nivel de prioridad, según los factores considerados, será necesario maximizar el total de bondad de pacientes a los que se asignó una cita médica.

BM_i: Hace referencia a la bondad del médico asignado al paciente "*i".* Debido a que se desea asignar un médico con experiencia necesaria para poder atender a un paciente con condiciones médicas adversas, tales como edad y carga de enfermedad diagnosticada, será necesario maximizar la bondad del médico que se asigne a un paciente dado.

T_i: Hace referencia al tiempo que deberá esperar el paciente *"i"* para poder recibir atención médica. Se deberá considerar que el tiempo de atención para cada

paciente deberá ser menor en aquellos que poseen un mayor nivel de prioridad, con la finalidad de maximizar la función objetivo.

A continuación, se muestra un breve ejemplo sobre el funcionamiento de la función objetivo.

Pacientes

P1

Edad: 3
Carga Familiar: 0
Carga por enfermedad: 390
Ingreso mensual: 0
Bondad: 73 679

P2

Edad: 70
Carga Familiar: 0
Carga por enfermedad: 300
Ingreso mensual: 400
Bondad: 158

P3

Edad: 30
Carga Familiar: 2
Carga por enfermedad: 40
Ingreso mensual: 1500
Bondad: 1.43

Médicos

M1

Experiencia: 40
Tiempo cita: 1
Turnos trabajo: 1
Turnos disp. : 1
Bondad: 40

M2

Experiencia: 10
Tiempo cita: 1
Turnos trabajo: 1
Turnos disp. : 1
Bondad: 10

A continuación se presentan dos posibles soluciones:

Solución 1

 P3	**Bondad: 1.43** **T. atención: 1**	 M2	**Bondad: 10**	**Función Objetivo (FS1)** **438.71**
 P2	**Bondad: 27.07** **T. atención: 1**	 M1	**Bondad: 40**	

Tabla 3.2 Solución 1 – Ejemplo
[Elaboración propia]

Solución 2

 P1	**Bondad: 73 679**	 M1	**Bondad: 40**	**Función Objetivo (FS2)** **2 948 740**
 P2	**Bondad: 158** **T. atención: 1**	 M2	**Bondad: 10**	

Tabla 3.3 Solución 2 - Ejemplo
[Elaboración propia]

Como se puede apreciar en el ejemplo anterior, los valores obtenidos por las funciones objetivos de cada solución nos permiten determinar que la solución 2 (FS2) es mejor que la solución 1 (FS1).

CAPÍTULO 4: MODELO COMPUTACIONAL VORAZ

1. Algoritmo Voraz

En el presente apartado se presenta el algoritmo Voraz desarrollado para este proyecto. Si bien existen otros algoritmos utilizados en problemas de asignación, se optó por un algoritmo voraz debido a que éste refleja el comportamiento del modelo utilizado actualmente para la asignación de citas médicas, el cual consiste en atender a los pacientes en orden de llegada y asignarles un médico que posea el turno de atención más próximo. Además, la experimentación numérica entre el algoritmo Voraz y la solución Grasp propuesta, la cual será explicada y detallada en el siguiente capítulo, permitirá determinar si este último obtiene mejores resultados que el modelo utilizado actualmente, según los factores establecidos.

Asimismo, se realiza una breve descripción de las variables y estructuras utilizadas y una explicación detallada sobre el pseudocódigo propuesto para la resolución del problema.

1.1 Variables y Estructuras

Antes de poder visualizar el pseudocódigo del algoritmo Voraz, se presenta una breve descripción sobre las variables auxiliares utilizadas en la resolución del problema. Éstas serán usadas en conjunto con las estructuras de datos definidas en el capítulo anterior.

Representación	Descripción
numP	Cantidad de pacientes
numM	Cantidad de médicos
SolVoraz	Solución obtenida por la ejecución del algoritmo voraz
listaPendientesP	Lista utilizada para la selección de elementos
listaPendientesM	Lista utilizada para la selección de elementos

Representación	Descripción
listaMedicos	Lista original de todos los médicos involucrados en el problema. Esta lista será utilizada para poder obtener la posición relativa de cada médico en las matrices de orden de atención, tiempo de atención y turnos disponibles.
listaPacientes	Lista original de todos los pacientes involucrados en el problema. Esta lista será utilizada para poder obtener la posición relativa de cada paciente en las matrices de orden de atención y tiempo de atención.
PElegido	Paciente elegido
MElegido	Médico elegido

Tabla 4.1. Descripción de variables
[Elaboración propia]

1.2 Pseudocódigo

En esta sección se presenta el pseudocódigo de la solución voraz, la cual representa el modelo de asignación actual. En primer lugar, se muestra el diseño principal del procedimiento que hace el llamado al procedimiento voraz propiamente.

```
Procedimiento MainVoraz()
Inicio
        Leer(numIt, numC, numP, aplhaP, alphaM);
        Leer(listaPacientes,listaMedicos);
        InicializarMatrices(MA,MO,MT,S);

        SolVoraz<-Algoritmo_Voraz(numC, numP,MA, MTCopia,
        S,listaPacientes,listaMedicos)
Fin
```

Figura 4.1. Pseudocódigo MainVoraz
[Elaboración propia]

A continuación se muestra el pseudocódigo del algoritmo voraz desarrollado, así como una breve explicación del funcionamiento del algoritmo.

```
Procedimiento Algoritmo_Voraz(numC, numP, MA, MTCopia, S, listaPacientes, listaMedicos)
Inicio
1.      listaPendientesP= listaPacientes
2.      listaOrdenAt <- {0}
3.      Para cada especialidad hacer
        Inicio
3.1             Mientras !(esVacia(listaPendientesM)) y !((esVacia(listaPendientesP))

3.1.1               listaPendientesM=OrdenarTiempoAtencion(listaMédicos)

3.1.2               Pelegido<-listaPendientesP[0]

3.1.3               Melegido<-ObtenerMedicoMenorTiempo(Pelegido)

3.1.4               tAtencion<-7*S[obtenerIndice(listaMedicos,Melegido)] + obtenerDiaSemanaLibre(MTCopia,Melegido)

3.1.5               MT[obtenerIndice(listaPacientes,Pelegido)][obtenerIndice(listaMedicos,Melegido)]=tAtencion

3.1.6               listaOrdenAt[obtenerIndice(listaMedicos,Melegido)]++

3.1.7               MO[obtenerIndice(listaPacientes,Pelegido)][obtenerIndice(listaMedicos,Melegido)]<-
                    listaOrdenAt[obtenerIndice(listaMedicos,Melegido)]

3.1.8               Si (TurnosDisponibles(Melegido)==0) entonces

3.1.9               Remover(listaPendientesM,Melegido)
                Fin Mientras
        Fin Para
Fin
```

Figura 4.2. Pseudocódigo Voraz

[Elaboración propia]

- **Explicación Voraz**

Línea 1 – Se inicializa la variable listaPendientesP con los valores que posee la variable listaPacientes, la cual está ordenada en función al orden de llegada de cada paciente.

Línea 2 – Se inicializa la variable listaOrdenAt con valor 0.

Línea 3 – Se ejecuta la lógica para cada especialidad médica.

Línea 3.1 – Se inicia un bucle mientras la lista de médicos pendientes y la lista de pacientes pendientes no sean vacía.

Línea 3.1.1 – Se ordena la variable listaPendientesM en orden ascendente en relación a los tiempos de atención de cada médico.

Línea 3.1.2 – Se selecciona al primer paciente de la listaPendientesP.

Línea 3.1.3 – Se selecciona al primer médico cuyo tiempo de atención permita atender rápidamente al paciente seleccionado.

Línea 3.1.4 – Se obtiene el tiempo de la próxima cita disponible del elemento.

Línea 3.1.5 – Se asigna el tiempo de atención del paciente, según el médico asignado.

Línea 3.1.6 – Se incrementa el orden de atención para un médico dado.

Línea 3.1.7 - Se asigna el orden de atención del paciente, según el médico que lo atenderá.

Línea 3.1.8 – Si el médico elegido se queda sin turnos disponibles, se le remueve de listaPendientesM.

Línea 3.3.15 – Se remueve al médico elegido de la MCL

CAPÍTULO 5: MODELO COMPUTACIONAL GRASP

1 Algoritmo GRASP

En el presente apartado se presenta el algoritmo GRASP desarrollado, asimismo, se realiza una breve descripción de las variables y estructuras utilizadas, una explicación detallada sobre el pseudocódigo propuesto para la resolución del problema, y el análisis sobre el proceso de calibración de las constantes de relajación.

1.1 Variables y Estructuras

Antes de poder visualizar el pseudocódigo del algoritmo GRASP, se presenta una breve descripción sobre las variables auxiliares utilizadas en la resolución del problema. Éstas serán usadas en conjunto con las estructuras de datos definidas en el capítulo anterior.

Representación	Descripción
numIt	Esta variable hace referencia a la cantidad de iteraciones que se ejecutará el algoritmo GRASP.
numP	Cantidad de pacientes
numM	Cantidad de médicos
alphaP	Constante de relajación de pacientes
alphaM	Constante de relajación de médicos
SolConst	Solución de la fase de construcción
SolMejora	Solución de la fase de mejora
OptimaSol	Solución óptima del algoritmo GRASP
listaPendientesP	Lista utilizada para la selección de elementos del PCL
listaPendientesM	Lista utilizada para la selección de elementos del MCL
listaMedicos	Lista original de todos los médicos involucrados en el problema. Esta lista será utilizada para poder obtener la posición relativa de cada médico en las matrices de orden de atención, tiempo de atención y turnos disponibles.
listaPacientes	Lista original de todos los pacientes involucrados en el problema. Esta lista será utilizada para poder obtener la

Representación	Descripción
	posición relativa de cada paciente en las matrices de orden de atención y tiempo de atención.
listaOrdenAt	Lista auxiliar que permitirá saber el orden de atención de cada paciente, según el médico asignado.
PElegido	Candidato elegido de la lista PCL
MElegido	Candidato elegido de la lista MCL

Tabla 5.1. Descripción de variables

[Elaboración propia]

1.2 Pseudocódigo

En esta sección se presenta el pseudocódigo de la solución Grasp planteada. Como se mencionó en capítulos anteriores, se ha establecido que el algoritmo cuente con una fase de construcción y una de mejora.

```
Procedimiento MainGRASP()
Inicio
        Leer(numIt, numC, numP, aplhaP, alphaM);
        Leer(listaPacientes,listaMedicos);
        InicializarMatrices(MA,MO,MT,S);

        Para (contador:=0 hasta contador:=numIt)

                MTCopia<-MT
                SolConst<-Algoritmo_GRASP_Construccion(numC, numP, alphaP,
                alphaM, MA, MTCopia, S,listaPacientes,listaMedicos)

                SolMejora<-Algoritmo GRASP_Mejora()

                Si (contador=0) entonces
                        OptimaSol=SolMejora
                caso contrario
                        Si (FuncionCosto(OptimaSol) > FuncionCosto(SolMejora))
                        entonces
                                OptimaSol=SolMejora
        Fin Para
Fin
```

Figura 5.1. Pseudocódigo MainGrasp

[Elaboración propia]

A continuación se muestra el pseudocódigo de la fase de construcción del algoritmo, así como una breve explicación sobre su funcionamiento.

```
Procedimiento Algoritmo_GRASP_Construccion(numC, numP, alphaP, alphaM, MA,
MTCopia, S, listaPacientes, listaMedicos)
Inicio
 1.      listaPendientesP=OrdenarAscendenteBondad(listaPacientes)
 2.      listaOrdenAt <- {0}
 3.      Para cada especialidad hacer
         Inicio
3.1              PCL <- {}
3.2              MCL <- {}

3.3              Mientras !(esVacia(listaPendientesM)) y !((esVacia(listaPendientesP))

3.3.1               listaPendientesM=OrdenarAscendenteBondad(listaMédicos)
3.3.2               minP=obtenerBondad(listaPendientesP[0])
3.3.3               maxP=obtenerBondad(listaPendientesP[listaPendientesP.size-1])

3.3.4               Para cada (itemP<-listaPendientesP) hacer
                    Inicio
3.3.4.1                  bondadItem=obtenerBondad(itemP)
3.3.4.2                  Si (bondadItem<=maxP) y (bondadItem>=maxP-
                         alphaP*(maxP-minP)) entonces
                              AgregarRCL(PCL,itemP)
                    Fin Para

3.3.5               Pelegido<-Aleatorio(PCL)
3.3.6               minM=obtenerBondad(listaPendientesM[0])
3.3.7               maxM=obtenerBondad(listaPendientesM[listaPendientesM.size -
                    1])

3.3.8               Si ((NoTieneMedicoAsignado(Pelegido)) entonces
                    Inicio
 3.3.9                 Para cada (itemM <- listaPendientesM)
                       Inicio
3.3.9.1                   bondadItem=obtenerBondad(itemM)

3.3.9.2                   Si (bondadItem<=maxM) and (bondadItem>=maxM-
                          alphaP*(maxM-minM)) entonces
                               AgregarRCL (MCL,itemM)

 3.3.10                   Melegido<-Aleatorio(MCL)
                       Fin Para
                    Fin Si
                    Caso contrario
 3.3.11                Melegido <- SeleccionarCoincidencias(listaPendientesP)
```

```
3.3.12      tAtencion<-7*S[obtenerIndice(listaMedicos,Melegido)] +
            obtenerDiaSemanaLibre(MTCopia,Melegido)

3.3.13      MT[obtenerIndice(listaPacientes,Pelegido)][obtenerIndice(listaMedi
            cos,Melegido)]=tAtencion

3.3.14      listaOrdenAt[obtenerIndice(listaMedicos,Melegido)]++

3.3.15      MO[obtenerIndice(listaPacientes,Pelegido)][obtenerIndice(listaMed
            icos,Melegido)]<-
            listaOrdenAt[obtenerIndice(listaMedicos,Melegido)]

3.3.16      Si (TurnosDisponibles(Melegido)==0) entonces
                  Remover(listaPendientesM,Melegido)

3.3.17      Remover(MCL,Melegido)
3.3.18      Remover(listaPendientesP,Pelegido)
3.3.19      Remover(PCL,Pelegido)


          Fin Mientras
      Fin Para
Fin
```

Figura 5.2. Pseudocódigo GRASP – Construcción
[Elaboración propia]

- **Explicación GRASP – Construcción**

Línea 1 – Se inicializa la variable listaPendientes P con los valores que posee la variable listaPacientes, ordenados de forma ascendente en relación a su bondad.

Línea 2 – Se inicializa la variable listaOrdenAt con valor 0.

Línea 3 – Se ejecuta la lógica de la fase de construcción para cada especialidad médica.

Línea 3.1 – Se inicializa la variable PCL

Línea 3.2 – Se inicializa la variable MCL

Línea 3.3 – Se inicia un bucle mientras la lista de médicos pendientes y la lista de pacientes pendientes no sean vacías.

Línea 3.3.1 – Se inicializa la variable listaPendientesM con los valores que posee la variable listaMédicos, ordenados de forma ascendente en relación a su bondad.

Línea 3.3.2 – Se obtiene el valor mínimo de bondad en pacientes.

Línea 3.3.3 – Se obtiene el valor máximo de bondad en pacientes.

Línea 3.3.4 – Se realiza un análisis de todos los elementos de listaPendientesP con la finalidad de detectar a aquellos que deberán ser agregados al PCL.

Línea 3.3.4.1 – Se obtiene el valor de bondad del elemento.

Línea 3.3.4.2 – Si el elemento cumple con poseer una bondad dentro de los rangos establecidos entonces se agrega al PCL.

Línea 3.3.5 – Se selecciona un elemento de forma aleatoria del PCL.

Línea 3.3.6 – Se obtiene el valor mínimo de bondad en médicos.

Línea 3.3.7 – Se obtiene el valor máximo de bondad en médicos.

Línea 3.3.8 – Se verifica si el paciente elegido posee ya algún médico tratante.

Línea 3.3.9 - Se realiza un análisis de todos los elementos de listaPendientesM con la finalidad de detectar a aquellos que deberán ser agregados a la MCL.

Línea 3.3.9.1 – Se obtiene el valor de bondad del elemento.
Línea 3.3.9.2 – Si el elemento cumple con poseer una bondad dentro de los rangos establecidos entonces se agrega al MCL.

Línea 3.3.10 – Se selecciona un elemento de forma aleatoria del MCL.
Línea 3.3.11 – Si el paciente tiene ya un médico tratante, entonces se busca la mejor coincidencia dentro de listaPendientesM

Línea 3.3.12 - Una vez seleccionado el paciente y médico, y haciendo uso de la matriz de turnos disponibles y semana de trabajo de médico, se obtiene la cantidad de días necesarios para que el paciente pueda ser atendido.

Línea 3.3.13 – Se asigna el tiempo de atención del paciente, según el médico asignado.

Línea 3.3.14 – Se incrementa el orden de atención para un médico dado.

Línea 3.3.15 - Se asigna el orden de atención del paciente, según el médico que lo atenderá.

Línea 3.3.16 – Si el médico elegido de la MCL se queda sin turnos disponibles, se le remueve de listaPendientesM

Línea 3.3.17 – Se remueve al médico elegido de la MCL

Línea 3.3.18 – Se remueve al paciente de listaPendientesP

Línea 3.3.19 – Se remueve al paciente elegido del PCL.

Finalmente, se muestra el pseudocódigo de la fase de mejora del algoritmo, así como una breve explicación sobre su funcionamiento.

```
Procedimiento Algoritmo_GRASP_Mejora(numC, numP, alphaC, alphaP, MA, MTCopia, S, listaPacientes, listaMedicos)
Inicio

1.      Para cada columna (c) de MA
            Para f=0 hasta f=numFilas(MA)-1 hacer
1.1             Si (MA[f][c]!=0) entonces
```

```
                    Para cada t=f+1 hasta t=numFiles(MA) hacer
                        switch(MA[f][c],MA[t][c]
1.2                     Si (MejoraSolucion())
                            switch(MT[f][c],MT[t][c])
                            switch(MO[f][c],MO[t][c])
                        else
                            switch(MA[t][c],MA[f][c]
                    Fin Si
                Fin Si
            Fin Para
        Fin Para
Fin
```

Figura 5.3. Pseudocódigo GRASP – Mejora
[Fuente: Elaboración propia]

- **Explicación GRASP – Mejora**

  Línea 1 – Se realiza un recorrido de la estructura matriz de tiempo de atención (MT). Se procederá a evaluar cada fila (paciente) de cada columna (médico).

  Línea 1.1 – Si el tiempo de atención del paciente es distinto a cero (0), se procede a evaluar la mejora de la solución si es intercambiado con los demás pacientes del mismo doctor asignado.

  Línea 1.2 – Si la solución mejora, entonces se procede a realizar los cambios respectivos en la matriz de orden de atención.

1.3 Calibración de la constante de relajación para pacientes

A continuación se procederá a realizar la calibración del valor de alfa (Ver Tabla 5.2), el cual es la constante de relajación para los pacientes utilizada en el algoritmo Grasp implementado. Para la constante de calibración de médicos, la cual será calibrada en el apartado 1.4, se utilizó el valor de 0.5, y además, se utilizaron 1000 iteraciones por cada juego de datos. En este primer experimento se hizo variar el valor de alfa de 0.2 hasta 0.8, con un intervalo de 0.05. A continuación se muestra

la tabla resumen de estos primeros resultados. Para mayor detalle de este primer experimento, véase Anexo 1.

	0.2	0.25	0.30	0.35	0.4	0.45	0.5
Promedio	58,080.21	58,068.28	58,057.20	58,048.94	58,045.59	58,030.20	58,020.48
Valor Máximo	78,026.13	78,002.75	77,979.01	77,979.25	77,968.44	77,941.94	77,945.47
Valor Mínimo	50,220.86	50,206.94	50,202.90	50,198.44	50,186.46	50,176.93	50,165.11
Desviación	5,950.64	5,949.12	5,948.48	5,947.38	5,945.94	5,945.13	5,947.23

	0.55	0.60	0.65	0.7	0.75	0.8
Promedio	58,003.73	57,974.41	57,951.52	57,920.47	57,875.63	57,832.25
Valor Máximo	77,900.97	77,876.47	77,860.83	77,752.87	77,697.13	77,628.69
Valor Mínimo	50,155.16	50,120.56	50,121.62	50,092.75	50,059.43	50,042.97
Desviación	5,942.48	5,937.98	5,937.18	5,924.99	5,920.47	5,916.55

Tabla 5.2. Calibración de constante – Pacientes I

[Elaboración propia]

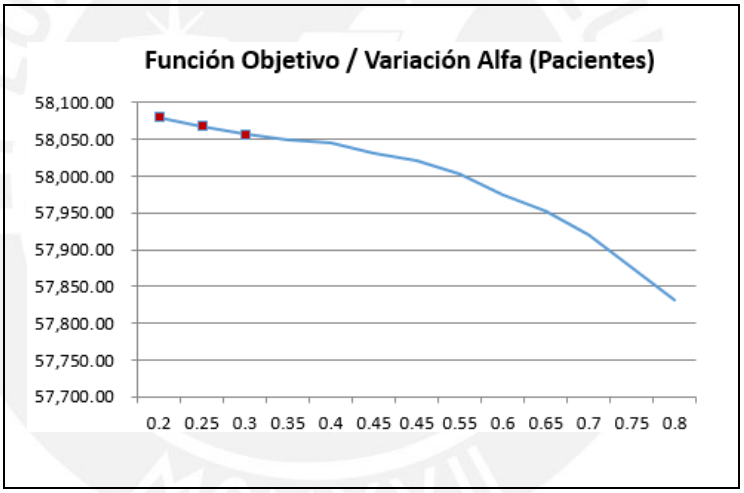


Figura 5.4. Gráfico resumen de calibración - Pacientes I

[Elaboración propia]

Debido a que se desea maximizar los valores de la función objetivo de la solución, se puede observar que entre los valores 0.2 y 0.3 se obtienen, en promedio, los mejores resultados, según la función objetivo. Por lo tanto, se decide realizar otra prueba, ahora haciendo variar los valores de alfa entre 0.2 y 0.3, en un intervalo de 0.01, con la finalidad de detectar algún valor dentro del rango mencionado que permita obtener mejores resultados. Para mayor detalle de este segundo experimento, véase Anexo 2.

	0.2	0.21	0.22	0.23	0.24	0.25
Promedio	58,081.12	58,080.58	58,077.76	58,077.61	58,073.64	58,066.89
Valor Máximo	78,017.69	78,022.67	78,021.58	78,020.01	78,004.75	78,004.98
Valor Mínimo	50,223.89	50,216.65	50,222.14	50,218.75	50,217.91	50,206.78
Desviación	5,949.92	5,950.82	5,950.74	5,950.18	5,948.16	5,950.01

	0.26	0.27	0.28	0.29	0.3
Promedio	58,066.85	58,064.75	58,062.98	58,059.89	58,058.46
Valor Máximo	78,004.09	78,005.85	77,990.29	77,990.62	77,989.93
Valor Mínimo	50,210.50	50,210.90	50,211.81	50,206.00	50,204.36
Desviación	5,949.91	5,949.12	5,948.53	5,949.60	5,949.70

Tabla 5.3. Calibración de constante – Pacientes II

[Elaboración propia]

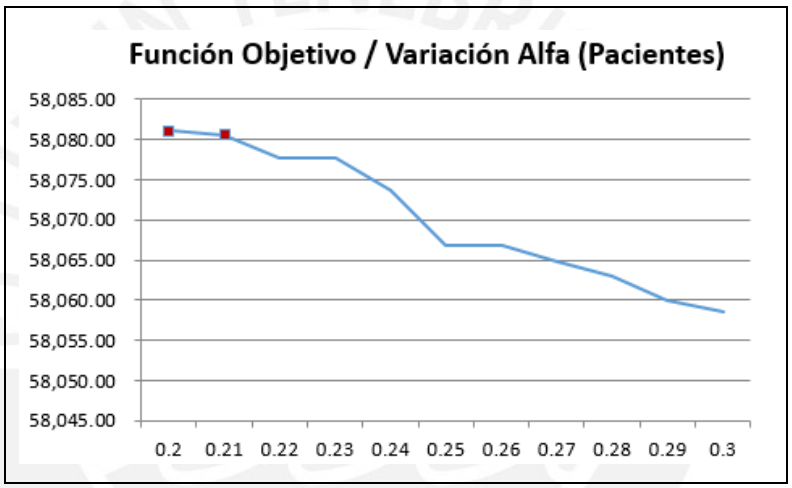


Figura 5.5. Gráfico resumen de calibración – Pacientes II

[Elaboración propia]

Como resultado de este según experimento, se puede observar que dentro del rango 0.20 y 0.21 para el valor de alfa se obtienen, en promedio, los mejores resultados. Por lo tanto, se procedió a hacer variar, una vez más, los valores para alfa entre el rango de 0.20 y 0.21, con un intervalo de 0.001. Para mayor detalle de este tercer experimento, véase Anexo 3.

	0.2	0.201	0.202	0.203	0.204	0.205
Promedio	58,080.62	58,079.64	58,079.56	58,081.23	58,080.81	58,080.47
Valor Máximo	78,015.94	78,020.73	78,019.30	78,019.60	78,024.05	78,018.63
Valor Mínimo	50,219.80	50,223.52	50,217.21	50,224.46	50,219.79	50,218.13
Desviación	5,951.11	5,950.34	5,950.73	5,950.77	5,950.73	5,950.09

	0.206	0.207	0.208	0.209	0.3
Promedio	58,080.37	58,080.78	58,080.39	58,080.68	58,080.94
Valor Máximo	78,016.99	78,026.11	78,017.10	78,014.43	78,018.19
Valor Mínimo	50,220.69	50,224.97	50,220.82	50,227.90	50,220.71
Desviación	5,950.38	5,949.65	5,949.38	5,950.07	5,950.29

Tabla 5.4. Calibración de constante – Pacientes II

[Elaboración propia]

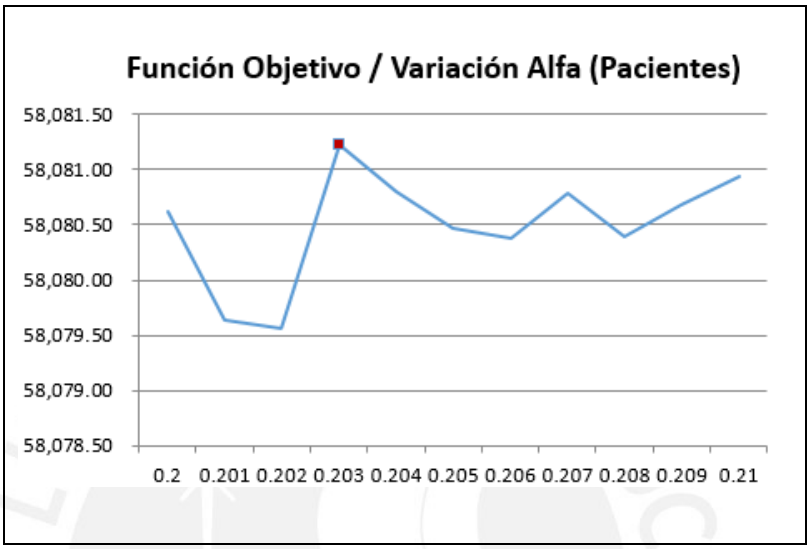


Figura 5.6.Gráfica resumen calibración– Pacientes III

[Elaboración propia]

Finalmente, se puede observar que para el valor de alfa igual a 0.203, se obtiene, en promedio, los mejores resultados para el problema planteado. Por lo tanto, se utilizará este valor para la constante de relajación para pacientes.

1.4 Calibración de la constantes de relajación para médicos

A continuación se procederá a realizar la calibración del valor de teta, el cual es la constante de relajación para los médicos utilizada en el algoritmo Grasp implementado. Para este experimento, se utilizó el valor obtenido de la calibración realizada en el apartado anterior para la constante de relajación para pacientes, y, de forma similar, la misma cantidad de iteraciones.

En este primer experimento se hizo variar el valor de teta de 0.2 hasta 0.8, con un intervalo de 0.05. A continuación se muestra la tabla resumen de estos primeros resultados. Para mayor detalle de este primer experimento, véase Anexo 4.

	0.2	0.25	0.3	0.35	0.4	0.45	0.5
Promedio	58,080.68	58,081.63	58,081.00	58,080.22	58,080.42	58,079.95	58,081.33
Valor Máximo	78,019.19	78,018.36	78,020.63	78,019.83	78,019.92	78,019.24	78,026.22
Valor Mínimo	50,220.16	50,225.49	50,225.15	50,221.15	50,221.22	50,221.07	50,218.67
Desviación	5,950.63	5,949.07	5,950.81	5,949.96	5,950.74	5,951.50	5,951.02

	0.55	0.6	0.65	0.7	0.75	0.8
Promedio	58,080.01	58,080.63	58,079.84	58,079.95	58,080.36	58,080.69
Valor Máximo	78,022.00	78,020.47	78,019.47	78,020.67	78,018.83	78,019.96
Valor Mínimo	50,223.57	50,223.24	50,220.30	50,220.00	50,220.60	50,219.64
Desviación	5,950.37	5,951.28	5,949.82	5,949.92	5,950.97	5,950.46

Tabla 5.5. Calibración de constante – Médicos I

[Elaboración propia]

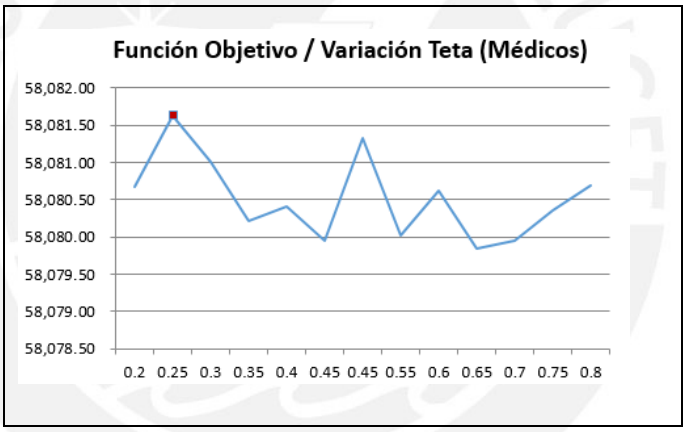


Figura 5.7. Gráfica resumen calibración– Médicos I

[Elaboración propia]

Debido a que se desea maximizar los valores de la función objetivo de la solución, se puede observar que entre los valores 0.2 y 0.3 se obtienen, en promedio, los mejores valores para teta. Por lo tanto, se decide realizar otra prueba, ahora haciendo variar los valores de teta entre 0.2 y 0.3, en un intervalo de 0.01, con la finalidad de detectar algún valor dentro del rango mencionado que permita obtener mejores resultados. Para mayor detalle de este segundo experimento, véase Anexo 5.

	0.2	0.21	0.22	0.23	0.24	0.25
Promedio	58,080.53	58,081.54	58,079.94	58,080.74	58,080.84	58,081.47
Valor Máximo	78,022.79	78,017.07	78,020.58	78,017.80	78,018.95	78,022.61
Valor Mínimo	50,221.36	50,220.54	50,225.14	50,222.54	50,223.68	50,222.59
Desviación	5,950.80	5,950.32	5,950.83	5,950.35	5,949.95	5,949.48

	0.26	0.27	0.28	0.29	0.3
Promedio	58,080.71	58,082.28	58,080.13	58,081.09	58,080.11
Valor Máximo	78,019.73	78,017.25	78,018.52	78,021.10	78,016.34
Valor Mínimo	50,223.19	50,229.47	50,221.25	50,224.19	50,223.98
Desviación	5,949.64	5,949.98	5,950.30	5,950.44	5,950.13

Tabla 5.6. Calibración de constante – Médicos II

[Elaboración propia]

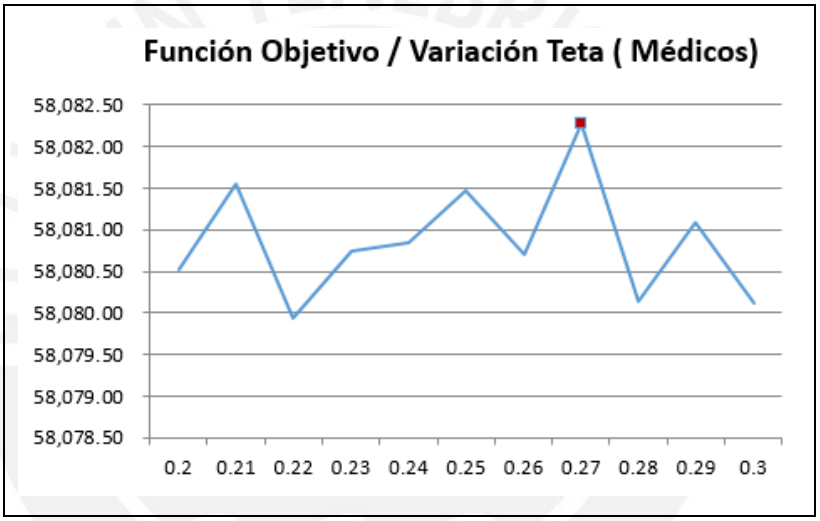


Figura 5.8. Gráfica resumen calibración– Médicos II

[Elaboración propia]

Debido a que se desea maximizar los valores de la función objetivo de la solución, se puede observar que entre los valores 0.25 y 0.27 se obtienen, en promedio, los mejores valores para teta. Por lo tanto, se decide realizar otra prueba, ahora haciendo variar los valores de teta entre 0.25 y 0.27, en un intervalo de 0.001, con la finalidad de detectar algún valor dentro del rango mencionado que permita obtener mejores resultados. Para mayor detalle de este tercer experimento, véase Anexo 6.

	0.25	0.251	0.252	0.253	0.254	0.255
Promedio	58,024.09	58,025.21	58,023.87	58,023.58	58,025.07	58,022.53
Valor Máximo	77,952.76	77,943.39	77,967.62	77,950.73	77,949.77	77,935.53
Valor Mínimo	50,172.22	50,179.50	50,177.37	50,186.95	50,167.38	50,173.26
Desviación	5,945.25	5,945.13	5,947.59	5,946.25	5,944.02	5,944.30

	0.256	0.257	0.258	0.259	0.26	0.261
Promedio	58,020.52	58,023.64	58,022.42	58,024.01	58,022.82	58,025.66
Valor Máximo	77,937.92	77,938.12	77,938.93	77,931.74	77,959.47	77,956.09
Valor Mínimo	50,164.10	50,159.38	50,180.66	50,169.29	50,180.29	50,176.81
Desviación	5,944.25	5,946.27	5,944.90	5,945.16	5,947.05	5,948.37

	0.262	0.263	0.264	0.265	0.266	0.267
Promedio	58,022.42	58,022.59	58,024.67	58,023.71	58,024.63	58,021.63
Valor Máximo	77,954.19	77,935.88	77,938.58	77,941.70	77,944.17	77,956.24
Valor Mínimo	50,176.63	50,188.52	50,177.86	50,164.09	50,186.31	50,181.75
Desviación	5,947.11	5,946.82	5,943.24	5,944.59	5,945.96	5,946.58

	0.268	0.269	0.27
Promedio	58,020.32	58,022.59	58,022.74
Valor Máximo	77,945.88	77,948.27	77,949.55
Valor Mínimo	50,171.07	50,170.03	50,179.20
Desviación	5,947.24	5,947.06	5,945.49

Tabla 5.7. Calibración de constante – Médicos III

[Elaboración propia]

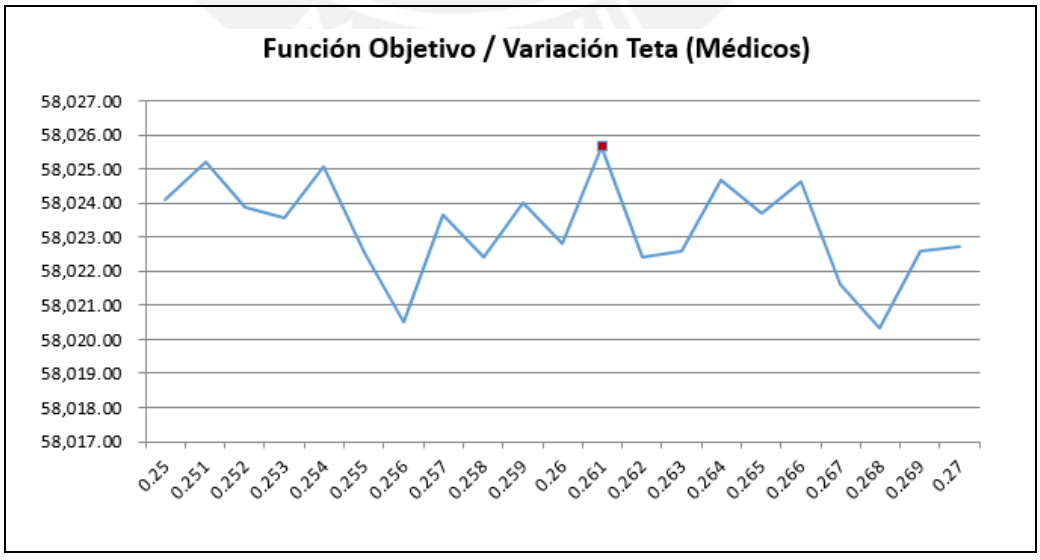


Figura 5.9. Gráfica resumen calibración– Médicos III

[Elaboración propia]

Finalmente, se puede observar que para el valor de teta igual a 0.261, se obtiene, en promedio, los mejores resultados para el problema planteado. Por lo tanto, se utilizará este valor para la constante de relajación para médicos.

1.5 Calibración del número de iteraciones

Una vez obtenidos los valores de las constantes de relajación, tanto para paciente como para médicos, se procedió a realizar la calibración del número de iteraciones del algoritmo GRASP implementado. Para esto se ejecutó una variedad de pruebas haciendo variar la cantidad de iteraciones desde 1000 hasta 10500 con un intervalo de 500. Finalmente, se obtiene aquella cantidad de iteraciones que generen un mayor valor de función objetivo en un tiempo aceptable. Para mayor detalle de este primer experimento, véase Anexo 7.

N.Iteraciones	Función Objetivo	TiempoRespuesta
1000	58,086.39	5,531.88
1500	58,086.80	8,259.95
2000	58,086.66	10,984.10
2500	58,087.46	13,737.75
3000	58,087.79	16,444.38
3500	58,087.76	19,199.18
4000	58,087.59	21,957.73
4500	58,088.63	24,795.93
5000	58,088.57	27,403.03
5500	58,089.60	30,134.58
6000	58,089.17	32,942.98
6500	58,090.04	35,679.05
7000	58,088.99	38,327.98
7500	58,088.96	41,166.45
8000	58,088.72	43,778.78
8500	58,089.05	46,549.60
9000	58,089.16	49,268.20
9500	58,090.03	51,986.20
10000	58,089.89	54,760.40
10500	58,090.17	57,424.20

Tabla 5.8. Calibración de constante – Médicos III

[Elaboración propia]

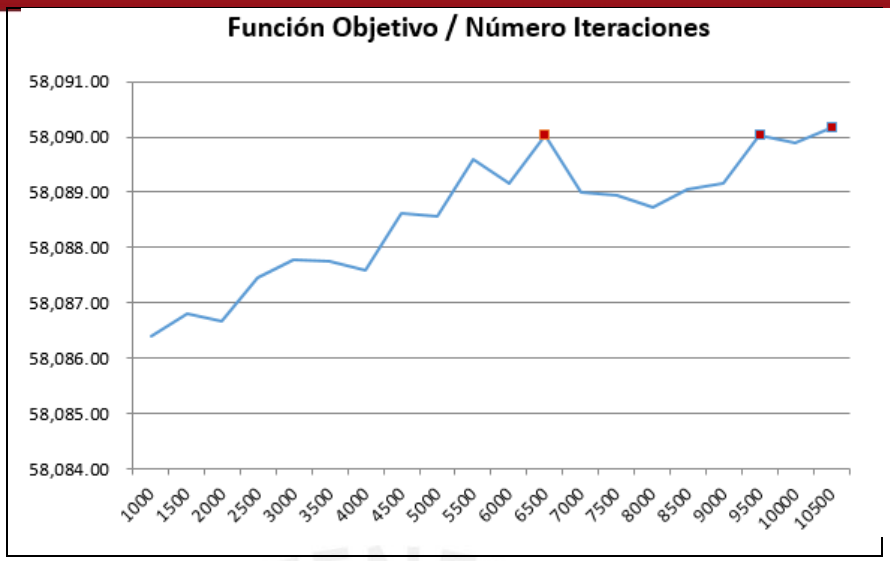


Figura 5.10. Gráfica resumen iteraciones
[Elaboración propia]

De los resultados obtenidos, se puede observar que para las 6500, 9500 y 10500 iteraciones el algoritmo GRASP proporciona los mejores resultados en relación a la función objetivo, los cuales son muy cercanos. De estos tres valores, se optó por el valor de 6500 debido a que el tiempo de ejecución es mucho menor en relación a los otros dos valores.

CAPÍTULO 6: MÉTODOS ESTADÍSTICOS Y RESULTADOS

1. Recolección de los Datos

Para realizar la experimentación numérica entre el algoritmo Voraz y el Grasp, se realiza una serie de ejecuciones de ambos algoritmos utilizando un número definido de juegos de datos, los cuales son la data de entrada generada para ser utilizada por los algoritmos. Al tratarse de una serie de juegos de datos y estos ser una muestra representativa del comportamiento general de los algoritmos mencionados, al finalizar la experimentación numérica, se pueden generalizar los resultados obtenidos más allá de las pruebas realizadas, pudiendo determinar cuál de los dos algoritmos permite obtener mejores resultados.

Para este caso, se realizarán 40 pruebas, cada una haciendo uso de un juego de datos diferentes, pero que, por más distintos que sean, cumplen con cierta semejanza, de manera que se tenga una muestra representativa con una varianza aceptable. Configurando la herramienta generadora de datos, se generaron los 40 juegos de datos con los siguientes parámetros: número de pacientes igual a 800, número de médicos igual a 15, número de especialidades igual a 5, turnos mínimos diarios de atención igual a 5 y turnos máximos diarios de atención igual a 15.

Adicionalmente a ello, al tratarse de algoritmos que tienen un componente aleatorio, se decidió tratar de quitar este posible sesgo en la experimentación numérica, realizando una serie de ejecuciones por cada uno de los juegos de datos obtenidos, utilizando como dato para la experimentación el promedio de los resultados obtenidos. De tal manera se tiene 10 ejecuciones por cada uno de los juegos de datos obtenidos, teniendo así en total 400 ejecuciones para ambos algoritmos.

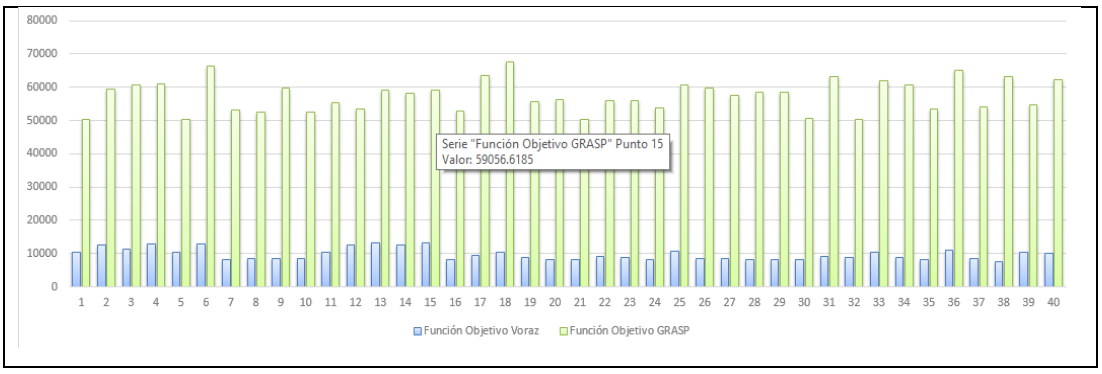


Figura 6.1. Gráfico de función objetivo de la muestra obtenida
[Elaboración propia]

2. Herramienta generadora de datos

Debido a que es necesario contar con datos de entrada que puedan ser utilizados por los algoritmos desarrollados, se implementó una herramienta que ayude a generar esta información en base a ciertos requerimiento iniciales establecidos, tales como el número de pacientes, número de médicos, número de especialidades, turnos mínimos y máximos de atención diarios, y los valores de carga por cada categoría de enfermedad.

2.1 Restricciones consideradas

Con la finalidad de que la herramienta a utilizar brinde información coherente y semejante a una posible data real, se han considerado ciertas restricciones respecto a los datos resultantes.

Factor considerado	Restricción
Pacientes	
Edad	Según la esperanza de vida de una persona en el Perú, se ha considerado que la edad de una persona puede variar de 0 años a 82 años.
Ingreso mensual	Según la edad del paciente: Si el paciente es aún menor de edad, el ingreso mensual deberá ser equivalente a S/. 0.00 (Cero soles). Si el paciente es mayor de 65 años (adulto mayor), el ingreso mensual considerado varía entre los S/. 200.00 y S/. 600.00. Para los demás pacientes, el ingreso mensual considerado varía entre los S/.

	750.00 y los S/. 5000.00
Carga familiar	Según la edad del paciente: Si el paciente es aún menor de edad o mayor a 65 años, la carga familiar asignada es igual a 0. Para los demás pacientes, la carga familiar varía entre los 0 y 5 hijos.
Número de citas	Según el MINSA, el número promedio de visitas que realiza un paciente en un determinado año equivale a 15 visitas/año. Por lo tanto, este valor ha sido considerado como tope máximo de número de citas de un paciente.
Médicos	
Experiencia	Según el MINSA, la cantidad de años de experiencia de sus médicos varía entre los 0 y 40 años. Por lo tanto, se han considerado esos valores para el rango de años de experiencia por cada médico.

Tabla 6.1 Restricciones consideradas
[Elaboración propia]

A continuación se muestra la pantalla de la herramienta generadora:

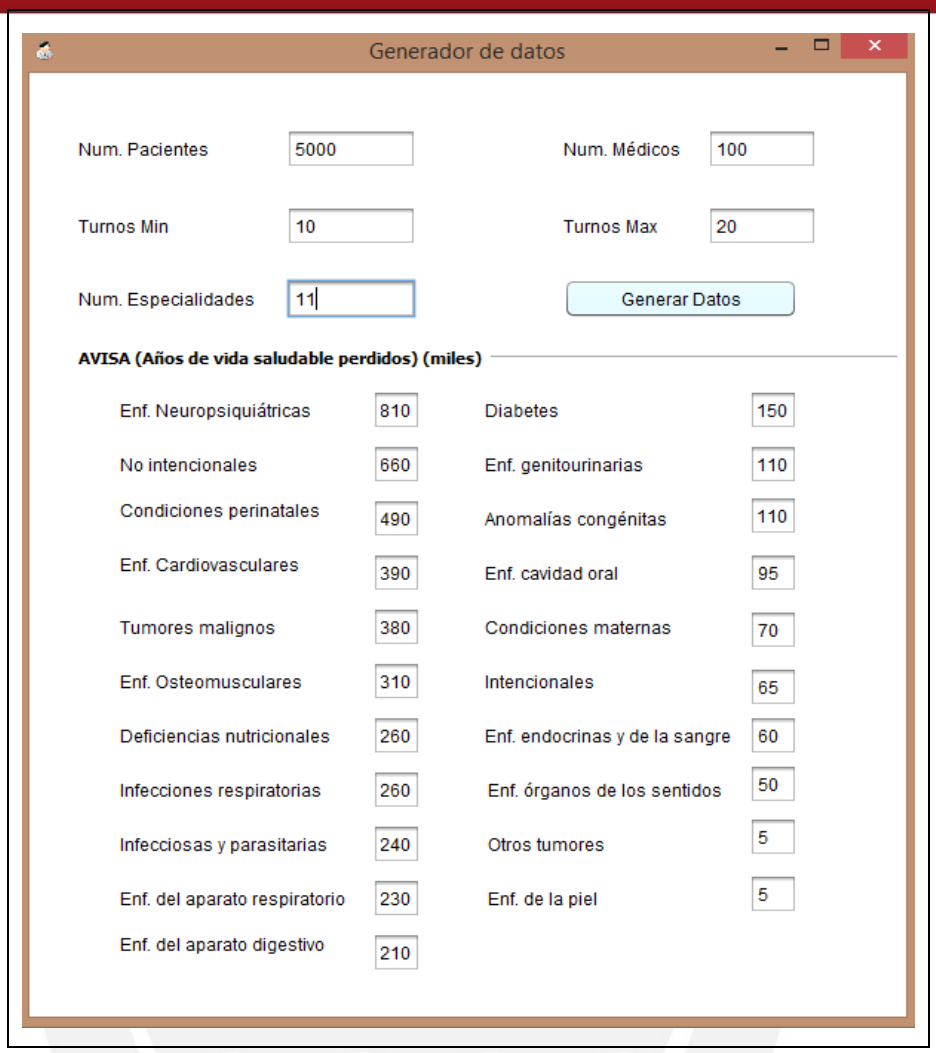


Figura 6.2. Herramienta generador a de datos
[Elaboración propia]

2.2 Estructura de archivo generado

Una vez que la herramienta haya generado la data necesaria, ésta será almacenada en un archivo de texto para, posteriormente, ser utilizada por la aplicación que permita la asignación de citas médicas. Con la finalidad de facilitar la carga de la información en la aplicación desarrollada, se ha definido la siguiente estructura de archivo:

1er Bloque: Información general

Cantidad de pacientes	Cantidad de médicos	Cantidad de especialidades

Figura 6.3. Bloque 1 – Formato de datos generados
[Elaboración propia]

2do Bloque: Información de médicos

Nombre médico	CMP	Experiencia	Especialidad	#Turnos (Lunes)	…	#Turnos (Domingo)

Figura 6.4. Bloque 2 – Formato de datos generados
[Elaboración propia]

3er Bloque: Información de pacientes

Nombre paciente	Edad	Carga familiar	Ingreso del paciente	Esp. requerida	Carga de enfermedad diagnosticada	CMP

Figura 6.5. Bloque 3– Formato de datos generados
[Elaboración propia]

Finalmente, y siguiendo la estructura definida anteriormente, el archivo de texto almacenaría la información de la siguiente forma:

```
5000 50 11

Medico0 M-0 35 2 19 15 14 11 10 16 15
Medico1 M-1 3 2 14 14 18 11 18 10 15
Medico2 M-2 4 2 11 18 16 10 18 15 17
Medico3 M-3 33 2 17 14 18 16 15 17 12
Medico4 M-4  10 2 16 17 14 16 19 18
Medico5 M-5 19 2 1 19 19 10 19 19 17

Paciente0 6 0 1 2 310 M-2 5
Paciente1 27 4 1200 1 110 M-4 17
Paciente2 54 4 1900 2 1 -1 14
Paciente3 3 0 1 1 1 -1 9
Paciente4 67 0 300 1 1 -1 14
```

Figura 6.6 Estructura de datos de archivo de texto generado
[Elaboración propia]

3 Pruebas estadísticas empleadas

La experimentación numérica tiene por objetivo determinar qué algoritmo, de los definidos para realizar la experimentación, posee un mejor desempeño. Para poder realizar esto, se hará uso de la prueba de comparación de medias de los resultados obtenidos por cada algoritmo implementado. Debido a que el número de elementos a evaluar es igual a 40, tal y como se indicó al inicio de este capítulo, el uso de la prueba de t-Student está descartada ya que, según la teoría, esta prueba solo puede ser aplicada cuando el tamaño de muestreo es menor o igual a 30, por lo que se ha optado por la prueba estadística Z, la cual es una generalización de la prueba t-Student.

Sin embargo, para poder emplear la prueba Z, es necesario verificar que las varianzas que poseen las muestras a analizar deben ser diferentes y que, además, deben poseer un comportamiento normal.

Por lo tanto, será necesario aplicar la prueba F de Fisher. Esta prueba verifica la homogeneidad de la varianza de los elementos a evaluar. Asimismo, con la finalidad de determinar que las muestras siguen un comportamiento normal, se aplica la prueba de Kolmogorov-Smirnov.

3.1 Prueba de Kolmogorov-Smirnov

Como ya se mencionó anteriormente, lo que se busca en esta prueba es determinar si cada una de las muestras obtenidas de forma independiente, tanto del algoritmo Grasp como del Voraz, cumplen con seguir una distribución normal. Para ello se presenta las siguientes hipótesis:

$$H_0: La\ muestra\ sigue\ una\ distribucion\ normal$$
$$H_1: La\ muestra\ no\ sigue\ una\ distribucion\ normal$$

Figura 6.7 Hipótesis Kolmogorov-Smirnov
[Elaboración propia]

Dado que la hipótesis nula H_0 presentada es el caso de igualdad, se puede determinar que ambas hipótesis mostradas en la Figura 6.7 siguen un esquema que

planteará una prueba de dos colas. Por otro lado, para la hipótesis alternativa H_1 se presenta una desigualdad.

Una vez aplicada esta prueba a la muestra correspondiente a los resultados del algoritmo Voraz, los resultados obtenidos son presentados en la Tabla 6.2. Para mayor detalle, véase el Anexo 8.

Resultados Kolmogorov - Voraz	
Valor del estimador	0.205
Número de muestras	40
Nivel de significancia	0.05
Valor crítico	0.215
Conclusiones del experimento: Estimador fuera de la zona crítica, por lo tanto, se acepta la hipótesis nula $\boldsymbol{H_0}$	

Tabla 6.2 Prueba Kolmogorov - Voraz
[Elaboración propia]

Como se puede apreciar en la Tabla 6.2, el valor del estimador obtenido es menor en relación al valor crítico, por lo que se determina que el valor estimado de la muestra analizada se encuentra fuera del rango de la zona crítica y, por ende, la aceptación de la hipótesis nula $\boldsymbol{H_0}$. Dados estos resultados, se puede concluir que este modelo sigue una distribución normal.

Asimismo, esta misma prueba también fue aplicada a la muestra correspondiente a los resultados del algoritmo Grasp, cuyos resultados se pueden observar en la Tabla 6.3. Para mayor detalle, véase el Anexo 9.

Resultados Kolmogorov - Grasp	
Valor del estimador	0.086
Número de muestras	40
Nivel de significancia	0.05
Valor crítico	0.215
Conclusiones del experimento: Estimador fuera de la zona crítica, por lo tanto, se acepta la hipótesis nula $\boldsymbol{H_0}$	

Tabla 6.3 Prueba Kolmogorov - Grasp
[Elaboración propia]

Para este caso, tal como se puede apreciar en la Tabla 6.3, se obtuvo que el valor del estimador es menor al valor crítico obtenido en esta prueba, por lo que se determina que la muestra analizada se encuentra fuera del rango de la zona crítica, y por lo tanto, se acepta la hipótesis nula $\boldsymbol{H_0}$**.** Dados estos resultados, se puede que este modelo sigue, también, una distribución normal.

3.2 Prueba F de Fischer

El objetivo de la presenta prueba es el de determinar si las varianzas de ambas muestras son significativamente homogéneas o significativamente diferentes. Como requisito para esta prueba, es necesario que ambas muestras posean una distribución normal, lo cual pudo ser verificado haciendo uso de la prueba anteriormente mencionada. Para cada una de las muestras dadas, se presentan las siguientes hipótesis.

$$H_0\text{: } Las\ varianzas\ son\ significativamente\ homogéneas$$
$$H_1\text{: } Las\ varianzas\ son\ significativamente\ diferentes$$

Figura 6.8 Hipótesis Fisher
[Elaboración propia]

Las hipótesis planteadas permitirán determinar el comportamiento de las varianzas de ambas muestras. La hipótesis nula hace referencia a que las varianzas de ambas muestras son significativamente homogéneas, mientras que la hipótesis alternativa indica que las varianzas de las muestras analizadas son significativamente distintas.

Una vez aplicada esta prueba a las muestras correspondientes a los resultados de los algoritmos Voraz y Grasp, los resultados obtenidos son presentados en la Tabla 6.4.

Resultados Fisher		
	Voraz	**Grasp**
Media	9774.1993	57463.4654
Varianza	2967959.6	22248633.02

Número de muestras	40		40
Grados de libertad	39		39
Valor crítico para F		2.14	
Valor F		0.1334	

Conclusiones del experimento:

Valor F para ambas muestras es menor al valor crítico de F, por lo tanto, se acepta la hipótesis nula $\boldsymbol{H_0}$.

Tabla 6.4 Prueba Fisher

[Elaboración propia]

Como se puede apreciar en la Tabla 6.4, el valor F obtenido es menor en relación al valor crítico obtenido para ambas muestras, por lo que se determina que el valor F de la muestra analizada se encuentra fuera del rango de la zona crítica y, por ende, la aceptación de la hipótesis nula $\boldsymbol{H_0}$. Dados estos resultados, se puede concluir que las varianzas de ambos algoritmos son significativamente homogéneas.

3.3 Prueba Z

Finalmente, esta última prueba, aplicada a los modelos presentados, busca determinar cuál de los dos modelos tiene la media más alta. Esta prueba es sumamente importante ya que se busca demostrar que los resultados obtenidos por el algoritmo Grasp son mejores que los obtenidos por el algoritmo Voraz. Para ello, se utilizarán los valores de las funciones fitness obtenidos de ambas muestras, por lo que, el modelo que contenga el valor de media más alto, será aquel con los resultados más óptimos según lo planteado.

Para poder determinar lo ya mencionado se realizaran dos pruebas de hipótesis. La primera consiste en determinar si ambos algoritmos poseen resultados similares, y la segunda, determinar si el algoritmo Grasp obtiene mejores resultados a los obtenidos por el algoritmo Voraz. Por lo tanto, se plantean las siguientes hipótesis.

$$H_0\colon \textit{La media del algoritmo Voraz es igual a la media del algoritmo Grasp}$$
$$H_1\colon \textit{La media del algoritmo Voraz es diferente a la media del algoritmo Grasp}$$

Figura 6.9 Hipótesis Z

[Elaboración propia]

La hipótesis nula intenta determinar que los resultados obtenidos por el algoritmo Grasp son equivalentes a los resultados obtenidos por el algoritmo Voraz, mientras que la hipótesis alterna intenta determinar lo contrario.

Una vez aplicada esta prueba a las muestras correspondientes a los resultados de ambos algoritmos, los resultados obtenidos son presentados en la Tabla 6.5.

Resultados Z		
	Voraz	**Grasp**
Media	9774.1993	57463.4654
Varianza	2967959.6	22248633.02
Número de muestras	40	40
Nivel de significancia	0.05	
Z	60.0630	
Valor crítico de Z (dos colas)	1.96	
Conclusiones del experimento: Valor Z dentro de la zona crítica, por lo tanto, se rechaza la hipótesis nula $\boldsymbol{H_0}$		

Tabla 6.5 Hipótesis Z

[Elaboración propia]

Como se puede apreciar en la Tabla 6.5, el valor de Z es mayor al valor crítico de Z, por lo que se determina que el valor Z está dentro de la región crítica, y por ende, el rechazo de la hipótesis nula $\boldsymbol{H_0}$. Dados estos resultados, se puede concluir que las medias de los resultados de ambos algoritmos son diferentes.

Debido a que ya se sabe que las medias de las muestras dadas son distintas, es necesario determinar, qué algoritmo obtiene mejores resultados. Para ello, es necesario plantear la segunda prueba de hipótesis, por lo que se presentan las siguientes hipótesis para esta segunda prueba.

H_0: *La media del algoritmo Voraz es mayor a la media del algoritmo Grasp*

H_1: *La media del algoritmo Voraz es menor a la media del algoritmo Grasp*

Figura 6.10 Hipótesis Z
[Elaboración propia]

La hipótesis nula intenta determinar que los resultados obtenidos por el algoritmo Voraz son mejores que los resultados obtenidos por el algoritmo Grasp, mientras que la hipótesis alterna intenta determinar lo contrario.

Una vez aplicada esta prueba a las muestras correspondientes a los resultados de ambos algoritmos, los resultados obtenidos son presentados en la Tabla 6.6.

Resultados Z		
	Voraz	**Grasp**
Media	9774.1993	57463.4654
Varianza	2967959.6	22248633.02
Número de muestras	40	40
Nivel de significancia	0.05	
Z	-60.0630	
Valor crítico de Z (una cola)	-1.6448	
Conclusiones del experimento:		
Valor Z dentro de la zona crítica, por lo tanto, se rechaza la hipótesis nula $\boldsymbol{H_0}$		

Tabla 6.6 Hipótesis Z
[Elaboración propia]

Como se puede apreciar en la Tabla 6.6, el valor de Z es menor al valor crítico de Z, por lo que se determina que el valor Z está dentro de la región crítica, y por ende, el rechazo de la hipótesis nula $\boldsymbol{H_0}$. Dados estos resultados, se puede concluir que los resultados obtenidos por el algoritmo Grasp son mejores a los obtenidos por el algoritmo Voraz.

CAPÍTULO 7: INTERFAZ GRÁFICA

En este capítulo se mostrará el diseño de las ventanas gráficas necesarias para la elaboración de la herramienta informática que permita la asignación de citas médicas.

1. Panel de carga de datos generados

En la Figura 7.1, se puede apreciar la interfaz gráfica definida para la carga de datos, los cuales serán utilizados por el algoritmo. Los datos que serán utilizado por el algoritmo, serán obtenidos a través de un archivo de texto, el cual poseerá una estructura ya definida, tal y como se muestra en el capítulo anterior (Figura 5.1, 5.2 y 5.3). Una vez realizada la carga, los datos obtenidos del archivo podrán ser visualizados, lo que permitirá verificar que son los datos que se quieren.

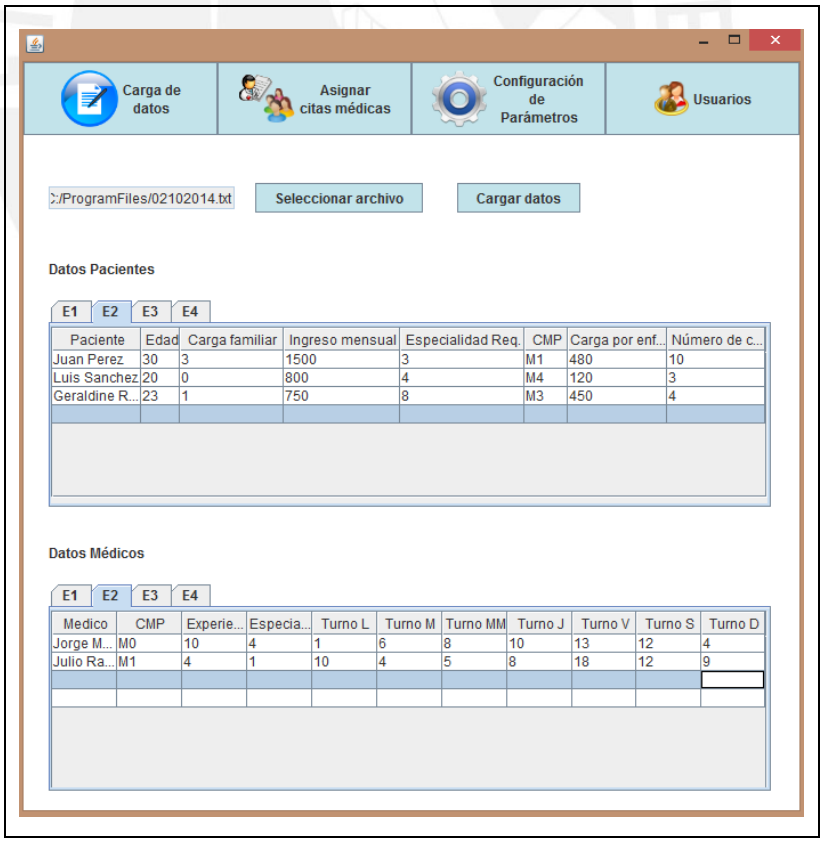


Figura 7.1 Interfaz herramienta - Carga de datos

[Elaboración propia]

2. Panel de configuración de parámetros del algoritmo

En la Figura 7.2, se puede apreciar la interfaz gráfica definida para la configuración de los parámetros del algoritmo.
En esta ventana, se podrá asignar valores a las constantes de relajación, establecer el número de iteraciones que usará el algoritmo, y los pesos correspondientes a cada uno de los factores definidos para cada función objetivo.

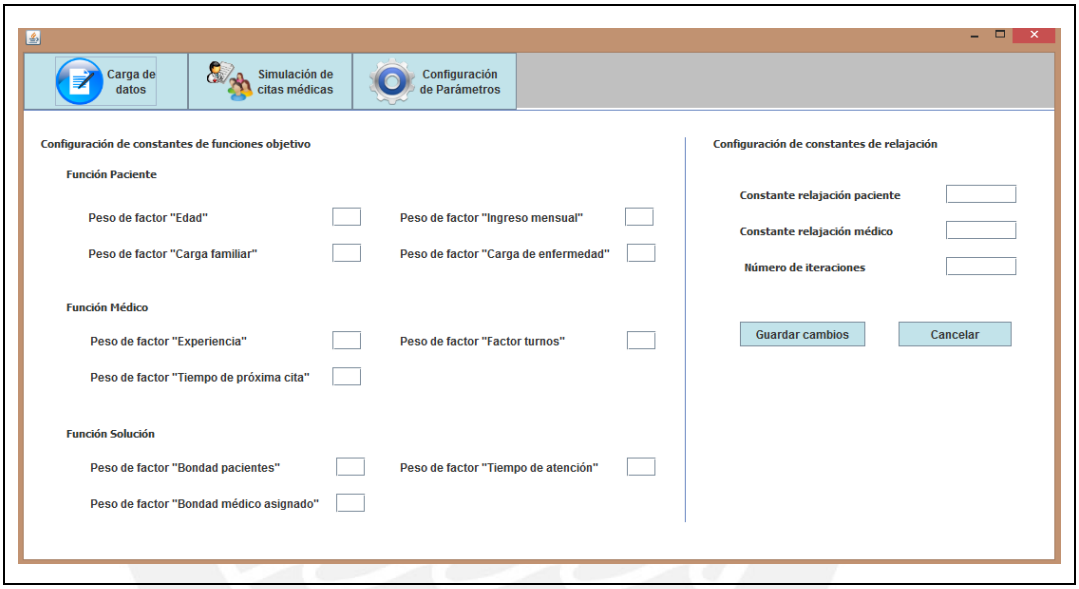


Figura 7.2 Interfaz herramienta - Configuración de parámetros
[Elaboración propia]

3. Panel para la asignación de citas médicas

En la Figura 7.3, se puede apreciar la interfaz gráfica definida para la asignación de citas médicas. Dentro de esta ventana, se podrán ingresar los parámetros iniciales utilizados por el algoritmo y, una vez realizada la asignación, se podrá visualizar los resultados (Ver Figura 7.4).

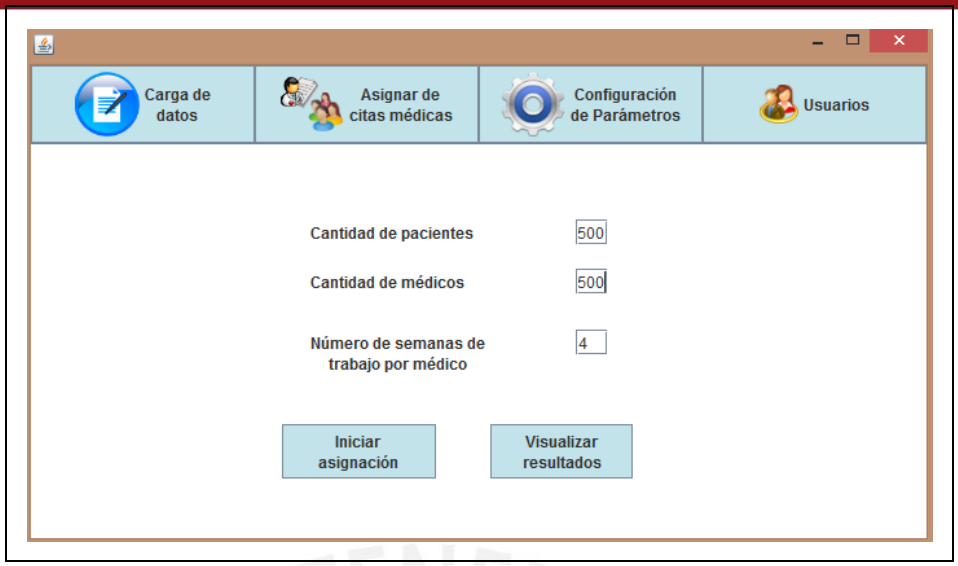


Figura 7.3 Interfaz herramienta - Asignación de citas

[Elaboración propia]

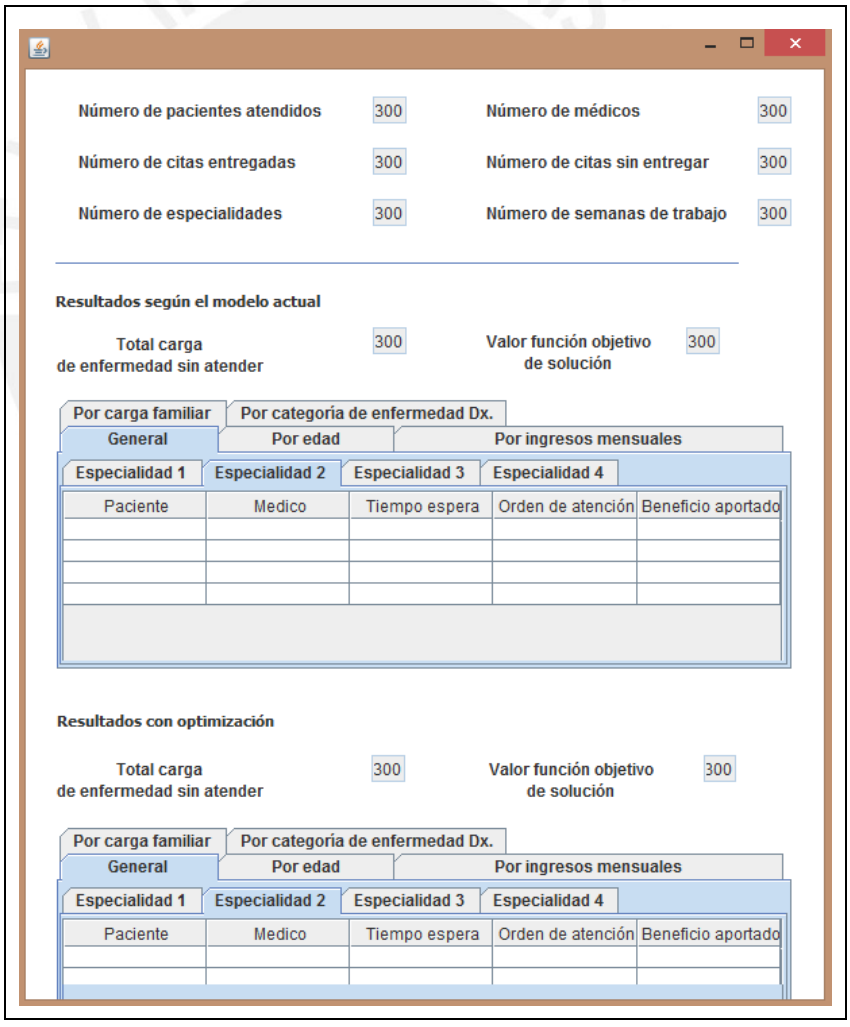


Figura 7.4. Interfaz herramienta - Visualización de resultados

[Elaboración propia]

CAPÍTULO 8: CONCLUSIONES

1 Conclusiones

Luego de haber desarrollado los algoritmos Voraz y Grasp, para resolver el problema de la asignación de citas médicas en hospitales, así como la comparación de los resultados obtenidos a través de la experimentación numérica, se puede concluir el presente trabajo con las siguientes afirmaciones:

Para el algoritmo Voraz, se puede determinar que, según los factores definidos para este proyecto, genera resultados no tan buenos como el del algoritmo Grasp propuesto. Para el caso del algoritmo Grasp desarrollado, éste cuenta con un factor de aleatoriedad, lo cual obliga a realizar una calibración de las dos constantes de relajación definidas. Gracias a esto se pueden obtener resultados mucho más precisos y con un mejor acercamiento a la mejor solución, así como delimitar el tiempo de ejecución del mismo. Mediante la calibración del algoritmo, se ha podido determinar que los valores para las constantes de relajación, tanto para pacientes como para médicos, es de 0.203 y 0.260 respectivamente, y que, además, el número de iteraciones más adecuado equivale a 6500 iteraciones.

Los resultados obtenidos a través de la experimentación numérica permiten poder afirmar que las entregas de citas médicas a través de la ejecución del algoritmo Grasp resulta mejor que aquellas realizadas bajo el modelo que actualmente se utiliza, el cual es representado mediante el esquema del algoritmo Voraz desarrollado en este proyecto.

2 Recomendaciones y Trabajos a Futuro

Si bien se ha desarrollado una herramienta que permita la asignación de citas médicas, ésta no permite realizar otras actividades propias de un centro médico, como por ejemplo, la programación de médicos, gestión de sala de operaciones, entre otros. Por lo tanto, se propone como trabajo futuro, la integración de este algoritmo a algún sistema de información que permita gestionar, de forma amplia, las actividades de los centros médicos.

Además, y debido a que en la actualidad todo tipo de reserva tiende a ser vía internet, éste algoritmo puede ser integrado al módulo de reserva de citas online de algún sistema de salud público.

Referencias Bibliográficas

Libros

[CHRISTOS,et al. 1982] Christos Papadimitriou, Kenneth. Steiglitz
1982 – "*Combinatorialoptimization*, *Algorithms and complexity*" - 1ra Edición – USA - Prentice-Hall

[DUARTE 2007] Duarte Muñoz, Abraham
2007 - "*Metaheurísticas*" - 1ra Edición – España – Dykinson

[GAREY, JOHNSON 1979] Garey, M. and D. Johnson
1979 – "Computers and Intractability; A Guide to the Theory of NP-Completeness" - 1ra Edición – USA - W. H. Freeman

[GLOVER, LAGUNA 1997] Glover, F. and Laguna, M.
1997 - "*TabuSearch*" - 1ra Edición – USA - Kluwer Academic Publishers.

[GLOVER, LAGUNA 2007] Glover, F., Laguna, M.
2007 - "*ApproximationAlgorithmsandMetaheuristics*" - 1ra Edición – USA - Chapman & Hall/CRC.

[HOLLAND 1975] Holland, John
1975 - "*Adaptation in natural and artificial systems*." Ann Arbor: The University of Michigan Press

[KELLY,et al. 1996]
Osman, I. , Kelly,H. and James, P
1996 – "Meta-heuristics : theory and applications" –1ra Edición – USA - Springer

[MITCHELL 1998] Mitchell, Melanie.
1998 – "*AnintroductiontoGeneticAlgorithms*" - 1ra Edición – Inglaterra - MIT Press

[MOSCATO 1996] Moscato, P.
1996 – "Optimización heurística y redes neuronales" – 1ra Edición – España - Paraninfo

[MULLER 1991] Muller-Merbach, H.
1991 – "*Grundlagen des Operations Researc*h" - 1ra Edición – Alemania - Springer

[PMBOK 2009] Project Management Insitute
2009 - *"A guide to The Project Management Body of Knowledge (Pmbok Guide)"* – 4ta Edición – USA - PMIPublications

[POLYA 1965] Pólya, George
1965 - "*Cómo Plantear y Resolver Problemas*" - 3ra edición - Editorial Trillas.

[RICH et al, 1990]Rich,E. and Knight, K
1990 – "*Inteligencia Artificial*" – 2da Edición - McGraw-Hill Science

[RIOS, BARD 2000] Ríos R. , Bard J.
2000 –"*Heurísticas para el secuenciamiento de tareas en líneas de flujo"*– 1ra Edición – USA - Universidad de Texas (USA)

[RUSSELL et al, 2009]Russell, S. and Norvig, P.
2009 – "Artificial Intelligence: A Modern Approach" – 3ra Edición - Prentice Hall

[SIVANANDAM, DEEPA 2010] S.N.Sivanandam, S.N.Deepa
2010 - "*Introduction to Genetic Algorithms*" - 1ra Edición – USA - Springer

[TUPIA 2009] Tupia, M.
2009 – "Fundamentos de Inteligencia Artificial" –1ra Edición – Perú -Tupia Consultores y Auditores S.A.C

[VOSS,et al 1999]
Voss,S. , Martello.S , Osman, I. and Roucairol,C.
1999 – *"Meta-heuristics: advances and trends in local search paradigms for optimization*" – 1ra Edición - Kluwer Academic Publishers

Tesis

[BEJARANO 2010] Bejarano Nicho, Gissella
2010 - "*Planificación de horarios del personal de cirugía de un hospital del Estado aplicando algoritmos genéticos (Time Tabling problem)*"
Lima/Perú -Pontificia Universidad Católica del Perú

[GALLART 2009] Gallart, Joseph
2009 – *"Análisis, diseño e implementación de un algoritmo meta heurístico GRASP que permita resolver el problema de rutas de vehículos con capacidad"*
Lima/Perú – Pontificia Universidad Católica del Perú

[RAMIREZ 2006] Ramírez Rodríguez, César
2006 - "*Algoritmo GRASP con doble relajación para resolver el problema del flow show scheduling*."
Lima/Perú - Pontificia Universidad Católica del Perú

[RODRIGUEZ 2006] Rodríguez Calvo, Ericka
2006 – *"Asignación multicriterio de tareas a trabajadores polivalentes"*
Cataluña/España – Universidad Politécnica de Cataluña

Revistas (Journal)

[ALVAREZ 2002] Alvarez, R., Crespo, E., and Tamarit, J. M.
2002 - "*Design and implementation of a course scheduling system using tabu search.*" - European Journal of Operational Research - pp. 512–523.

[BAAR, et al. 1999] T. Baar, P. Brucker, and S. Knust.
1999 – "*Tabu search algorithms for the resource constrained project scheduling problem."* - Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization - pp 1–18

[BAKER 1985]Baker, James E.
1985 – "*Adaptive Selection Methods for Genetic Algorithms*" - Proceedings of the 1st International Conference on Genetic Algorithms - pp.101-111

[BINITHA, SATHYA 2012] Binitha S, S Siva Sathya

2012 - "*A Survey of Bio inspired Optimization Algorithms*" - International Journal of Soft Computing and Engineering (IJSCE) – Volumen 2 - pp 137 – 151

[BRUCKER, et al. 1998] P. Brucker, S. Kunst, A. Schoo, and O. Thierel.
1998 - "*A branch and bound algorithm for the resource constrained project scheduling.*" - European Journal of Operational Research - Volumen 107 - pp 272-288

[COOK 1971] Cook, Stephen
1971 – "*The complexity of theorem proving procedures*" - 3rd annual ACM symposium on Theory of computing - pp 151 – 158

[DEMEESTER,et al. 2010]
Peter Demeester, Wouter Souffriau, Patrick De Causmaecker, Greet Vanden Berghe,
2010 - "*A hybrid tabu search algorithm for automatically assigning patients to beds*" - Artificial Inteligent in Medicine - Volumen 48 - pp 61-70

[DEXTER,et al. 2002]
Dexter F., Blake J., Penning D., Sloan B., Chung P., Lubarsky D.
2002 - "*Use of linear programming to estimate impact of changes in a hospital's operating room time allocation on perioperative variable costs*" – Anesthesiology – Volumen 96 - pp.718–724

[FEO, RESENDE 1995] Feo, T. and Resende, M.
1995 – "Greedy Randomized Adaptive Search Procedure" - Journal of Global Optimization – Volumen 6 – pp. 109-133

[GLOVER 1989] Glover, F.
1989 - "*Tabu Search – Part I*" - *ORSA Journal on Computing* **-** Volumen 1 - pp 90-206

[GLOVER 1990] Glover, F.
1990 - "*Tabu Search — Part II*" - *ORSA Journal on Computing* **-** Volumen 2 - *pp* 4-32.

[GOLDBERG, MILLER 1995] Brad L. Miller and David E. Goldberg
1995 – "*Genetic Algorithms, Torunament Selections, and the Effects of Noise*" - Complex Systems - Volumen 9 - pp 193-212

[GUPTA, DENTON 2008] Gupta D. and Denton B.
2008 - "*Appointment scheduling in health care: Challenges and opportunities*" - IIE Transactions - Volumen 40 - pp. 800–819

[HERTZ 1991] A. Hertz.
1991 – "*Tabú search for large scale timetabling problems.*" - European Journal of Operational Research - Volumen 54 – pp 39-47

[HOLLAND 1973] Holland, J.H.
1973 – "*Genetic algorithms and the optimal allocation of trials*" - SIAM Journal of Computing - EE.UU - Volumen 2 - pp. 88–105

[LAMIRI,et al. 2008] Lamiri M., Xie, X., Dolgui A., Grimaud F.

2008 - "*A stochastic model for operating room planning with elective and emergency demand for surgery*" - European Journal of Operational Research – Volumen 185 – pp. 1026–1037

[LIM, RODRIGUEZ 2003] H., Lim, A., and Rodrigues, B.
2003 – "*A hybrid AI approach for nurse rostering problem*" - ACM Symposium on Applied Computing – pp 730-735

[LING 1992] S. Ling.
1992 – *"Integrating Genetic Algortihms with a Prolog Assignment Problem as an Hybrid Solution for a Polytechnic Timetable Problem"* - Nature II - pp321–329.

[LITCHFIELD 2003]
Litchfield, J.A., Ingolfsson, A., Cheng, K.J.
2003 – "*Rostering for a restaurant*" - Information Systems and Operacional Research -
Volumen 41 - pp 287-300

[OGULATA,et al. 2008] Ogulata S., Koyuncu M., Karakas E.
2008 - "*Personnel and patient scheduling in the high demanded hospital services: A case study in the physiotherapy service*" - Journal of Medical Systems. Volumen 32 - pp. 221–228

[PUNNAKITIKASHEM,et al. 2008] Punnakitikashem P., Rosenberger J., Behan D.
2008 - "Stochastic Programming for Nurse Assignment" - Computational Optimization and Applications - Volumen 40 - pp. 321-349

[TUPIA 2004] Tupia, M.
2004 – "Un algoritmo GRASP para resolver el problema de la programación de tareas dependientes en máquinas diferentes" - a Conferencia Latino Americana de Informática CLEI - Volumen 1 – pp. 129 - 139

[WARNER 1972] Warner, D.M., Prawda, J.
1972 – "*A mathematical programming model for scheduling nursing personnel in a hospital"* - Management Science – Volumen 19 – pp 411-422

Web

[ESSALUD 2014]
Portal oficial de EsSalud.
Disponible en: http://www.essalud.gob.pe

[GALENUS 2014]
Galenus: Software médico especializado.
Disponible en: http://www.galenuspro.com/

[GPT 2014]
GPT: Gestión y Planificación de Turnos para Hospitales.
Disponible en: http://www.ilog.com

[INEI 2014]
Portal oficial del Instituto Nacional de Estadística e Informática.
Disponible en: http://www.inei.gob.pe

[MEF 2014]

Portal oficial del Ministerio de Economía y Finanzas.
Disponible en: http://www.mef.gob.pe

[MICROSOFT 2014]
Portal oficial de Microsoft
Disponible en: http://www.microsoft.com

[MINSA 2014]
Portal oficial del Ministerio de Salud.
Disponible en: http://www.minsa.gob.pe

[NETBEANS 2014]
Portal oficial de Netbeans
Disponible en: http://www.netbeans.org

[OMS 2014]
Portal oficial de la Organización Mundial de la Salud.
Disponible en: http://www.oms.org

[ORACLE 2014]
Portal oficial de Oracle
Disponible en. http://www.oracle.com