Inteligencia Artificial Avanzada

Scheduling Radiotherapy Problem

Felipe Nicolás Vega Valencia

15 de Abril del 2018

Abstract

En este documento se realiza un estudio sobre el Scheduling Radiotherapy Problem, el cual consiste en asignar las sesiones de tratamiento contra el cáncer de diversos pacientes a una determinada máquina (LINAC) en un determinado día cumpliendo una serie de restricciones dependiendo de la institución que realice el tratamiento. Todo esto con el objetivo de entender el problema y poder implementar un método para resolverlo. En la primera parte del documento se brinda una descripción del problema, para luego estudiar las técnicas presentes en la literatura y presentar un modelo matemático que busca reducir el tiempo de espera de los pacientes para el inicio del tratamiento[13]. En la segunda parte se presenta un acercamiento basado en Simulated Anneling y uno basado en GRASP para resolver el problema, para luego realizar diversos experimentos para comparar estos dos acercamientos, determinando es que es mejor utilizar el acercamiento basado en GRASP. Finalmente se realizan comparaciones entre GRASP y las heurísticas constructivas ASAP y JIP [12], donde se obtienen mejores resultados al usar el acercamiento presentado.

1 Introducción

La rapidez con la que se combate el cáncer una vez detectado puede ser determinante para la sobre-vivencia de un paciente, debido a la agresividad con la que ataca esta enfermedad. Uno de los métodos utilizados para el tratamiento del cáncer es la Radioterapia, técnica en la cual se utiliza una máquina llamada "Acelerador Lineal" (LINAC) la cual irradia un rayo ionizador para controlar o eliminar las células cancerigenas. Lamentablemente los recursos de los LINAC son limitados para la cantidad de pacientes que necesita recibir el tratamiento, es por esto que encontrar una buena planificación para las sesiones de tratamiento de los pacientes es de vital importancia.

El Scheduling Radiotherapy Problem busca asignar estas sesiones dentro de un horizonte de planificación buscando minimizar el tiempo de espera de un paciente entre el día en que puede iniciar su tratamiento y el día en que efectivamente este comienza, o buscando maximizar la cantidad de pacientes asignados. Para encontrar una planificación se debe seguir una serie de restricciones, las cuales varían de país en país, las que pueden considerar interrupciones en el tratamiento, la disponibilidad del paciente e incluso días de la semana en los que no se puede iniciar el tratamiento.

En la sección 2 se describe el Scheduling Radiotherapy Problem con un poco más de detalle, desde una perspectiva general y luego del problema en particular a resolver. En la sección 3 se describen los diferentes acercamientos utilizados para resolver este problema en la literatura, los cuales van desde el uso de software hasta el uso de heurísticas y algoritmos de búsqueda local. En la sección 4 se presenta un modelo matemático desarrollado por Riff, Cares y Neveu (2016)[13] para el problema particular a resolver. En la sección 5 se presentan dos algoritmos de búsqueda local, Simulated Annealing y GRASP-Simplificado respectivamente, incluyendo

la representación utilizada para ambos, la función de evaluación utilizada, la manera en la que construyen la solución inicial, el movimiento utilizado para reparar la solución inicial y los parámetros que rigen su comportamiento. En la sección 7 se describen los experimento realizados para comprobar cual de los dos acercamientos tiene mejor desempeño para presentar los resultados y análisis de partir de ellos en la sección 8. Finalmente, en la sección 9 se presentan las conclusiones en base a los resultados obtenidos y el trabajo futuro.

2 Descripción del Problema

Un problema de Scheduling consiste en encontrar una asignación óptima entre un conjunto de tareas, un conjunto de máquinas y un conjunto de recursos. Este tipo de problemas se puede encontrar en todo tipo de dominios, por ejemplo, uno de los problemas de Scheduling clásicos es el *Job Shop Scheduling*[7][9], en el cual se tiene un conjunto de tareas que se deben asignar a distintas máquinas para realizar un procesamiento óptimo. En el área de la salud también existen problemas de Scheduling, donde el más conocido es el Nurse Schduling Problem [6][8], el cual busca asignar enfermeras a diferentes turnos. Es en el área de la salud donde se encuentra el Scheduling Radiotherapy Problem.

La radioterapia es uno de los métodos más utilizados para combatir el cáncer. Para esto, se utilizan unas máquinas llamadas *aceleradores lineales* (LINAC), las cuales emiten radiación que destruye las células cancerígenas minimizando el daño a las células sanas.

En un tratamiento con radioterapia, el tiempo que pasa entre que el oncólogo determina las sesiones a realizar y la fecha en que se parte el tratamiento puede afectar la efectividad de este. Es por esta razón que el determinar una buena programación de sesiones es una tarea esencial para los centros que combaten esta enfermedad.

El ***Scheduling Radiotherapy Problem*** busca encontrar una planificación para nuevos pacientes, la cual asigne cada sesión de un paciente a un día, turno y máquina, buscando cumplir con alguno de los siguientes objetivos: minimizar el tiempo de espera entre que el paciente está preparado para la radioterapia y la primera sesión [13], maximizar la cantidad de pacientes con sus sesiones de radioterapia planificadas [2], reducir la cantidad de pacientes con planificaciones que rompen un intervalo de tiempo [11] o más de uno de los objetivos presentador anteriormente [10][12].

Las estrategias para resolver este problema tienen dos versiones. La primera, la versión ***Block*** del problema, considera que el día se separa en bloques de largo fijo (10 - 15 minutos). La segunda es la versión ***Non-Block***, en la cual se consideran largos variables para las divisiones en un mismo día.

Si bien, los supuestos que existen al momento de armar una programación de sesiones varía dependiendo del centro médico o del país, hay algunas consideraciones que son comunes.

- Se tiene un horizonte de planificación, en el cual se programarán las sesiones de un conjunto de pacientes.
- Cada paciente tiene una prioridad distinta, dependiendo del grado de cáncer que este padezca.
- Existe un conjunto de máquinas para realizar radioterapia, las cuales tienen una capacidad diaria definida.
- Solo se puede atender un paciente por máquina al mismo tiempo.
- El número de sesiones y la duración de estas se conocen de antemano.
- Un paciente no puede tener más de una sesión diaria.

- Cada paciente tiene una fecha en la que puede empezar con las sesiones de radioterapia ***(Release Time)*** y una fecha máxima para el inicio de la primera sesión ***(Due Time)***.

Dentro de las consideraciones propias de cada centro, se encuentra el hecho de tener máquinas de distinta energía, la cantidad de categorías (prioridades) en la que puede estar un paciente, la manera en la que se asignan las sesiones del tratamiento (días consecutivos o permitiendo interrupciones), etc.

3 Estado del Arte

Los problemas de *Scheduling* consisten en encontrar una asignación óptima entre un conjunto de tareas, un conjunto de máquinas y un conjunto de recursos. Estos problemas han sido ampliamente estudiados en la literatura y existen en diversos dominios de problemas.

Uno de los ejemplos más clásicos de los problemas de Scheduling es el *Job Shop Scheduling Problem*, en el cual se tiene un conjunto de tareas que se deben asignar a un conjunto de máquinas para realizar todo el procesamiento de la manera más óptima posible. Este es un problema que ha sido abarcado utilizando diversas técnicas, donde entre las más recientes se encuentra un método basado en el comportamiento de la *hierba invasora* (Mishra (2017))[7] y el uso de *PSO (Particle Swarm Optimization)* distribuido (Nouiri (2018))[9].

Dentro del dominio de la salud también existen varios problemas de Scheduling, donde destaca el denominado *Nurse Scheduling Problem (NSP)*, el cual busca programar los turnos de un conjunto de enfermeras en los diferentes turnos disponibles, respetando una serie de restricciones fuertes, dependiendo de la institución en la cual se realiza la programación y restricciones débiles, asociadas las enfermeras en si. Debido a la dificultad del NSP es que ha sido abarcado con distintas técnicas y heurísticas, como PSO [8] o algoritmos genéticos [6]. Una de las últimas técnicas presentadas para el NSP es el uso de una heurística de 2 niveles (Senbel (2018)[15]), en la cual se genera una programación que cumpla con todas las restricciones fuertes ordenadas por prioridad, para luego optimizar dicha programación para satisfacer 3 restricciones débiles sin romper las restricciones fuertes existentes.

Si bien, los problemas del dominio de la salud como el *NSP*[8][6][15] y el *Surgery Scheduling Problem* (programación de cirugías)[3] han sido estudiados ampliamente, el *Scheduling Radiotherapy Problem* solo presenta unos pocos estudios, principalmente debido a que las restricciones que este presenta son bastante más complejas.

Los enfoques para el *Scheduling Radiotherapy Problem* se pueden separar en dos áreas. El desarrollo de modelos matemáticos para su posterior resolución mediante el uso de Solvers y el desarrollo de algoritmos y metaheurísticas. A continuación se analizan distintas técnicas utilizadas para ambos enfoques.

3.1 Uso de Solvers

Conforti (2010)[2], propone un modelo para resolver el problema en una versión *Non-Block*. El modelo matemático contempla un horizonte de planificación de 6 días, considera pacientes que ya han iniciado su tratamiento y pacientes que están en espera para iniciar sus sesiones. El modelo permite que pacientes que ya iniciaron sus sesiones puedan ser re-programados basados en su disponibilidad. Además obliga a que un paciente siempre sea atendido con la misma máquina.

Lo bueno que tiene el modelo propuesto es que es fácil de adaptar. Entre las desventajas que posee es que no está hecho para entregar resultados en tiempo real y que en el peor de los casos no asigna a ningún paciente.

Se realizan experimentos utilizando datos del gobierno de Nueva Zelanda (`http://www.moh.govz.nz/cancerwaitingtimes`), donde utilizan 6 instancias distintas donde consideran 2 máquinas, con 2 turnos diarios y 116 pacientes (76 que ya iniciaron su tratamiento y 40 en espera).

Resuelven utilizando el *MIP solver of CPLEX 10.0*, y obtienen que se utiliza la capacidad máxima de las máquinas. Obtienen resultados entre 1.94 y 733 segundos, por lo que se podría utilizar en la práctica.

En el año 2010, Jacquemin [5] propone un modelo que mejora las falencias del modelo presentado por Conforti [2]. Expanden el horizonte de planificación a 15 semanas y redefinen la función para calcular el peso de cada paciente, también agregan la posibilidad de trabajo a sobre-horas para poder atender a pacientes que ya iniciaron su tratamiento. Es por esto que a las 17 restricciones del modelo de Conforti le agregan restricciones para que no se sobrepase el tiempo disponible por máquina, se utilicen o no las horas extras. También agregan una penalización en la función objetivo para cuando se utilicen horas extras.

Dado que existe la posibilidad de extender un turno (horas extras), se debe considerar la disponibilidad de los pacientes desde el momento al que ingresan al sistema, por lo que agregan un parámetro que entregue la disponibilidad de los pacientes en espera para el inicio de su tratamiento.

Finalmente, toman en consideración la disponibilidad de los radioterapeutas, dando la posibilidad de que un paciente comience su tratamiento con un radioterapeuta distinto. De ocurrir esta situación, la primera sesión demorará más de lo habitual, por lo que se agrega una nueva penalización en la función objetivo.

Para realizar pruebas utilizan datos basados en un centro oncológico Frances (*Centre Léon Bérard, CLB*). Utilizan el solver *LINGO©* para encontrar las soluciones del modelo. Como se tenía un horizonte de planificación de 15 semanas, los tiempos de resolución eran bastante altos, por lo que solo se permitía que se buscara solución durante 240 segundos para cada resolución del modelo.

Los resultados que obtienen muestran que las decisiones tomadas llevan a una mejora en la calidad de la atención y en el uso de recursos en comparación al modelo de Conforti.

Por último, Jacquemin (2011)[14] propone un nuevo modelo para mejorar los resultados obtenidos el 2010 [5]. En los resultados obtenidos en modelos anteriores, todas las sesiones se acumulaban en los primeros días de la semana, dejando tiempo libre al final de esta. Es por esto que desarrolla un modelo que utiliza patrones pre-definidos para la programación de sesiones.

Si el número de sesiones por pacientes es menor a 5, el patrón que se sigue es el de tener dos sesiones en días consecutivos, seguidos de un descanso de un día, para luego continuar con las sesiones. El otro patrón es cuando el número de sesiones es igual a 5, donde se programan todas las sesiones a lo largo de la semana.

Para realizar experimentos utilizan datos generados de manera aleatoria. Nuevamente utilizan instancias que cuentan con dos máquinas y un horizonte de planificación de 15 semanas. Utilizan *LINGO©* para resolver, teniendo las mismas limitaciones de tiempo que en el modelo anterior [5].

Comparan este nuevo modelo con el modelo anterior utilizando 3 indicadores de calidad.

1. Porcentaje de pacientes que iniciarán su tratamiento.

2. Número de semanas que debe esperar un paciente antes de iniciar su tratamiento.

3. Porcentaje de pacientes que inician su tratamiento con otro radioterapeuta.

Los resultados obtenidos indican mejoras en los tres indicadores de calidad, además se obtienen soluciones que privilegian más la disponibilidad de los radioterapeutas sin bajar la calidad de las soluciones encontradas. Sin embargo, al llevar el problema a instancias del mundo real (8 máquinas), el uso de solvers no es suficiente para encontrar soluciones.

3.2 Heurísticas y Metaheurísticas

3.2.1 Enfoques Constructivos y GRASP

Petrovic (2006)[12] y (2008)[11] desarrolla diversos enfoques constructivos para construir la programación de sesiones para pacientes de radioterapia. Para todos los enfoques utiliza como supuesto una separación de pacientes en tres categorías ***(Radical, Palliative y Emergency)*** donde solo los pacientes en la categoría *Emergency* pueden ser atendidos el fin de semana. También asume que los pacientes con menos de 5 sesiones tienen todas sus sesiones dentro de la misma semana.

Primero propone los enfoques ***ASAP*** (As Soon As Posible) y ***JIP*** (Just In Time), estos métodos cuentan con dos fases. Primero se ordenan los pacientes de acuerdo a la categoría a la que pertenecen y luego por *Due Time*. Por último, se programan las sesiones siguiendo diferentes criterios. Para el primer enfoque intentan generar una planificación desde el *Release Time* del paciente, donde si no encuentra una asignación válida continua con el siguiente día. Para el segundo se trata de asignar desde el día anterior al *Due Time*, donde en caso de no encontrar una asignación válida intenta asignarlo en el día anterior. El criterio para comparar la efectividad de estos acercamientos es minimizar la cantidad de pacientes que inician su tratamiento después del *Due Time*, además de minimizar el *Waiting Time*. Para realizar experimentos se basan en datos reales de *Nottingham University Hospitals NHS Trust*. Los resultados indican que *JIP* funciona mejor que *ASAP*.

Luego, en el año 2008, Petrovic [11] busca generalizar los métodos *ASAP* y *JIP*. Por lo que genera cuatro nuevos enfoques constructivos y una técnica ***GRASP***. En el primero intenta asignar las sesiones desde un día pre-definido por una constante llamada ***Target*** dentro del intevalo [*Release Time*, *Due Time*]. Al probar este enfoque con distintos *Target* obtiene que es mejor no utilizarlo.

En el segundo enfoque se agrega un umbral máximo de utilización a cada máquina para cada categoría de paciente, es decir, cuando se alcanza dicho umbral no se siguen asignado más pacientes de dicha categoría en dicha máquina. Al experimentar con este enfoque determina que el umbral para los pacientes de la categoría *Emergency* debe ser del 100% de la capacidad de la máquina y para las otras categorías debe ser de un 90%.

Para el tercer enfoque utilizan la información obtenida de los dos enfoque anteriores y agregan días específicos en los que se puede comenzar a planear la programación para un paciente. Los resultados que obtienen con este acercamiento indican que es mejor iniciar la asignación los días lunes, miércoles y viernes.

En el cuarto enfoque, definen una ventana de tiempo en la que es aceptable asignar la primera sesión de un tratamiento antes del *Due Time* para cada categoría de paciente. También integran los resultados obtenidos en los tres enfoques anteriores. Las pruebas realizadas con este último enfoque indican que es mejor esperar un día antes de comenzar con la planificación de un paciente.

Por último, diseña un algoritmo ***GRASP*** para intentar mejorar los resultados encontrados por la mezcla de los cuatro enfoques. Primero se ordena la lista de acuerdo a las prioridades en cada paciente y se genera una solución usando los enfoques descritos anteriormente. Luego, desde la lista ordenada se sacan elementos utilizando una distribución exponencial, de esta manera generan una nueva solución. Esta nueva solución la intentan mejorar usando búsqueda local con swap como movimiento, entre un paciente que haya sobrepasado su *Waiting Time* con alguien que haya sido asignado antes.

Al comparar la solución encontrada con el algoritmo *GRASP* con la solución encontrada con los enfoques constructivos, obtienen que *GRASP* mejora la solución en un 38% de los casos, la mantiene igual en un 39% y la empeora en un 23%.

En los experimentos utilizaron datos generados a partir de datos reales pertenecientes a *Nottingham University Hospitals NHS Trust*. No se presentan datos sobre el tiempo de ejecución de los algoritmos.

3.2.2 Algoritmo Genético Multi-objetivo

En el año 2011, Petrovic [10] busca encontrar una planificación buscando minimizar el ***Waiting Time*** (tiempo de espera) y minimizar la cantidad de pacientes que inician su tratamiento pasado el *Due Time*, considerando los siguientes supuestos (aparte de los definidos en la sección 2).

- Se trabaja estrictamente de lunes a viernes.
- No se puede interrumpir un tratamiento.
- Existen tres categorías de pacientes.
  1. ***Radical***, con un *waiting time* de 28 días.
  2. ***Palliative***, con un *waiting time* de 14 días.
  3. ***Emergency***, con un *waiting time* de 2 días.
- El tratamiento se realiza durante días consecutivos, sin contar fines de semana y feriados.
- Las máquinas siempre están disponibles (no se realiza mantención).

Para esto, propone tres algoritmos genéticos; ***Standart-GA***, el cual considera que todos los pacientes tienen la misma prioridad, ***KB-GA***, el cual añade conocimiento del dominio para la programación de los pacientes de la categoría *Emergency* y ***Weighted-GA*** el cual considera un peso para cada categoría.

Como *representación* se utiliza un string donde almacena el orden en que serán asignadas las sesiones de cada paciente. El largo del string viene dado por la cantidad de pacientes multiplicada por el número máximo de sesiones de todos los pacientes. Además se apoyan de dos matrices, las cuales contendrán la información de que máquina se usará para cada paciente en cada sesión y el tiempo utilizado por sesión respectivamente. Lo bueno que tiene esta representación es que permite trabajar solo con soluciones factibles, pero a costas de tener que decodificar cada vez que se quiera evaluar la calidad de una solución.

La *función de aptitud* considera los dos objetivos a minimizar, normalizando los valores para que esta entregue resultado entre 0 y 1.

Para generar la *población inicial* utilizan un ordenamiento fijo (todas las sesiones juntas por paciente, o todas las sesiones 1 juntas), para luego generar ordenamientos aleatorios a partir de ellos hasta completar el tamaño de población.

Se usa *elitismo* para la *selección*, pasando una porción de los mejores individuos directamente a la siguiente generación. Después se aplican los operadores de *mutación* y *cruzamiento* y se eligen los individuos con mejor función de aptitud para completar la próxima generación.

Como operador de *cruzamiento* se utiliza ***Linear Order Crossover***, el cual consiste en seleccionar una porción de pacientes en cada padre, los cuales mantendrán sus sesiones en la misma posición original. Con los pacientes restantes se arma un sub-string, al cual se le realizan tres shift hacia la izquierda y luego se agregan a los hijos en el orden en el que aparecen. Para el operador de *mutación* se utiliza la misma estrategia pero utilizando un solo shift.

Para la versión *KB-GA* se modifican los operadores para que se coloquen las sesiones de los pacientes de la categoría *Emergency* en las primeras posiciones del string. En el caso de la versión *Weighted-GA* se altera la función de aptitud agregando el peso de cada categoría.

Para realizar experimentos se generan instancias a partir de las características de datos reales pertenecientes a *Arden Cancer Centre* del Reino Unido. Se comparan los tres algoritmos genéticos para cada categoría de paciente por separado y considerando las tres al mismo tiempo. En todos los experimentos el que mejores resultados entrega es el *Weighted-GA*, pero en el caso en que se consideran las tres categorías la mejoría en resultados no es tan significativa. No se entregan los tiempos de ejecución de los algoritmos.

3.2.3 RASON

Riff, Cares y Neveu (2016)[13] presentan ***RASON***, un algoritmo de búsqueda local que permite resolver instancias con restricciones del mundo real, considerando el *Waiting Time* actualizado de cada paciente. Los supuestos y restricciones que consideran son las mismas que las consideradas por Petrovic (2006)[12].

RASON considera dos fases para encontrar una solución. En la primera se construye una solución inicial utilizando un algoritmo *Greedy*, en el cual se ordenan los pacientes a ingresar y los pacientes en lista de espera. Una vez ordenados los pacientes, se procede a tratar de asignar a los pacientes de la lista siguiendo el siguiente criterio: *"Si el paciente no es de categoría Urgente y no se pueden programar sus sesiones en dias consecutivos, se ingresa al paciente a la lista de espera. Si no, se intenta programar las sesiones del paciente siempre que se tenga disponibilidad de máquinas"*.

Una vez generada la solución inicial, se mejora utilizando tres *Hill Climbing* (HC) con distintos movimientos para minimizar el tiempo de espera de los pacientes. El primer *HC* utiliza dos movimientos para mejorar la solución: Swap entre un paciente de la lista de espera y uno con sus sesiones asignadas, y el segundo es intentar programar un paciente de la lista de espera siempre que sea posible. El segundo HC utiliza un swap entre pacientes ya asignados como movimiento. Los dos HC anteriores producen espacios vacíos en la planificación, por lo que el tercer HC intenta rellenar dichos espacios adelantando las sesiones de un paciente.

Para sintonizar parámetros utilizan instancias generadas con ***GeneRa*** (2016)[1], un generador de instancias creado por ellos mismos. Comparan los resultados obtenidos con *RASON* con los resultados obtenidos por *ASAP* y *JIP*[12] utilizando datos pertenecientes al *Radiotherapy Institute IRAM* de Chile. Los resultados de la comparación muestran que *RASON* logra encontrar programaciones de calidad con un tiempo de espera menor que el encontrado por las otras dos técnicas. No se reporta el tiempo de ejecución del algoritmo.

4 Modelo Matemático

Los modelos matemáticos propuestos para este problema generalmente tienen uno de dos enfoques de optimización; maximizar la cantidad de pacientes que inicien su tratamiento dentro del horizonte de planificación[2], o minimizar el tiempo de espera entre que el paciente puede iniciar su tratamiento y la primera sesión[13].

A continuación se presenta un modelo del segundo enfoque, el cual busca minimizar el *Waiting Time* (tiempo de espera) de los pacientes (Riff, Cares y Neveu (2016)[13]), lo que aumenta la efectividad del tratamiento recibido por los pacientes.

4.1 Minimizar el Waiting Time [13]

Este modelo originalmente considera el problema como un ***CSP*** (Constaint Satisfaction Problem), por lo que no cuenta con función objetivo. Para minimizar el *Waiting Time* se le agrega una función objetivo utilizada por Riff, Cares y Neveu en su algoritmo ***RASON*** [13].

Se tienen las siguientes consideraciones:

- Existen máquinas de alta energía y baja energía.
- No hay sesiones el fin de semana.
- Cada paciente puede pertenecer a 1 de 3 categorías *(Radical, Palliative y Urgent)*.
- Cada paciente tiene un número de interrupciones máximas, las que no incluyen el fin de semana.

- El tipo de máquina con la que se trata cada paciente depende de la categoría a la que pertenezca.
- Los pacientes deben tener al menos dos sesiones programas para la semana en la que inician su tratamiento.

4.1.1 Parámetros

Parámetro	Descripción
N	Número de pacientes
H	Número de máquinas de alta energía
L	Número de máquinas de baja energía
r_i	El primer día en que el paciente i puede iniciar su tratamiento
d_i	El último día en que el paciente i puede iniciar su tratamiento
F_i	Número de sesiones requeridas por el paciente i
V_h	Número máximo de sesiones para una máquina de alta energía
V_l	Número máximo de sesiones para una máquina de baja energía
I_i^{max}	Número máximo de interrupciones permitidas para el paciente i
H_i	1 si el paciente i requiere de una máquina de alta energía, 0 si no.
L_i	1 si el paciente i requiere de una máquina de baja energía, 0 si no.
T_i	1 si el paciente i es de tipo $T \in \{\text{A = Urgent, B = Palliative, C = Radical}\}$ 0 si no.
W	Primer día de la programación ($1 = Lunes, 2 = Martes, ...$)

Table 1: Parámetros del modelo.

4.1.2 Variables

$$x_{ik} = \begin{cases} 1 & \text{Si el paciente } i \text{ tiene sesión el día } k. \\ 0 & \text{Si no.} \end{cases} \tag{1}$$

4.1.3 Función Objetivo

$$MIN \sum_{i=1}^{N} (s_i - r_i) \tag{2}$$

Donde s_i es el día donde el paciente i inicia su tratamiento, lo que busca esta función objetivo es minimizar la suma de los *Waiting Time* de todos los pacientes.

4.1.4 Restricciones

La función objetivo (2) está sujeta a las siguientes restricciones.

$$x_{ik} = 0 \quad \forall k = 1, 2, ..., D \text{ sujeto a } \texttt{week\_day(k)} \in [6, 7] \tag{3}$$

La restricción (3) se aplica solamente cuando la función `week_day(k)` retorna 6 o 7. Esto quiere decir que el día k es sábado y domingo. Por lo que la restricción se encarga de no asignar sesiones en un fin de semana.

Por otro lado, la función `week_day(k)` se define de la siguiente manera.

$$\texttt{week\_day(k)} = ((W + k) \bmod 7) + 1 \tag{4}$$

Con dicha definición, se obtiene un 1 para el día lunes, un 2 para el día martes y así para el resto de los días de la semana.

$$\sum_{k=r_i}^{d_i} x_{ik} \geq 1 \quad \forall i = 1, ..., N \tag{5}$$

$$\sum_{k=r_i}^{D} = F_i \quad \forall i = 1, ..., N \tag{6}$$

Con las restricciones (5) y (6) se controla que las sesiones partan desde el día r_i y que se cumplan todas las sesiones necesarias para cada paciente.

Para restringir el número de interrupciones máximas de un paciente primero se deben hacer las siguientes definiciones.

$$Fl_i = F_i + I_i^{max} \tag{7}$$

$$WE_i(k) = \left\lceil \frac{Fl_i - (6 - \texttt{week\_day(k)})}{5} \right\rceil \tag{8}$$

$$k_i^{max}(k) = k + Fl_i + 2 \cdot (WE_i(k) - 1) \tag{9}$$

Donde Fl_i es el número máximo de interrupciones del paciente i, $WE_i(k)$ es el número de fines de semana que hay entre el día k y el día k^{max}, que es el día máximo para la última sesión.

Con las definiciones anteriores se puede restringir que no se realicen sesiones que produzcan un número mayores de interrupciones que las permitidas.

$$(x_{ik} = 1) \implies \left(\sum_{k'=k_i^{max}+1}^{D} x_{ik'} = 0 \right) \quad \forall i = 1, ..., N, \forall k = 1, ..., D \tag{10}$$

Luego, con la restricción (11) se obliga a que en la semana donde inicia el tratamiento se tengan al menos dos sesiones.

$$\sum_{k=k'}^{k'+4} = 1 \implies \sum_{k=1}^{k'-1} x_{ik} \geq 2 \quad \forall k' = 1, ..., D. \text{ Sujeto a } \texttt{week\_day(k)} = 1 \tag{11}$$

Finalmente restringe que no se sobrepase la capacidad de las máquinas.

$$\sum_{i=1}^{N} H_i \cdot x_{ik} \leq V_h \quad \forall k = 1, ..., D \tag{12}$$

$$\sum_{i=1}^{N} L_i \cdot x_{ik} \leq V_l \quad \forall k = 1, ..., D \tag{13}$$

5 Propuesta

Para resolver el problema se proponen dos algoritmos con el objetivo de minimizar el tiempo de espera de los pacientes asignados. El primero es un Simulated Annealing con recalentamiento, mientras que el segundo es una modificación de un algoritmo de tipo GRASP utilizando el Simulated Annealing anterior para la fase de post-procesamiento utilizando un movimiento enfocado en disminuir los tiempos de espera. Ambos algoritmos utilizan la misma representación y la misma función de evaluación.

5.1 Representación

Como representación se utilizan dos vectores de pacientes, uno de pacientes con sus sesiones asignadas y otro con pacientes cuyas sesiones no se asignaron durante la ejecución del algoritmo. Cada paciente cuenta un con vector binario de largo igual a los días del horizonte de planificación para representar las sesiones que tiene asignadas. La casilla i será 1 si el paciente tiene asignada una sesión en el día i y 0 si no. Un ejemplo de la planificación de un paciente se muestra en la tabla 2, donde se considera un horizonte de planificación de 5 días y un paciente con tres sesiones. En este caso, el paciente tiene asignadas sus sesiones en los días 3, 4, y 5.

i	1	2	3	4	5
Sesiones	0	0	1	1	1

Table 2: Ejemplo de la planificación de un paciente con 3 sesiones en un horizonte de planificación de 5 días.

Además, se utiliza un vector de enteros de largo igual a la cantidad de máquinas multiplicada por los días del horizonte de planificación. Este vector tiene como finalidad mantener contadores con la capacidad restante cada máquina en cada día, con el objetivo de no sobrepasar la capacidad diaria de una máquina, ayudando así a mantenerse en el espacio de soluciones factibles.

5.2 Función de Evaluación

La función de evaluación utilizada para ver la calidad de las soluciones se basa en calcular el tiempo de espera promedio (en días) considerando solo los pacientes con sus sesiones asignadas.

$$\frac{\sum_{\text{Paciente} \in \text{Asignados}} \text{DiaInicioTratamiento} - \text{ReleaseDay}}{\#\text{Asignados}}$$

El objetivo de los dos algoritmos presentados en minimizar el tiempo de espera promedio sin disminuir la cantidad de pacientes asignados, por lo que se fija una cantidad mínima de pacientes igual a la cantidad de pacientes asignados al momento de generar la solución inicial.

5.3 Constructor de Soluciones Aleatorias

Para construir soluciones se ordenan los pacientes por release day ascendente, donde si hay empate se ordenan por categorías (desde urgente a radical) y en último caso por cantidad de sesiones descendente. El último criterio se elige de tal manera para favorecer la asignación de pacientes con un alto número de sesiones, puesto que será más difícil asignarlos a medida que avanza el proceso de construcción.

La manera en que se asignan las sesiones depende un número aleatorio el cual controla el criterio de asignación. Esto con el objetivo de poder construir soluciones aleatorias que permitan

explorar más el espacio de búsqueda. Para los criterios de asignación se utilizan dos métodos basados en los algoritmos constructivos ASAP y JIP [12] descritos en la sección 3. Estos dos métodos intentan asignar todas las sesiones de un solo paciente; en el caso de ASAP se intenta asignar desde el primer día, donde de no se posible se continua con el día siguiente, hasta llegar al último día posible, y en el caso de JIP se intenta asignar desde el último día y de no ser posible se intenta con el día anterior hasta llegar al primer día posible para el inicio del tratamiento. Estos métodos se apoyan del vector que mantiene la capacidad de las máquinas y están diseñados para no romper las restricciones descritas en la sección 2, con el objetivo de siempre trabajar con soluciones factibles.

El algoritmo 1 muestra el pseudo-código del algoritmo de construcción de soluciones aleatorias.

Algoritmo 1 ConstructorSolucion

```
1: Entrada:ListaPacientes
2: ListaPacientes.sort()
3: Asignados = Vector<Pacientes>()
4: NoAsignados = Vector<Pacientes>()
5: capacidadMaquinas = Vector<int>(dias * cantidadMaquinas, tiempo)
6: for Paciente in ListaPacientes do
7:    randomNumber ~ U(0, 1)
8:    if randomNumber < heuristicParam then
9:       ASAP(Paciente, capacidadMaquinas, Asignados, NoAsignados)
10:   else
11:      JIP(Paciente, capacidadMaquinas, Asignados, NoAsignados)
12:   end if
13: end for
14: Return  Asignados, NoAsignados, CapacidadMaquinas
```

5.4 Movimiento

El movimiento diseñado está pensando en que si se des-asignan las sesiones de un paciente con una alta cantidad de sesiones se podrán asignar las sesiones de pacientes con una cantidad menor de sesiones minimizando el tiempo de espera. En el algoritmo 2 se presenta el pseudo-código del movimiento.

Al comienzo del movimiento se ordenan los pacientes asignados por cantidad de sesiones descendiente, en caso de empate se ordenan por tiempo de espera descendiente y finalmente por release day ascendiente. La elección de un orden descendiente para la cantidad de sesiones y el tiempo de espera se realiza considerando la implementación de los vectores en C++, dado que está optimizado para sacar elementos del final de este.

Una vez ordenada la lista de pacientes asignados se genera un número aleatorio siguiendo una distribución uniforme discreta entre 0 y un parámetro con valor fijo (**"Param1"**), este número define la cantidad de pacientes que se quitaran de la lista de asignados y serán ingresados a la lista de pacientes no ingresados. Para evitar posibles loops de pacientes entre las listas se genera un número aleatorio siguiendo una distribución uniforme entre 0 y 1, si el valor generado es menor al parámetro **"paramProb"** se revuelve el final de la lista, para así reducir el riesgo de caer en un loop. Cuando se elimina un paciente de la lista de asignados se utiliza el método **"Recalculador"**, el cual actualiza la capacidad de la máquina utilizada por el paciente en los días que tenía sesión.

Antes de seleccionar los pacientes que se intentarán asignar se ordena la lista de no asignados por sesiones ascendentes. Luego, se trata de asignar un cuarto de los pacientes no asignados

usando el método ASAP, dado que así se puede intentar minimizar el waiting time del paciente asignado.

Algoritmo 2 Movimiento

```
1: nuevoAsignados = asignadosActual
2: nuevoNoAsignados = noAsignadosActual
3: nuevaCapacidad = capacidadActual
4: nuevoAsignados.sort()
5: randomNumber ~ U(0, param1)_discreta
6: for int i = 0; i < randomNumber; i++ do
7:    randomProb ~ U(0, 1)
8:    if randomProb < paramProb then
9:       nuevoAsignados.shuffle(size/2,end)
10:   end if
11:   PacienteDesasignado = nuevoAsignados.pop()
12:   recalculador(nuevaCapacidad, PacienteDesasignado)
13:   nuevoNoAsignados.push(PacienteDesasignado)
14: end for
15: noAsignados.sort()
16: for int j = 0; j < NoAsignados.size()/4; j++ do
17:   randomNumber ~ U(0, NoAsignados.size())
18:   PacientePorAsignar = nuevoNoAsignados.pop(randomNumber)
19:   ASAP(PacientePorAsignar, nuevaCapacidad, nuevoAsignados, nuevoNoAsignados)
20: end for
21: Return nuevoAsignados, nuevoNoAsignados, nuevaCapacidadMaquinas
```

5.5 Simulated Annealing

Simulated Annealing (**SA**) es una técnica de búsqueda local reparadora, la cual permite aceptar soluciones de peor calidad bajo una cierta probabilidad asociada a un parámetro que modela la temperatura del sistema. Esta versión de SA utiliza recalentamiento completo cuando se baja de cierto umbral de temperatura, esto con el objetivo de poder diversificar en base a la mejor solución encontrada y así escapar de posibles estancamientos en óptimos locales. El algoritmo 3 muestra un pseudo-código del Simulated Annealing implementado.

Algoritmo 3 Simulated Annealing

```
1: solucionInicial = ConstructorSolucion()
2: solucionActual = solucionInicial
3: mejorSolucion = solucionInicial
4: cantidadMinima = solucionActual.asignados.size
5: for i = 0; i < iter; i++ do
6:    for j = 0; j < iterTemp; j++ do
7:       nuevaSolucion = movimiento(solucionActual)
8:       delta = FuncionEvaluacion(solucionActual) - FuncionEvaluacion(nuevaSolucion)
9:       if delta > 0 and nuevaSolucion.asignados.size ≥ cantidadMinima then
10:         solucionActual = nuevaSolucion
11:      else, if randomNumber ~ U(0,1) < e^(delta/temp) then
12:         solucionActual = nuevaSolucion
13:      end if
14:      if solucionActual < mejorSolucion and solucionActual.asignados.size ≥ cantidadMinima then
15:         mejorSolucion = solucionActual
16:      end if
17:   end for
18:   temp = temp * multiplicador
19:   if temp < 0.00001 then
20:      SolucionActual = MejorSolucion
21:      temp = tempInicial
22:   end if
23: end for
24: Return mejorSolucion
```

6 GRASP-simplificado

Si bien, el algoritmo SA presentado minimiza el tiempo de espera de los pacientes, no aprovecha la aleatoriedad del la construcción de soluciones porque solo la utiliza una vez, es por esto que se diseña un algoritmo GRASP simplificado para sacar provecho de la construcción de soluciones. Un algoritmo GRASP se enfoca en construir soluciones de alta calidad para después mejorar para encontrar algo aún mejor.

Una de las diferencias que tiene este enfoque con respecto a un GRASP clásico es que no considera una lista restringida de candidatos (LRC), si no que siempre se asignarán los pacientes en el orden en que aparecen. El criterio con los que se ordenan los pacientes en la lista sigue los diseños que debería tener la función miope al elegir el paso *Greedy*. Al no tener LRC la aleatoriedad de la fase constructiva viene dada por el hecho de elegir si asignar al paciente utilizando ASAP o JIP.

En la fase de post-procesamiento se utiliza el Simulated Annealing presentado anteriormente para minimizar los tiempos de espera utilizando los espacios que se crean al momento de utilizar JIP para asignar un paciente.

El algoritmo 4 presenta el pseudo-código de este algoritmo. Esta versión no considera una fase de pre-procesamiento, algo que se puede agregar en un trabajo futuro para detectar pacientes que al ingresarlos primero se obtengan soluciones de mejor calidad.

Algoritmo 4 GRASP-simplificado

```
1: solucionInicial = ConstructorSolucion()
2: mejorSolucion = funcionEvaluacion(solucionInicial)
3: for i = 0; i < Iteraciones; i++ do
4:    nuevaSolucion = ConstructorSolucion()
5:    nuevaSolucion = SimulatedAnnealing(nuevaSolucion)
6:    if nuevaSolucion < mejorSolucion then
7:       mejorSolucion = nuevaSolucion
8:    end if
9: end for
10: Return mejorSolucion
```

6.1 Parámetros

En la tabla 3 se presentas los parámetros de los componentes anteriores junto a su respectiva descripción. Los parámetros "Param1", "ParamProb" y "heuristicParam" están dedicados a manejar la parte de diversificación de los acercamientos. Para los primeros dos parámetros anteriores valores muy altos (dentro de su dominio) implican una alta diversificación en la búsqueda.

Parámetro	Descripción
Iter	Número de iteraciones del algoritmo Simulated Annealing
Param1	Número de pacientes que serán cambiados desde la lista de asignados a la de no asignados
ParamProb	Probabilidad de desordenar el fin la lista de pacientes asignados al momento de sacar un paciente
Temp	Temperatura inicial del algoritmo Simulated Annealing
Multi	Factor por el cual disminuye la temperatura en Simulated Annealing
IterTemp	Número de iteraciones con temperatura constante en Simulated Annealing
heuristicParam	Probabilidad de asignar un paciente con el método basado en ASAP
iterGrasp	Número de iteraciones para el algoritmo GRASP

Table 3: Parámetros utilizados por los distintos algoritmos con su respectiva descripción

En la tabla 4 se presentan los valores utilizados por parámetro al momento de correr los algoritmos. Cabe destacar el valor 1 para la temperatura inicial, el cual se debe al bajo valor que presenta la diferencia entre soluciones al momento de evaluar aceptar una solución de peor calidad en SA, lo que provoca que siempre se acepten soluciones de peor calidad, por lo que un valor pequeño para la temperatura inicial ayuda a corregir esta situación.

Parámetro	Rango de Valores utilizados
Iter	[0, 50, 100, 500]
Param1	[1, 2, 5, 10]
ParamProb	[0.2, 0.4, 0.6]
Temp	[1, 50, 100]
Multi	[0.8, 0.9, 0.98]
IterTemp	[10, 50, 70, 100]
heuristicParam	[0.5, 0.7, 0.9]
iterGrasp	[25 ,50 ,100]

Table 4: Parámetros utilizados por los distintos algoritmos junto a su rango de valores utilizados.

7 Experimentos

Se realizan dos tipos de experimentos con los acercamientos propuestos. La primera parte para ver el desempeño de ambos acercamientos, por lo que se hicieron pruebas referentes a la calidad de la solución y la convergencia, para cada uno de los acercamientos. No se realizaron comparaciones temporales debido a que GRASP Simplificado utiliza internamente el Simulated Annealing. La segunda parte de los experimentos consiste en una comparación con los algoritmos constructivos ASAP y JIP [12] presentes en la literatura.

7.1 Parámetros Utilizados

En la tabla 5 se presentan los valores de los parámetros utilizados para realizar los experimentos. No se ocupó ninguna técnica de sintonización de parámetros, debido a que es un proceso costoso en tiempo. A futuro se planea utilizar ParamILS [4] para encontrar una mejor configuración.

Parámetro	***Valor***
Iter	100
Paran1	2
ParamProb	0.4
Temp	1
Multi	0.8
IterTemp	70
GRASPIter	50
HeuristicParam	0.8

Table 5: Valor de los parámetros utilizados para los experimentos.

7.2 Instancias Utilizadas

Para realizar los experimentos se utilizaron instancias con las configuraciones presentadas en [13], las cuales fueron generadas usando *GeneRa* [1]. En particular se utilizaron instancias de 60 días con máximo 30 y 10 pacientes a asignar por día (**60_30** y **60_10**) y de 30 días con la misma cantidad de pacientes como máximo de las instancias anteriores (**30_30** y **30_10**).

Por cada categoría de paciente se utilizan tres sesiones distintas, cada una asociada a una cierta probabilidad, por ejemplo, los pacientes de categoría Urgente pueden tener una, dos o cuatro sesiones. La configuración de la cantidad de sesiones se presenta en la tabla 6. Además, el tipo de máquina que utiliza cada categoría tiene asociada una probabilidad. En este caso los pacientes de categoría Urgente usan solo máquinas de alta energía, los paciente paliativos solo utilizan máquinas de baja energía y los paciente radicales pueden usar de ambas.La configuración del tipo de máquina se presenta en la tabla 7.

Urgente		**Paliativo**		**Radical**	
Sesiones	*Probabilidad*	*Sesiones*	*Probabilidad*	*Sesiones*	*Probabilidad*
1	40%	1	50%	25	50%
2	40%	4	30%	30	50%
4	20%	10	20%		

Table 6: Configuración de la cantidad de sesiones por cada categoría de pacientes en las instancias utilizadas.

Urgente		Paliativo		Radical	
Alta Energía	*Baja Energía*	*Alta Energía*	*Baja Energía*	*Alta Energía*	*Baja Energía*
100%	0%	0%	100%	50%	50%

Table 7: Configuración del tipo de máquina que utiliza cada categoría de paciente para las instancias utilizadas.

7.3 Tiempo de Ejecución

Si bien el tiempo de ejecución de los algoritmos no es comparable entre sí, debido a que GRASP simplificado utiliza el acercamiento basado en Simulated Annealing internamente, se obtiene un promedio de los tiempos de ejecución para ambos algoritmos utilizando la instancia 60_30, dado que es la instancia que contiene más pacientes por asignar.

7.4 Desempeño entre Algoritmos

Los experimentos para analizar el desempeño de ambos algoritmos son de dos tipos. van destinados a analizar la calidad de la solución obtenida, donde se considera el tiempo de espera promedio por paciente, la cantidad de pacientes sin sus sesiones asignadas y el porcentaje de pacientes por categoría que no tienen su planificación sin sesiones. Los del segundo tipo son para ver la convergencia que tienen los algoritmos propuestos, donde se ve el comportamiento de ambos a medida que pasan las iteraciones.

7.4.1 Calidad de la solución

Para analizar la calidad de la solución se utilizan las instancias 60_30 y 30_30, debido a que incluyen una mayor cantidad de pacientes por asignar, por lo que es más difícil minimizar el tiempo de espera. Debido al comportamiento estocástico de los acercamientos propuestos cada uno se ejecuta 50 veces para la instancia de 30 días y 20 veces para la instancia de 60 días. En el caso de Simulated Annealing se utilizan 100 iteraciones y en el caso de GRASP simplificado se utilizan 50 iteraciones.

Este tipo de experimento se realiza para poder determinar si GRASP simplificado mejora los resultados obtenidos al aprovechar la aleatoriedad de la fase constructiva utilizada y que tanto los mejora, debido a que utilizar este acercamiento involucra un costo mayor en tiempo.

7.4.2 Convergencia

Para analizar la convergencia, se utilizan las instancias 60_30, 60_10, 30_30 y 30_10 puesto que el objetivo de este experimento es analizar el comportamiento que tienen los acercamientos dependiendo de la cantidad de días del horizonte de planificación y de la cantidad de pacientes. Se ejecuta cada algoritmo cuatro veces por cada instancia, debido al comportamiento estocástico, registrando el valor del tiempo de espera promedio en cada iteración.

7.5 Comparación ASAP/JIP versus GRASP Simplificado

Los últimos experimentos están destinados a comparar GRASP Simplificado con los algoritmos constructivos ASAP y JIP [12]. Para esto se utiliza la instancia 60_30, dado que esta contiene la mayor de cantidad de pacientes a asignar. El objetivo es determinar si el acercamiento desarrollado entrega mejoras con respecto a estas dos técnicas constructivas.

8 Resultados

A continuación se presentan los resultados para los experimentos detallados en la sección 8. Para su realización se utilizó un notebook con 8 Gb de memoria RAM y procesador Intel Core i7-4510U de cuatro núcleos de 2.00 GHz cada uno, utilizando Ubuntu 17.10 como sistema operativo.

8.1 Tiempo de Ejecución

El tiempo promedio de ejecución para los acercamientos GRASP Simplificado y Simulated Annealing se presentan en la tabla 8. Como era de esperar, GRASP Simplificado posee un mayor tiempo promedio de ejecución que Simulated Annealing debido a que utiliza Simulated Annealing durante 50 iteraciones.

GRASP Simplificado	***Simulated Annealing***
205.88 [s]	4.12 [s]

Table 8: Tiempo promedio de ejecución de GRASP Simplificado y Simulated Annealing utilizando la instancia 60_60.

8.2 Desempeño entre Algoritmos

8.2.1 Calidad de la solución

Un resumen de los resultados obtenidos al comparar el tiempo de espera promedio entre GRASP Simplificado y Simulated Annealing para la instancia 60_30 se presenta en la tabla 9. En promedio GRASP se comporta de mejor manera, además su peor resultado es mejor que el peor resultado de Simulated Annealing, mismo caso que sucede para los mejores resultados encontrados. Esto se puede observar de mejor manera en los Box plots de la figura 1, donde se puede observar que en un 75% de las veces que fue ejecutado GRASP Simplificado se obtuvo mejores resultados que el 75% de las ejecuciones de Simulated Annealing.

	GRASP Simplificado	**Simulated Annealing**
Max	3.187860	3.336260
Mean	2.894681	3.077072
Min	2.575140	2.752210

Table 9: Tabla resumen de tiempos de espera promedio por paciente para una muestra de $n = 20$, tomada a partir de una instancia de 60 días con 30 pacientes como máximo.

Para realizar un mejor análisis se realiza un test de Wilcoxon. Los resultados se presentan en la tabla 10. Se puede concluir que para la instancia 60_30, el desempeño de acuerdo al tiempo promedio de espera entre ambos acercamiento no es similar. Además, con un 99% de certeza, la calidad de la solución con respecto al tiempo de espera encontrada por GRASP Simplificado es mejor que la solución encontrada por Simulated Annealing.

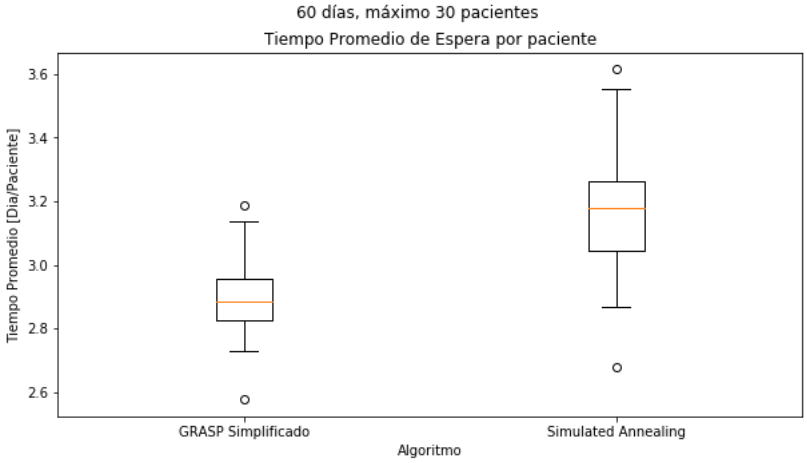

Figure 1: Box plot del tiempo de espera promedio por paciente para una instancia de 60 días y máximo 30 pacientes.

GRASP Simplificado - Simulated Annealing *Wilcoxon Test - Tiempo de Espera Promedio*	
N	20
W	17
p-value	0.00
Z	-0.20
Intervalo de confianza con 99% de certeza	
[-0.29, -0.052]	

Table 10: Resultados del test de Wilcoxon para el tiempo promedio de espera utilizando la instancia 60_30.

Un resumen de los resultados con respecto a los pacientes no asignados para la instancia 60_30 se presentan en la tabla 11. Se puede observar que el máximo, el mínimo y la media de los resultados obtenidos por GRASP Simplificado son mejores que los encontrados por Simulated Annealing. Los Box plots de la figura 2 muestran que el el 75% de los resultados obtenidos por GRASP Simplificado son mejores que casi el 100% de los resultados encontrados por Simulated Annealing, esto reflejado en el hecho que el tercer cuartil de los datos obtenidos por GRASP Simplificado coinciden con el mínimo valor encontrado por Simulated Annealing.

	GRASP Simplificado	**Simulated Annealing**
Max	399.000000	406.000000
Mean	395.950000	400.526316
Min	394.000000	397.000000

Table 11: Tabla resumen de pacientes no asignados para una muestra de $n = 20$, tomada a partir de una instancia de 60 días con 30 pacientes como máximo.

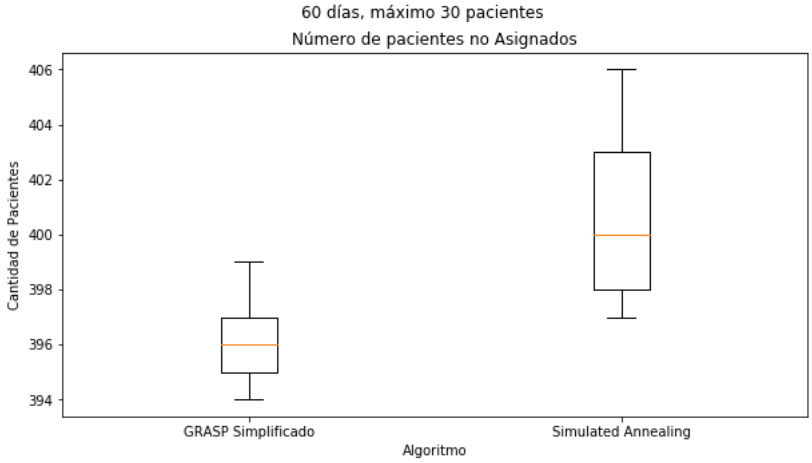

Figure 2: Box plot del número de pacientes no asignados para una instancia de 60 días y máximo 30 pacientes.

Para tener mayor respaldo estadístico se realiza un test de Wilcoxon para comparar el desempeño con respecto a los pacientes asignados para los dos métodos. Los resultados obtenidos se presentan en la tabla 12. Estos indican que el desempeño de los algoritmos no es similar, es más, con un 99% de certeza el algoritmo GRASP Simplicado obtiene mejores resultados que Simulated Annealing.

GRASP Simplificado - Simulated Annealing *Wilcoxon Test - Pacientes no asignados*	
N	20
W	210
p-value	0.00
Z	-4.50
Intervalo de confianza con 99% de certeza	
[-6.50, -3.00]	

Table 12: Resultados del test de Wilcoxon para los pacientes no asignados utilizando la instancia 60_30.

En el caso de la instancia 30_30, un resumen de los resultados obtenidos para el tiempo promedio por paciente se presenta en la tabla 13. Al considerar el valor máximo y el promedio, GRASP Simplificado obtiene mejores resultados, pero al considerar el mínimo encontrado se produce un empate entre las dos técnicas. En los box plots figura 3 no se puede apreciar mayor diferencia entre los dos algoritmos, salvo que Simulated Annealing tiene mayor variabilidad en los resultados obtenidos.

	GRASP Simplificado	**Simulated Annealing**
Max	1.186920	1.738320
Mean	0.975371	0.999548
Min	0.893204	0.893204

Table 13: Tabla resumen de tiempos de espera promedio por paciente para una muestra de $n = 50$, tomada a partir de una instancia de 30 días con 30 pacientes como máximo.

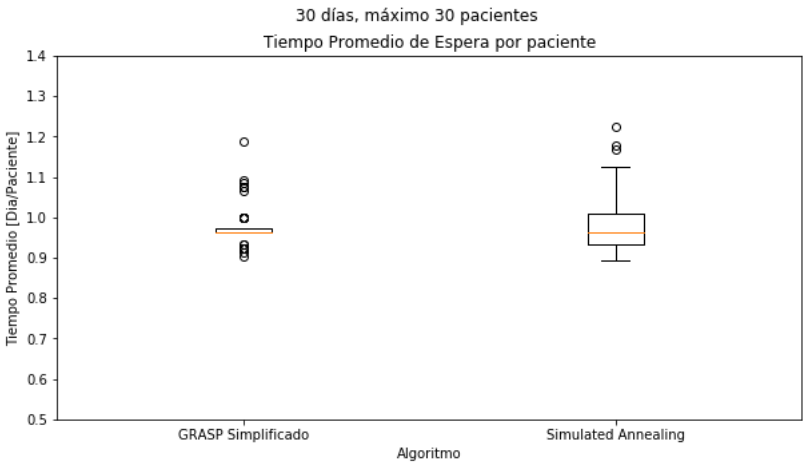

Figure 3: Box plot del tiempo de espera promedio por paciente para una instancia de 30 días y máximo 30 pacientes.

Se realiza un test de Wilcoxon para verificar estadísticamente si existe diferencia entre el desempeño de los dos algoritmos. Los resultados del test se presentan en la tabla 14. En base a los resultados del test no existe suficiente evidencia para determinar que existe diferencia en el comportamiento de los algoritmos.

GRASP Simplificado - Simulated Annealing *Wilcoxon Test - Tiempo de espera promedio*	
N	50
W	546
p-value	0.85
Z	-0.0044
Intervalo de confianza con 99% de certeza	
[-0.052, 0.029]	

Table 14: Resultados del test de Wilcoxon para el tiempo de espera promedio utilizando la instancia 30_30.

Con respecto a la cantidad de paciente asignados, la tabla 15 contiene un resumen de los datos. Nuevamente se repite la situación encontrada para el tiempo de espera promedio, en el cual considerando el máximo y el promedio GRASP Simplificado obtiene mejores resultados, pero existe un empate al considerar el valor mínimo. En la figura 4 se presentan box plot para los resultados obtenidos, donde se puede ver que un 75% de los datos obtenidos es mejor que el 75% de los resultados de Simulated Annealing, pero esto debido a la poca dispersión que presenta GRASP Simplificado.

	GRASP Simplificado	**Simulated Annealing**
Max	225.000000	226.000000
Mean	223.000000	223.900000
Min	222.000000	222.000000

Table 15: Tabla resumen de pacientes no asignados para una muestra de $n = 50$, tomada a partir de una instancia de 30 días con 30 pacientes como máximo.

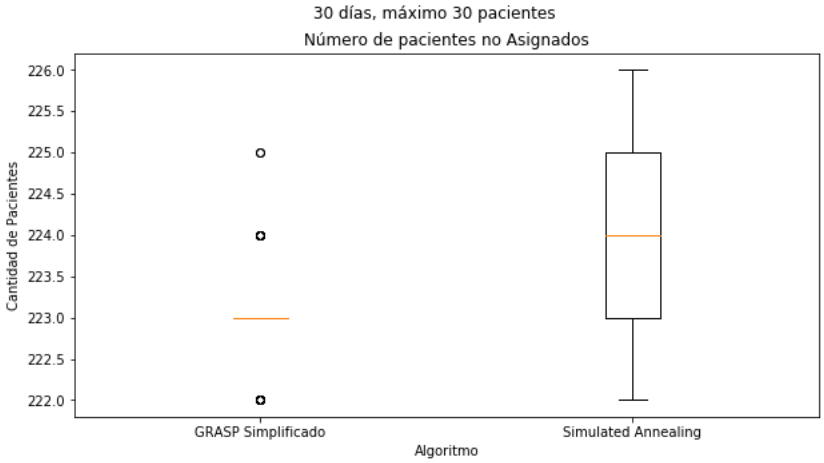

Figure 4: Box plot del número de pacientes no asignados para una instancia de 60 días y máximo 30 pacientes.

Se realiza un test de Wilcoxon para verificar si existen diferencia significativas en el comportamiento de los algoritmos con respecto a la cantidad de pacientes no asignados. Los resultados del test se presentan en la tabla 16. Se puede observar que el desempeño de los algoritmos es significativamente diferente, además GRASP-Simplificado es superior a Simulated Annealing con un 99% de certeza.

GRASP Simplificado - Simulated Annealing *Wilcoxon Test - Pacientes no asignados*	
N	50
W	70.5
p-value	$4.56 \cdot 10^{-5}$
Z	-1.50
Intervalo de confianza con 99% de certeza	
$[-2.00, -0.500]$	

Table 16: Resultados del test de Wilcoxon para los pacientes no asignados utilizando la instancia 30_30.

Si se considera el tiempo de espera promedio GRASP Simplificado funciona de mejor manera para la instancia de 60 días y tiene el mismo funcionamiento que Simulated Annealing para la instancia de 30 días. Si se considera el número de pacientes no asignados GRASP Simplificado obtiene mejores resultados para las dos instancias. De esto se desprende que es mejor utilizar GRASP Simplificado para ambos criterios de calidad.

8.3 Convergencia

8.3.1 Convergencia Simulated Annealing

Con respecto a las instancias 30_30 y 30_10, el comportamiento de Simulated Annealing conforme pasan las iteraciones se presenta en los gráficos de la figura 5. Se observa que para la instancia con 10 pacientes como máximo diario se estanca inmediatamente. Esto probablemente se deba al movimiento utilizado, donde considerando la cantidad de pacientes que intenta ingresar a la lista de asignados y la cantidad de pacientes totales de la instancia, al pasar la primera iteración

la planificación queda muy ajustada impidiendo nuevos cambios al continuar la ejecución. En cambio, se puede observar que este movimiento permite seguir mejorando la solución iteración a iteración al aumentar la cantidad máxima de pacientes diarios a asignar, debido a que existen más paciente en la lista de paciente no asignados, entregando así más posibilidades de mejora.

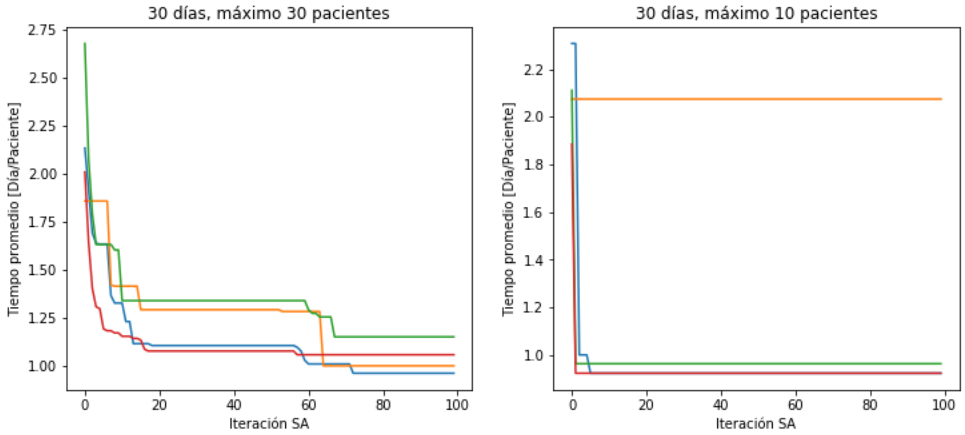

Figure 5: Tiempo de espera promedio versus iteración usando Simulated Annealing para instancias de 30 días como horizonte de planificación.

La misma situación se observa en los gráficos de la figura 6, donde se consideran las instancias 60_30 y 30_30. Se puede observar mejor comportamiento en la instancia para 30 pacientes como máximo diario que en la instancia con 10 pacientes, donde la búsqueda se estanca luego de una o dos mejoras a la solución.

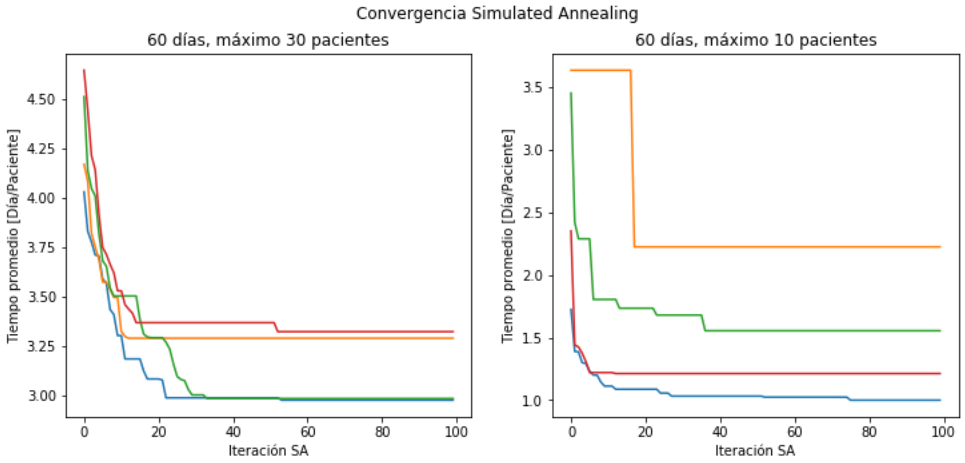

Figure 6: Tiempo de espera promedio versus iteración usando Simulated Annealing para instancias de 60 días como horizonte de planificación.

Esto indica que la cantidad máxima de paciente diarios es más influyente en el comportamiento que tendrá la búsqueda que la cantidad de días del horizonte de planificación. Donde el movimiento utilizado para mejorar la solución tiende a estancar la búsqueda para instancias con menos pacientes y a funcionar de mejor manera a medida que esta cantidad aumenta. Esto

indica la necesidad de desarrollar nuevos movimientos para optimizar la solución de instancias con menor cantidad de pacientes.

8.3.2 Convergencia GRASP Simplificado

Con respecto a la convergencia de GRASP Simplificado, se puede observar el comportamiento para la instancia 30_30 y 30_10 en los gráficos de la figura 7. En el caso con 10 pacientes como máximo diario se observa un rápido estancamiento, lo que puede venir el estancamiento que presenta Simulated Annealing para la misma instancia. Para la instancia de 30 pacientes como máximo diario se observa menos estacamiento, pero la cantidad de veces que mejora la solución objetivo es baja.

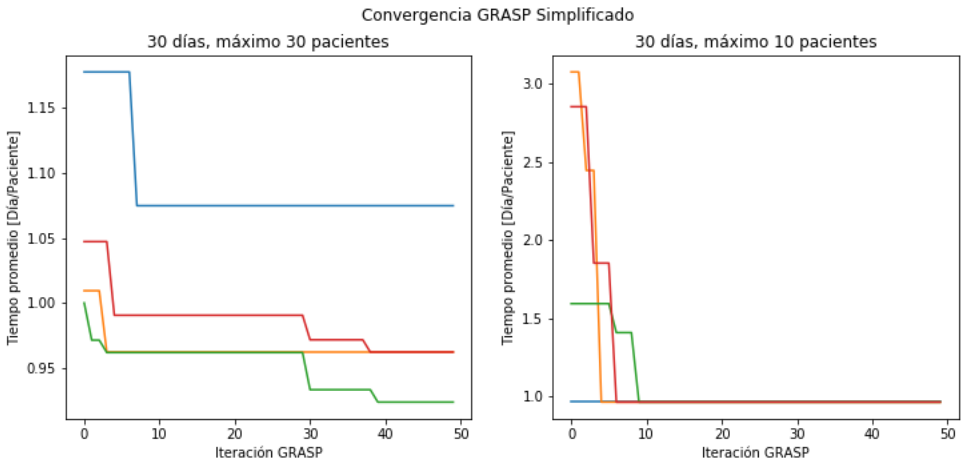

Figure 7: Tiempo de espera promedio versus iteración usando GRASP Simplificado para instancias de 30 días como horizonte de planificación.

El comportamiento de GRASP Simplificado las instancias 60_30 y 60_10 se presenta en los gráficos de la figura 8, donde nuevamente se puede observar un rápido estancamiento para la instancia de 10 pacientes como máximo diario y un mejor comportamiento para la instancia con 30 pacientes como máximo.

Esto indica, que si bien se obtiene una mejora al utilizar GRASP Simplificado la búsqueda se estanca muy rápido, de igual manera en que sucede con Simulated Annealing. Para revertir esta situación se puede modificar el movimiento utilizado por Simulated Annealing o modificar la fase constructiva de soluciones para agregar más diversidad. También se podría agregar una fase de pre-procesamiento adaptativo al acercamiento GRASP, de manera de encontrar maneras de mejorar la solución objetivo conforme avanza la búsqueda.

8.4 Comparación ASAP/JIP versus GRASP Simplificado

Como ASAP y JIP son técnicas constructivas deterministas se ejecutan solo una vez. En cambio, GRASP simplificado se ejecuta 10 veces utilizando solo dos iteraciones para mejorar la solución inicial. Los resultados se presentan en la tabla 17. Se puede observar que GRASP Simplificado obtiene un mejor desempeño al considerar los pacientes no asignados, logrando encontrar una planificación para 39 pacientes más que las técnicas constructivas.

Si se considera el tiempo de espera promedio por paciente ASAP obtiene mejores resultados que las otras dos técnicas. Comparando con GRASP simplificado se debe a que ASAP asigna 39 pacientes menos, los cuales de asignarse significarían un aumento en el tiempo de espera.

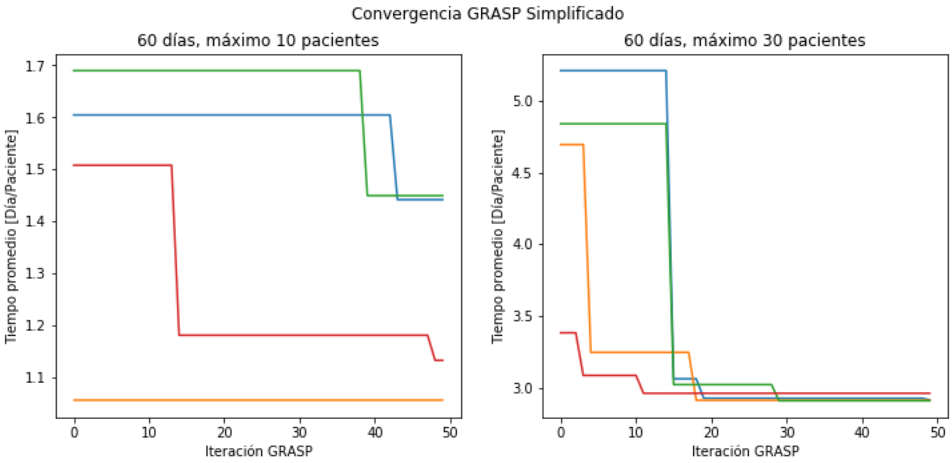

Figure 8: Tiempo de espera promedio versus iteración usando GRASP Simplificado para instancias de 60 días como horizonte de planificación.

	Tiempo de espera promedio	*Pacientes no Asignados*
ASAP	1.22	442
JIP	8.84	442
GRASP Simplificado	3.01	403
Dev-Std GRASP Simplificado	0.90	2.44

Table 17: Resultados obtenidos para ASAP, JIP y Grasp Simplificado utilizando la instancia 60_30.

Comparando JIP con las otras dos técnicas, esta obtiene tiempos de espera más altos debido a que intenta asignar pacientes desde el último día posible hacia atrás, provocando que los primeros pacientes ingresados tengan un alto tiempo de espera.

En la tabla 18 se presentan los resultados obtenidos para las tres algoritmos utilizando la instancia 30_30. Considerando el tiempo de espera promedio GRASP Simplificado obtiene mejores resultados que las otras dos técnicas, donde el tiempo de JIP sigue siendo muy superior a las otras dos. Ahora, tomando en cuenta la cantidad de pacientes asignados, la que obtiene mejores resultados es ASAP, superando a GRASP Simplificado por 4 pacientes.

	Tiempo de espera promedio	*Pacientes no Asignados*
ASAP	1	220
JIP	8.55	232
GRASP Simplificado	0.93	224
Dev-Std GRASP Simplificado	0.67	0.93

Table 18: Resultados obtenidos para ASAP, JIP y Grasp Simplificado utilizando la instancia 30_30.

Considerando ambas instancias GRASP Simplificado es la que obtiene mejores resultados. Si bien ASAP obtiene mejores resultados en una de las dos métricas de calidad (tiempo de espera promedio/pacientes no asignados), los resultados de la otra métrica hacen que sea mejor utilizar GRASP Simplificado. Las mejores presentes en GRASP Simplificado se deben a que aprovecha las dos heurísticas constructivas anteriores agregando un factor de aleatoriedad, lo que permite obtener mejores planificaciones en la fase de post-procesamiento.

9 Conclusiones

Considerando los dos acercamientos propuestos, GRASP Simplificado obtiene mejores resultados en las instancias utilizadas considerando tiempo de espera promedio y cantidad de pacientes no asignados, por lo que es preferible utilizarlo a pesar del tiempo adicional que involucra GRASP Simplificado al utilizar internamente Simulated Annealing para la fase de post-procesamiento.

Al momento de considerar la convergencia de los algoritmos, se hace visible la necesidad de cambiar el movimiento utilizado en Simulated Anneling, de manera que este con converja tan rápido para instancias con menos pacientes. Con respecto a la convergencia de GRASP simplificado, la cual también es rápida, una mejora que se puede realizar el agregar pre-procesamiento adaptativo a la construcción de soluciones, de manera de partir de la base de buenas soluciones iniciales desde las cuales poder comenzar a optimizar.

Al comparar GRASP Simplificado con ASAP y JIP se obtienen mejores resultados al utilizar GRASP Simplificado, por lo que este acercamiento presenta una mejora con respecto a estas dos soluciones, debido a que aprovecha las dos heurísticas constructivas para generar una planificación más fácil de optimizar en la fase de post-procesamiento.

Como trabajo futuro se planean buscar nuevos movimientos para evitar el estancamiento en la fase de post-procesamiento, además de agregar una fase de pre-procesamiento para ayudar al proceso de búsqueda. Además, es necesario realizar una sintonización de parámetros utilizando ParamILS [4], para tener mayor certeza de que los parámetros que se utilizan son los mejores posibles y de esta manera encontrar mejores soluciones.

Por último, es necesario adaptar este acercamiento para que resuelva instancias de todos los países y no solo del caso chileno, para comparar con más técnicas presentes en la literatura.

References

[1] Juan Pablo Cares, María-Cristina Riff, and Bertrand Neveu. Genera: A problem generator for radiotherapy treatment scheduling problems. *Annals of Mathematics and Artificial Intelligence*, 76(1):191–214, Feb 2016.

[2] D. Conforti, F. Guerriero, and R. Guido. Non-block scheduling with priority for radiotherapy treatments. *European Journal of Operational Research*, 201(1):289 – 296, 2010.

[3] Rosita Guido and Domenico Conforti. A hybrid genetic approach for solving an integrated multi-objective operating room planning and scheduling problem. *Computers & Operations Research*, 87:270 – 282, 2017.

[4] Frank Hutter, Holger H Hoos, Kevin Leyton-Brown, and Thomas Stützle. Paramils: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36:267–306, 2009.

[5] Y. Jacquemin, E. Marcon, and P. Pommier. Towards an improved resolution of radiotherapy, scheduling. In *2010 IEEE Workshop on Health Care Management (WHCM)*, pages 1–6, Feb 2010.

[6] Chun-Cheng Lin, Jia-Rong Kang, Ding-Jung Chiang, and Chien-Liang Chen. Nurse scheduling with joint normalized shift and day-off preference satisfaction using a genetic algorithm with immigrant scheme. *International Journal of Distributed Sensor Networks*, 11(7):595419, 2015.

[7] S. K. Mishra, P. S. C. Bose, and C. S. P. Rao. An invasive weed optimization approach for job shop scheduling problems. *The International Journal of Advanced Manufacturing Technology*, 91(9):4233–4241, Aug 2017.

[8] Norhayati Mohd Rasip, Abd Samad Basari, Burairah Hussin, and Nuzulha Khilwani. A guided particle swarm optimization algorithm for nurse scheduling problem. 8:5625 – 5632, 07 2014.

[9] Maroua Nouiri, Abdelghani Bekrar, Abderezak Jemai, Smail Niar, and Ahmed Chiheb Ammari. An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem. *Journal of Intelligent Manufacturing*, 29(3):603–615, Mar 2018.

[10] Dobrila Petrovic, Mohammad Morshed, and Sanja Petrovic. Multi-objective genetic algorithms for scheduling of radiotherapy treatments for categorised cancer patients. *Expert Systems with Applications*, 38(6):6994 – 7002, 2011.

[11] S. Petrovic and P. Leite-Rocha. Constructive and grasp approaches to radiotherapy treatment scheduling. In *Advances in Electrical and Electronics Engineering - IAENG Special Edition of the World Congress on Engineering and Computer Science 2008*, pages 192–200, Oct 2008.

[12] Sanja Petrovic, William Leung, Xueyan Song, and Santhanam Sundar. Algorithms for radiotherapy treatment booking, 2006.

[13] María-Cristina Riff, Juan Pablo Cares, and Bertrand Neveu. Rason: A new approach to the scheduling radiotherapy problem that considers the current waiting times. *Expert Systems with Applications*, 64:287 – 295, 2016.

[14] Jacquemin Yoan, Marcon Eric, and Pommier Pascal. A pattern-based approach of radiotherapy scheduling. *IFAC Proceedings Volumes*, 44(1):6945 – 6950, 2011. 18th IFAC World Congress.

[15] A. Youssef and S. Senbel. A bi-level heuristic solution for the nurse scheduling problem based on shift-swapping. In *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 72–78, Jan 2018.