Inteligencia Artificial

Informe Final: Nurse Scheduling Problem

José Miguel Castro

18 de agosto de 2018

Evaluación

Mejoras 1ra Entrega (10 %):	_____
Código Fuente (10 %):	_____
Representación (15 %):	_____
Descripción del algoritmo (20 %):	_____
Experimentos (10 %):	_____
Resultados (10 %):	_____
Conclusiones (20 %):	_____
Bibliografía (5 %):	_____
Nota Final (100):	_____

Resumen

El problema de planificación de enfermeros (**NSP**) consiste en obtener una asignación de turnos de trabajo para un determinado periodo dentro de un hospital. Las restricciones de este problema estarán definidas por condiciones dadas tanto por el hospital como por los empleados y la calidad de las soluciones obtenidas estará definida por la cantidad de restricciones cumplidas, donde se busca minimizar las penalizaciones asociadas a cada restricción blanda.

En el presente documento se explicará el problema estudiado y se mostrarán los distintos métodos de modelamiento y técnicas de resolución utilizados actualmente, haciendo énfasis en lo más usado y efectivo.

1. Introducción

El problema de asignación de turnos, conocido en el área como *Nurse Scheduling Problem* (**NSP**) ó *Nurse Rostering Problem* (**NRP**) busca encontrar una asignación de enfermeros válida para cubrir los turnos requeridos por un hospital, intentando respetar la mayor cantidad de restricciones posibles, tales como preferencias de horarios por parte de los enfermeros ó cantidad mínimas de horas requeridas por el hospital.

Clásicamente, esta tarea es realizada de forma manual, lo que consume mucho tiempo y muchas veces es casi imposible encontrar una solución que sea satisfactoria debido a la gran

cantidad de aristas que se deben considerar. La utilización de un método automático para encontrar un horario adecuado para cada conjunto de restricciones beneficiará claramente a quien esté encargado de planificar los turnos de el personal, aumentando la eficiencia y eficacia del cargo.

En la Sección 2 se definirá en detalle el problema estudiado, presentando todas las características que componen al problema, junto con las variables comúnmente utilizadas para describirlo.

Luego de haber definido el problema, en la Sección 3 se mostrará como es resuelto actualmente el problema, cuales son las mejores técnicas a utilizar, cual de estas tienen mejores resultados, etc. Tomando algunas de las técnicas presentadas, en la Sección 4 se presentarán las diferentes modelaciones matemáticas del problema a resolver, explicando detalladamente cada una de sus partes. En estas secciones se podrá estudiar y entender cómo el problema es resuelto y así poder desarrollar una solución propia a partir de estos referentes.

Basándose en esta definición matemática del problema, se definirá una representación en la Sección 5 para las instancias, lo que será utilizado en la implementación de un algoritmo basado en Hill Climbing para encontrar soluciones, el cual se describirá en la Sección 6.

Finalmente los experimentos realizados para llegar a una versión final del algoritmo serán descritos en la Sección 7, seguido de los resultados obtenidos en la Sección 8.

2. Definición del Problema

La asignación de turnos para personal de un hospital es un problema muy estudiado en los campos de investigación de operaciones e inteligencia artificial, debido a que es un problema con el que se debe lidiar en el día a día en muchos campos. Este problema se puede catalogar principalmente bajo tres categorías en investigación de operaciones, planificación, asignación de recursos y calendarización administrativa.

El NSP busca asignar turnos para cubrir un determinado periodo de tiempo en una unidad médica respetando las necesidades existentes. La dificultad que tiene la resolución de este problema se genera a partir del gran número de restricciones que se aplican, las cuales representan las políticas que tiene el hospital, las del personal, restricciones en tiempo de trabajo y otras. Es debido a estas restricciones que el problema pertenece a la categoría de NP-Complejo. Además se debe definir la cantidad de turnos por día, los cuales generalmente son clasificados en día, tarde y noche. Esta categorización es la más utilizada en los trabajos a estudiar.

Las restricciones anteriormente mencionadas pueden ser clasificadas en dos categorías:

- **Restricciones Duras**: son las restricciones que deben cumplirse obligatoriamente para que una solución sea factible.
- **Restricciones Blandas**: estas restricciones no son obligatorias pero el cumplimiento de estas aumentará la satisfacción que tendrá la solución encontrada. Muchas veces es imposible encontrar una solución que cumpla todas las restricciones, pero por eso que algunas pertenecientes a esta categoría deban ser sacrificadas para hacer posible la resolución del problema.

Para realizar la planificación, primero hay que ver las restricciones como la cantidad mínima de horas a trabajar por turno y por semana, cantidad mínima de empleados por turno,

los turnos de descanso obligatorio entre turnos trabajados, fin de semanas trabajados al mes, días libres, etc. Estás restricciones corresponderían a restricciones duras, ya que están definidas por políticas de trabajo. Ejemplos de restricciones blandas podrían ser las preferencias de un empleado por algún turno (día, tarde, noche) o por algún día de la semana, también se podría considerar tener varios turnos trabajados seguido de varios turnos libres [15].

Dado que sólo el encargado de la unidad conoce las preferencias, personalidades, facultades y salud de cada uno de los empleados, sólo este puede ver qué restricciones pueden infringidas, por lo que la solución entregada como resultado debe tener cierto grado de flexibilidad, como para poder ser adaptado en una segunda etapa [7].

Una solución válida del problema será una que cumpla todas las restricciones duras, y el puntaje o castigo de caga solución válida estará dada por las restricciones blandas y esta dependerá del objetivo que tiene, como por ejemplo maximizar la comodidad de cada empleado con los turnos asignados, disminución de costos, entre otros.

3. Estado del Arte

El NSP actualmente se encuentra aún en el proceso de desarrollo ya que todavía no se encuentra un método universal para su resolución. Los primeros estudios sobre este problema se pueden encontrar en la década del 60 [16, 17], donde el objetivo era principalmente distribuir enfermeras en distintas unidades. La solución era encontrada utilizando métodos matemáticos que, debido a el poco poder computacional existente, eran casi imposibles de resolver por lo que luego, con el tiempo se aplicaron heurísticas y meta-heurísticas para facilitar su desarrollo.

Desde sus inicios se han utilizado variados técnicas para su resolución, entre las que se incluyen las basadas en programación entera [3], algoritmos genéticos [2], modelamiento difuso [13] y algoritmos utilizando ant colony optimization [6].

Actualmente, los métodos más eficientes han sido presentados en la *International Nurse Rostering Competition*, la cual ha tenido dos versiones, la primera en el año 2014 [8] y la segunda en el año 2016 [9].

El ganador de la primera versión de esta competencia [14], dividieron cada instancia en subproblemas con un tamaño admitible y fueron resueltos secuencialmente usando programación entera. El proceso se dividió en dos fases, en la primera se asignó la distribución de trabajo por cada empleado y por cada día, mientras que en la segunda fase se asignaron los turnos específicos en el día a día. Además se utilizaron técnicas de optimización local para buscar entre combinaciones de asignaciones parciales para los empleados.

El segundo lugar, obtenido por Burke et al [5], quienes han tenido una gran influencia en el área, utilizaron dos algoritmos, el primero fue utilizado para instancias de menor tamaño, mientras el segundo fue usado para instancias de medio y gran tamaño. Para pequeñas instancias se usó una búsqueda en profundidad variable, basado en métodos de *ejection chain*. Para las instancias de mayor tamaño se aplicó un algoritmo de *branch and price* donde el problema de *pricing* fue modelado como un problema de ruta mínima con recursos limitados y resuelto utilizando programación dinámica.

Otro trabajo propuesto en la competencia [4], utiliza una hiper-heurística dividida en dos fases, donde en la primera parte la hiper-heurística es utilizada durante el 80 % incial del tiempo computacional, en donde se selecciona una heurística de forma aleatoria de una lista y luego se

utiliza *simulated annealing* para el criterio de aceptación. En la segunda se usa una heurística greedy para seleccionar algúna solución obtenida por la hiper-heurística.

4. Modelo Matemático

El siguiente modelo [10] fue definido en base a la descripción dada en las reglas de la competencia INRC-II antes mencionada [12], sobre las cuales se desarrollaron la mayoría de los algoritmos mencionados anteriormente.

En el contexto del problema enunciado en la competencia, se define lo siguiente

- Escenario
  - Horizonte de planificación: número de semanas a planificar.
  - Empleados (N): lista de empleados.
  - Habilidades (K): lista de habilidades que podrán tener los empleados.
  - Habilidades por empleado: lista de habilidades que podrá tener cada empleado.
  - Contratos: límites bajo los que trabaja cada empleado, mínimos y máximos de turnos, días libres, turnos consecutivos, fin de semanas trabajados y la opción de si trabajar ambos días del fin de semana o ninguno de los dos (W).
  - Tipo de turnos (S): para cada tipo de turno se define mínimo y máximo para asignaciones consecutivas de un tipo de turno, además de combinaciones no permitidas de estos (P), por ejemplo trabajar una noche seguida de una mañana.
- Datos semanales
  - Requerimientos: para cada turno, habilidad y día de la semana, el numero óptimo y mínimo de empleados para cumplir lo requerido.
  - Preferencias: 3-tupla dada por enfermera, día, turno y define la preferencia de dicha enfermera por no trabajar en ese día y turno.
- Historial
  - Datos de borde: turnos trabajados en el último día de la semana anterior, número de turnos consecutivos del mismo tipo en general y días libres consecutivos.
  - Contadores: número total de turnos trabajados, número de fin de semanas trabajados.

Las variables a utilizar se definen en la Tabla 4.
El conjunto de restricciones [12] a tomar será el siguiente

- **Restricciones Duras**
  - H1. **Asignación única por turno**: Un trabajador puede ser asignado como máximo una vez al turno.
  - H2. **Deficiencia de personal**: El número de empleados por turno y por habilidad deben ser al menos el mínimo requerido.
  - H3. **Sucesiones de turno**: Las asignaciones de turno en dos días consecutivos debe ser permitido.
  - H4. **Inhabilidad**: Un turno de cierta habilidad debe estar cubierto por un empleado que tenga esa habilidad.
- **Restricciones Blandas**

Símbolo	Dominio	Definición
n	N	Conjunto de empleados
d	D	Conjunto de días
s	S	Conjunto de turnos
h	$\mathbb{N}$	Número de turnos (h), donde h+1 representa un día libre
k	K	Conjunto de habilidades
p	P	Conjunto de combinaciones inválidas de turno
MR_{dsk}	$\mathbb{N}$	Requisito mínimo de enfermeras en el día d, turno s y habilidad k
OR_{dsk}	$\mathbb{N}$	Requisito óptimo de enfermeras en el día d, turno s y habilidad k
$MINCAS_s$	$\mathbb{N}$	Mínimo de asignaciones consecutivas por turno para s
$MAXCAS_s$	$\mathbb{N}$	Máximo de asignaciones consecutivas por turno para s
$NMCAS_{nsd}$	$\mathbb{N}$	Número de días faltantes
$NECAS_{nsd}$	$\{0,1\}$	Número de días excedidos
$MINCAG_n$	$\mathbb{N}$	Mínimo de asignaciones consecutivas globales para n
$MAXCAG_n$	$\mathbb{N}$	Máximo de asignaciones consecutivas globales para n
$NMCAG_{nd}$	$\mathbb{N}$	Número de días faltantes
$NECAG_{nd}$	$\{0,1\}$	Número de días excedidos
$DSns$	$\mathbb{N}$	Nivel de preferencia del empleado n hacia el turno s
W_n	$\{0,1\}$	Si el empleado n debe trabajar el fin de semana completo
MDW_{nd}	$\mathbb{N}$	Número de días faltantes en un fin de semana para n
$MINWD_n$	$\mathbb{N}$	Mínimo de asignaciones globales para n
$MAXWD_n$	$\mathbb{N}$	Máximo de asignaciones globales para n
$NMWD_n$	$\mathbb{N}$	Número de asignaciones globales faltantes para n
$NEWD_n$	$\mathbb{N}$	Número de asignaciones globales excedidas para n
$MAXWW_n$	$\mathbb{N}$	Máximo de fin de semanas trabajados para n
$NEWW_n$	$\mathbb{N}$	Número de fin de semanas trabajados en exceso para n
WW_{nd}	$\{0,1\}$	Si el empleado n trabaja al menos un día del fin de semana
BD_{nsk}	$\mathbb{N}$	Número de días continuos trabajados de s en s y k
IBD_{nsk}	$\{0,1\}$	Si BD_{nsk} es distinto de cero o no
x_{nsdk}	$\{0,1\}$	Si el empleado n trabaja el turno s, día d realizando k
r_{nk}	$\{0,1\}$	Si el empleado n puede realizar k

Cuadro 1: Variables de Modelado

S1. **Personal insuficiente para la cobertura mínima (30)**: El número de empleados en cada turno y para cada habilidad debe se igual al requerimiento óptimo. Cada empleado faltante es penalizado según el peso definido. Empleados por sobre el óptimo no son considerados en el costo.

S2. **Asignaciones consecutivas (15/30)**: El número máximo y mínimo de asignaciones por turno o globalmente debe ser respetado. Cada día extra o faltante es multiplicado por el peso correspondiente. El peso para restricciones sobre turnos consecutivos es 15 y para días trabajados consecutivos es 30.

S3. **Días libres consecutivos (30)**: El número mínimo y máximo de días libres consecutivos debe ser respetado. Cada día extra o faltante es multiplicado por el peso correspondiente.

S4. **Preferencias (10)**: Cada asignación a un turno no deseado es penalizado por el peso correspondiente.

S5. **Fin de semanas completos (30)**: Cada trabajador que tenga complete weekend deben trabajar sábado y domingo ó ninguno de estos. Si trabaja sólo uno de estos días, es penalizado por el peso correspondiente.

S6. **Asignaciones totales (20)**: Para cada empleado, el número total de días trabajados debe estar entre el mínimo y el máximo estipulado en su contrato. La diferencia, multiplicada por el peso será sumada a la función objetivo.

S7. **Total de fin de semanas trabajados (30)**: Para cada empleado el número de fin de semanas trabajados debe ser menor o igual al máximo. El exceso, multiplicado por su peso será agregadoa la función objetivo. Un fin de semana es considerado como trabajado si trabajo al menos uno de los dos días.

La función objetivo del modelo está dada por la suma de las penalizaciones de cada restricción blanda

$$min\ Z : \Delta Z_1 + \Delta Z_2 + \Delta Z_3 + \Delta Z_4 + \Delta Z_5 + \Delta Z_6 + \Delta Z_7 \tag{1}$$

la cual estará sujeta a las restricciones duras

$$\sum_s \sum_k x_{nsdk} = 1,\ \forall n \in N, d \in D \tag{2}$$

$$\sum_n x_{nsdk} \cdot r_{nk} \geq MR_{sdk},\ \forall s \in S, d \in D, k \in K \tag{3}$$

$$\sum_k (x_{n,s_1,d-1,k} - x_{n,s_2,d,k}) \leq 1,\ \forall n \in N, d \in D \backslash \{1\}, (s_1, s_2) \in P \tag{4}$$

donde la ecuación (2) hace referencia a la restricción **H1**, (3) a **H2** y (4) a **H3**.

Además, la restricción blanda **S1** está representadas por (5) y (6):

$$\sum_n x_{nsdk} \cdot r_{nk} + MR_{sdk} \geq RO_{sdk},\ \forall s \in S, d \in D, k \in K \tag{5}$$

$$\Delta Z_1 = C_1 \cdot \sum_s \sum_d \sum_k M_{sdk} \tag{6}$$

S2 está representadas por (7), (8), (9), (10) y (11):

$$\sum_{d=d_0}^{d_f} \sum_k x_{nsdk} + NMCAS_{nsd_0} \geq MINCAS_s \cdot \sum_k (x_{nsd_0k} - x_{n,s,d_0-1,k}), \tag{7}$$
$$\forall n \in N, s \in S, d_0 \in D \backslash \{1\}, d_0 \leq |D| - MINCAS_s + 1, d_f = d_0 + MINCAS_s - 1$$

$$\sum_k BD_n sk + \sum_k \sum_{d=1}^{d_f} x_{nsdk} + NMCAS_{ns1} \geq MINCAS_s \cdot \sum_k (x_{ns1k} - IBD_n sk), \tag{8}$$
$$\forall n \in N, s \in S, d_f = MINCAS_s - BD_{nsk}$$

$$\sum_{d=d_0}^{d_f} \sum_k x_{nsdk} - NECAS_{nsd_0} \leq MAXCAS_s, \tag{9}$$
$$\forall n \in N, s \in S, d_0 \in D \backslash \{1\}, D_f = min\{d_0 + MAXCAS_s, |D|\}$$

$$\sum_k BD_n sk + \sum_k \sum_{d=1}^{d_f} x_{nsdk} - NECAS_{ns1} \leq MAXCAS_s, \tag{10}$$
$$\forall n \in N, s \in S, d_f = MAXCAS_s - BD_{nsk} + 1$$

$$\Delta Z_2 = C_2 \cdot \sum_n \sum_s \sum_d (NMCAS_{nsd} - NECAS_{nsd}) \tag{11}$$

S3 está representadas por (12), (13), (14), (15) y (16):

$$\sum_{d=d_0}^{d_f} \sum_{s=1}^{h} \sum_k x_{nsdk} + NMCAG_{nd_0} \geq MINCAG_n \cdot \sum_s \sum_k (x_{nsd_0k} - x_{n,s,d_0-1,k}), \tag{12}$$
$$\forall n \in N, d_0 \in D \backslash \{1\}, d_0 \leq |D| - MINCAG_n + 1, d_f = d_0 + MINCAG_n - 1$$

$$\sum_{s=1}^{h} \sum_k BD_n sk + \sum_{s=1}^{h} \sum_k \sum_{d=1}^{d_f} x_{nsdk} + NMCAG_{n1} \geq MINCAG_n \cdot \sum_{s=1}^{h} \sum_k x_{ns1k}, \tag{13}$$
$$\forall n \in N, d_f = MINCAG_n - BD_{nsk}$$

$$\sum_{d=d_0}^{d_f} \sum_{s=1}^{h} \sum_k x_{nsdk} - NECAG_{nd_0} \leq MAXCAG_n, \tag{14}$$
$$\forall n \in N, d_0 \in D \backslash \{1\}, D_f = min\{d_0 + MAXCAG_n, |D|\}$$

$$\sum_{s=1}^{h} \sum_k BD_n sk + \sum_{s=1}^{h} \sum_k \sum_{d=1}^{d_f} x_{nsdk} - NECAG_{n1} \leq MAXCAG_n, \tag{15}$$
$$\forall n \in N, d_f = MAXCAG_n - BD_{nsk} + 1$$

$$\Delta Z_3 = C_3 \cdot \sum_n \sum_d (NMCAG_{nd} - NECAG_{nd}) \tag{16}$$

S4 está representadas sólo por la penalización (17)

$$\Delta Z_4 = C_4 \cdot \sum_n \sum_s \sum_d \sum_k DS_{ns} \cdot x_{nsdk} \tag{17}$$

S5 está representadas por (18), (19) y (20)

$$\sum_k \sum_{s=1}^{h} x_{nsdk} - NDW_{nd} \leq \sum_k \sum_{s=1}^{h} x_{n,s,d-1,k} + (1 - W_n), \tag{18}$$
$$\forall n \in N, d \in \{7, 14, 21, 28\}$$

$$\sum_k \sum_{s=1}^{h} x_{nsdk} + NDW_{nd} \geq \sum_k \sum_{s=1}^{h} x_{n,s,d-1,k} + (1 - W_n), \tag{19}$$
$$\forall n \in N, d \in \{7, 14, 21, 28\}$$

$$\Delta Z_5 = C_5 \cdot \sum_n \sum_{d \in D\prime} MDD_{nd}, \tag{20}$$
$$donde\ D\prime = \{7, 14, 21, 28\}$$

S6 está representadas por (21), (22) y (23)

$$\sum_{d}\sum_{s=1}^{h}\sum_{k} x_{nsdk} + NMWD_n \geq MINWD_n, \quad \forall n \in N \tag{21}$$

$$\sum_{d}\sum_{s=1}^{h}\sum_{k} x_{nsdk} - NEWD_n \leq MAXWD_n, \quad \forall n \in N \tag{22}$$

$$\Delta Z_6 = C_6 \cdot \sum_{n}(NMWD_n + NEWD_n) \tag{23}$$

S7 está representadas por (24), (25), (26) y (27)

$$\sum_{s=1}^{h}\sum_{k}(x_{nsdk} + x_{n,s,d-1,k}) \leq 2 \cdot WW_{nd}, \quad \forall n \in N, d \in \{7, 14, 21, 28\} \tag{24}$$

$$\sum_{s=1}^{h}\sum_{k}(x_{nsdk} + x_{n,s,d-1,k}) \geq WW_{nd}, \quad \forall n \in N, d \in \{7, 14, 21, 28\} \tag{25}$$

$$\sum_{d \in D\prime} WW_{nd} - NEWW_n \leq MAXWW_n, \quad donde\ D\prime = \{7, 14, 21, 28\} \tag{26}$$

$$\Delta Z_7 = C_7 \cdot \sum_{n} NEWW_n \tag{27}$$

El tamaño de búsqueda será determinado por el número de empleados, días, turnos y habilidades, como las variables de decisión es binaria, el tamaño calcula como

$$2^{n \cdot d \cdot s \cdot k} \tag{28}$$

5. Representación

La representación que se utilizará será un matriz, en donde las filas representarán a cada empleado y las columnas los turnos.

Esta representación nos facilitará el cálculo de penalizaciones y la verificación de la factibilidad de cada solución ya que al revisar de forma horizontal se pueden verificar la cantidad de turnos en total de un empleado, los turnos consecutivos y loas días consecutivos. Además de forma vertical se podrá verificar que cada turno tenga la cobertura mínima, para que así la

solución sea factible.

Un ejemplo de instancia es el mostrado en la siguiente matriz, donde hay 3 empleados y 8 turnos.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Como se puede apreciar, en esta instancia el primer empleado estaría trabajando por 4 turnos seguidos y además en el quinto turno, no hay ningún empleado de turno.

6. Descripción del algoritmo

Luego de haber definido la representación, se eligió un movimiento de tipo *bit flip* donde a un empleado se le agregará o quitará un turno, verificando que la única restricción dura se cumpla, moviéndose sólo por soluciones factibles.

Al iniciar el programa, luego de inicializar los parámetros del problema en específico, se ejecutará una función random que asignará aleatoriamente empleados a un turno de forma de cumplir el mínimo de cobertura.

```
Function realizarNSP(reintentos, iteraciones)
    instancias = []
    for i en reintentos do
        instancia = generarInstanciaRandom()
        score = calcularPenalizaciones(instancia)
        for j en iteraciones do
            mejora = false
            for movimiento en movimientosPosibles do
                nuevaInstancia = movimiento(instancia)
                if validar(nuevaInstancia) then
                    nuevoScore = calcularPenalizaciones(nuevaInstancia)
                    if nuevoScore < score then
                        instancia = nuevaInstancia
                        score = nuevoScore
                        mejora = true
                        break
                    end
                end
            end
            if mejora then
                break
            end
        end
        instancias ← instancia
    end
    return seleccionarMejor(instancias)
```

Algorithm 1: Solución al NSP utilizando *Hill Climbing*

Como ya se mencionó, al inicar se inicializa *scheduler* con los archivos indicados, los cuales contienen las variables como número de empleados, número de turnos y número de días, además de la matriz de cobertura y la de preferencia.

Luego se comenzará a iterar por los posibles movimientos, los cuales se han desordenado aleatoriamente para que los movimientos no sean siempre sobre las primeras enfermeras o turnos. Luego se verifica que el vecino generado sea factible. De no serlo se pasará al siguiente movimiento hasta encontrar uno que sea factible, el que luego es evaluado y comparado con la instancia actual, en donde si hay mejora, se guarda la instancia y se pasa a la siguiente iteración. en caso contrario, se prosigue con el siguiente movimiento.

Al finalizar todas las iteraciones, ya sea porque se agotaron o se llegó a un máximo local, la mejor instancia es guardada y realiza un *reset*, volviendo a iterar partiendo de una nueva instancia random. Luego de que los reintentos se terminen, se selecciona la mejor instancia de las obtenidas y se retorna, siendo esta la mejor asignación encontrada para el problema.

7. Experimentos

Luego de haber desarrollado el código y verificar que funcionaba correctamente, se inició un estudio del comportamiento que presentaba el algoritmo para resolver las distintas instancias existentes en el dataset [11]. Este proceso fue facilitado a través del uso de un script en python que ayudaba a elegir instancias de forma aleatoria y ejecutar el programa sobre estas.

Inicialmente se quiso estudiar la importancia del número máximo de iteraciones y la cantidad de reintentos. Para esto se ejecutó el algoritmo en variadas instancias del dataset, guardando el número de iteraciones realizadas hasta encontrar el máximo local y el puntaje de este.

Seguido a esto, se modificó la función que generaba las soluciones iniciales agregando aún más aleatoriedad y se analizó los puntajes obtenidos y las iteraciones realizadas.

Luego se siguió analizando el puntaje obtenido a través de una ejecución sobre una sola instancia con 5000 reintentos, con el fin de conocer la importancia de aumentar la exploración del algoritmo.

Finalmente se experimento con variados valores para cada penalización, con el fin de estudiar el comportamiento de las asignaciones finales en cada solución.

8. Resultados

Una pequeña muestra de los resultados obtenidos para la primera experimentación fueron graficados en la Figura 1, en donde se aplicó el algoritmo a 5 instancias de 25 empleados, seleccionadas de forma aleatoria y realizando 100 reintentos para cada una.

Una de las primeras características que se notaron sobre los resultados, es que el número de iteraciones realizadas es muy similar entre reintentos, por lo que al iniciar el algoritmo con un máximo muy grande no afectará el proceso de este, ya que la gran mayoría de intentos tendrán numero de iteraciones muy parecidas. Esto se puede ver en la Figura 1, en donde se nota que la distribución de iteraciones es muy compacta, teniendo diferencias de menos de 30 entre si.

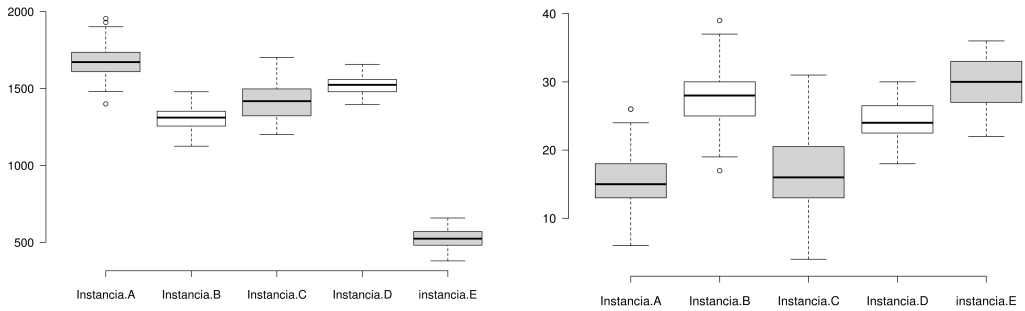

Figura 1: Distribución de puntaje obtenido (a la izquierda) y número de iteraciones realizadas (a la derecha) en distintas instancias del dataset utilizando 100 reintentos y un máximo de 10000 iteraciones

Esto último refleja que existe una gran cantidad de máximos locales, por lo que es muy fácil que se quede "atrapado" y no se siga iterando.

Por otro lado, también se puede notar que la distribución de puntajes obtenidos es mucho más dispersa, haciendo que el resultado de un intento pueda tener la mitad de la calidad de otro, lo que es causado por la aleatoriedad de la solución inicial y de los movimientos realizados.

Para confirmar esta aleatoriedad, se volvió a realizar otro experimento en donde una sola instancia fue resuelta utilizando 5000 reintentos, donde los puntajes fueron agrupados cada 50 puntos. La cuenta de cada grupo es graficada en la Figura 2, en donde claramente se puede ver la baja probabilidad de obtener el mejor puntaje.

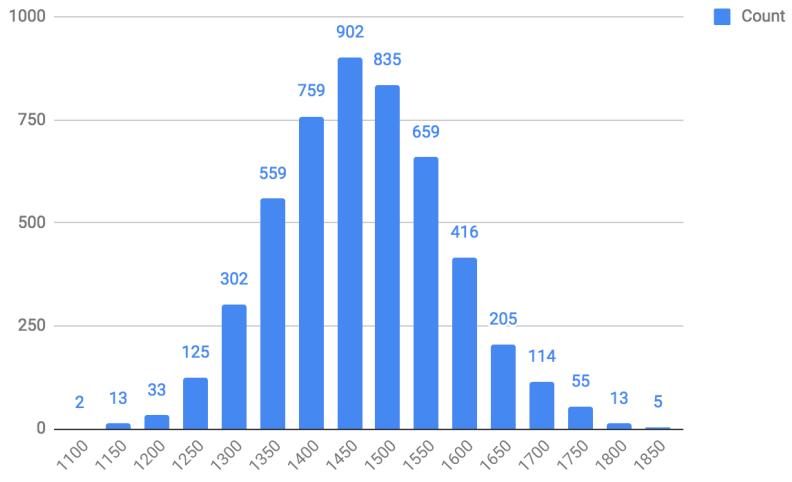

Figura 2: Distribución de puntajes obtenido realizando 5000 reintentos en una instancia aleatoria

De estos resultados se puede confirmar que la aleatoriedad es muy importante en esta técnica, ya que al realizar poco reintentos se tendrá muy baja probabilidad de obtener los mínimos encontrados, los que corresponden a penas al 0,04 % de los reintentos realizados en este experimento.

Otro factor importante en la búsqueda de las soluciones es la solución inicial generada. En

primera instancia se creó una función que asignaba x enfermeras de forma aleatoria en cada turno, donde x es el mínimo de cobertura para cada turno siendo asignado. Luego, se creó una variación de esta función, en donde x fue un número aleatorio mayor al mínimo requerido y menor al número de enfermeras en la instancia. La comparación entre iteraciones necesarias para llegar al óptimo local y puntaje obtenido fue graficado en la Figura 3

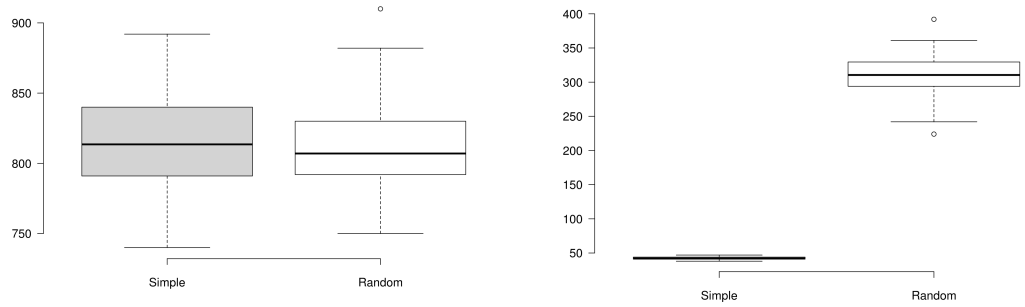

Figura 3: Distribución de puntaje obtenido (a la izquierda) y número de iteraciones realizadas (a la derecha) en una instancia aleatoria utilizando 100 reintentos

En estos gráficos se puede ver claramente que el número de iteraciones realizadas al iniciar con una solución inicial con asignaciones aleatorias es mucho mayor comparada a las instancias con asignaciones mínimas necesarias. A pesar de este aumento en la cantidad de iteraciones, los puntajes obtenidos tienen distribuciones muy similares, en donde las medias también son comparables, es decir, la instancia inicial no afectará a la solución final, si no que sólo aumentará el número de iteraciones necesaria, aumentando así el tiempo de resolución.

Además de variar la cantidad máxima de iteraciones y el número de reintentos, se varió múltiples veces la penalización dada por cada restricción blanda, pero no se encontró algún efecto además de la inherente variación en los puntajes finales obtenidos efecto de esta modificación en las penalizaciones. Debido a esto, estos parámetros fueron configurados de forma muy similar a lo sugerido por el modelo presentado en la Sección 4.

9. Conclusiones

A pesar de existir por más de 40 años, aún es un problema en desarrollo y se han seguido postulando soluciones de forma frecuente. Debido a que es un problema que puede ser utilizado en muchas áreas y cada aplicación en el mundo real tendrá un conjunto muy variable de restricciones, es difícil realizar un modelo global y por consiguiente, una solución que funcione en cualquier caso.

Como pudo ser estudiado, la eficiencia de cada técnica dependerá del tamaño de la instancia, el tiempo de planificación, la cantidad de empleados, etc. Es por estas diferencia que cada una de las técnicas utilizadas se enfoca en un tamaño específico de instancia y depende del tipo de restricciones existentes.

La historia del problema está aún siendo escrita, ya que año a año se producen nuevas técnicas para su resolución, prueba de esto es la conferencia bi-anual PATAT [1], en donde se presentan los avances en el área de planificación automática.

En cuando a la implementación del algoritmo se puede denotar la gran importancia de realizar un número alto de reintentos, ya que al tener muchos optimos locales, *Hill Climbing* se queda atrapado muy fácilmente a las pocas iteraciones, problema que los reinicios puede mitigar. Debido a este mismo fenómeno, las soluciones se encuentran con mucha rapidez, permitiendo realizar muchos reintentos sin impactar de gran manera el tiempo de ejecución.

Como trabajo a futuro se espera poder comparar esta implementación con otras existentes, esperando poder modificar el algoritmo para que sea competente frente a lo existente hoy en día.

10. Bibliografía

Referencias

[1] International series of conferences on the practice and theory of automated timetabling, 1995.

[2] Uwe Aickelin and Kathryn A. Dowsland. An indirect genetic algorithm for a nurse-scheduling problem. *Computers & Operations Research*, 31(5):761 – 778, 2004.

[3] M.N. Azaiez and S.S. Al Sharif. A 0-1 goal programming model for nurse scheduling. *Computers & Operations Research*, 32(3):491 – 507, 2005.

[4] Burak Bilgin, Peter Demeester, Tony Wauters, Greet Vanden Berghe, and KaHo Sint-Lieven. A hyper-heuristic combined with a greedy shuffle approach to the nurse rostering competition. 2010.

[5] Curtois T Burke, E.K. New computational results for nurse rostering benchmark instances, 2010.

[6] Walter J. Gutjahr and Marion S. Rauner. An aco algorithm for a dynamic regional nurse-scheduling problem in austria. *Computers & Operations Research*, 34(3):642 – 666, 2007. Logistics of Health Care Management.

[7] M. Hasebe, T. Yamazaki, M. Ryumae, W. Wu, K. Nonobe, and A. Ikegami. A comparison of integer programming formulations and variable-fixing method for the nurse scheduling problem. In *2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 70–74, Dec 2017.

[8] INRC. The international nurse rostering competition, 2014.

[9] INRC-II. The second international nurse rostering competition, 2016.

[10] S. Mesa J. Rivera. An integer programming-based local search algorithm for the nurse scheduling problem. Master's thesis, Functional Analysis and Applications Research Group Universidad EAFIT, Medellín, Colombia, 2015.

[11] Operations Research and Scheduling Research Group. Operations research and scheduling research group dataset, 2011.

[12] Sara Ceschia;Nguyen Thi Thanh Dang;Patrick De Causmaecker;Stefaan Haspeslagh;Andrea Schaerf;. Second international nurse rostering competition (inrc-ii) — problem description and rules —. *Computer Science*, 2015.

[13] Seyda Topaloglu and Hasan Selim. Nurse scheduling using fuzzy modeling approach. *Fuzzy Sets and Systems*, 161(11):1543 – 1563, 2010. Theme: Decision Systems.

[14] Christos Valouxis, Christos Gogos, George Goulas, Panayiotis Alefragis, and Efthymios Housos. A systematic two phase approach for the nurse rostering problem. *European Journal of Operational Research*, 219(2):425 – 433, 2012.

[15] G. Weil, K. Heus, P. Francois, and M. Poujade. Constraint programming for nurse scheduling. *IEEE Engineering in Medicine and Biology Magazine*, 14(4):417–422, Jul 1995.

[16] Harvey Wolfe and John P. Young. Staffing the nursing unit: Part i. controlled variable staffing. *Nursing Research*, 14(3), 1965.

[17] Harvey Wolfe and John P. Young. Staffing the nursing unit part ii. the multiple assignment technique. *Nursing Research*, 14(4), 1965.