Inteligencia Artificial

Informe Final: Office Space Allocation Problem

Manuel Matus H.

17 de marzo de 2020

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

Office space alocation problem es un problema de la vida real recurrente en instituciones de todo el mundo, el cual consiste en la asignación de espacios a personal o maquinaria de dicha organización. En el presente trabajo se proporcionará una revisión de la investigación realizada y relacionada al problema presentando su respectivo modelo matemático con miras de desarrollar una solución usando la técnica de búsqueda local incompleta Hill Climbing mejor mejora.

1. Introducción

Office Space Allocation Problem (OSAP) es un problema de optimización combinatoria recurrente en organizaciones los cuales buscan determinar la forma más adecuada para la asignación de espacios sujetos a las limitaciones físicas propias de estos además de las necesidades logísticas propias de cada institución. En el presente documento se planteará la definición propia del problema con su correspondiente formulación matemática, por otro lado se hará una revisión respecto al trabajo desarrollado frente a este desafío, para posteriormente utilizar un método de resolución propuesta en base a Hill climbing en su variante mejor mejora con miras de explorar y contrastar las soluciones generadas por algoritmos iterativos incrementales presentes en el estado del arte. La estructura del documento presenta en su sección 2 la definición del problema, luego en su sección 3 el estado del arte, posteriormente en la sección 4 el correspondiente modelo matemático para luego seguir con la representación de la solución en el punto 5, la descripción del algoritmo planteado en el puntos 6, continuando con la descripción del experimento realizado con sus resultados en las secciones 7 y 8 respectivamente, para así finalizar con la correspondiente conclusión.

2. Definición del Problema

Office-space allocation problem consiste en la asignación de un conjunto de entidades de diferente tamaño a un conjunto de espacios de distintas dimensiones (espacio), las cuales están sujeta a un grupo de restricciones que pueden ser duras o blandas (Hard Constraints y Soft Constraints), debido a su naturaleza OSAP pertenece a la familia de los problemas Bin-Packing [14], en los cuales se tiene un conjunto de objetos de diferentes volúmenes que deben ser empacados en un conjunto finito de containers. Por lo tanto, es posible destacar dos ascpectos de OSAP, en primer lugar la dificultad inherente del problema al ser NP-Hard [10] lo cual genera que buscar la solución global sea altamente costoso en tiempo y/o memoria, y en segundo lugar su presencia recurrente en instituciones corporativas y universitarias [13]

Para el modelado usualmente se hace uso de variables binarias para representar la asignación de una entidad e a una habitación r, en consecuencia, debido a que el problema consiste en la asignación de espacios sujeto a restricciones, el objetivo a alcanzar es la minimización de espacios mal usados y la violación de restricciones blandas, respecto a las restricciones estas se clasifican de la siguiente forma:

Tipo		Descripción
Asignación	al	La entidad e debe estar en cierta habitación r
Asignación prohibida	na	La entidad e no debe estar en cierta habitación r
Misma habitación	sr	Dos entidades e_1 y e_2 deben compartir habitación
Distinta habitación	nsr	Dos entidades e_1 y e_2 no deben compartir habitación
Habitación no compartida	nsh	La entidad e_1 no debe compartir habitación
Adyacencia	ad	Dos entidades e_1 y e_2 deben estar adyacentes
Agrupadas por	gr	Dos entidades e_1 y e_2 deben estar cercanas
Lejanía	aw	Dos entidades e_1 y e_2 deben estar alejadas
Capacidad	cp	La habitación r no debe estar con sobre uso

Cuadro 1: Restricciones

Finalmente, respecto a las variaciones conocidas del problema se encuentra Space-planning problem [1], que consiste en la asignación de funciones administrativas y personal a oficinas, o bien variaciones de OSAP con una menor cantidad de restricciones.

3. Estado del Arte

En diversas instituciones ha existido un problema recurrente, el de asignar habitaciones para máquinas y personas, los primeros trabajos relacionados al abordaje y solución de este desafío se presencian con Ritzman et al [13]., en 1980 los investigadores trabajaron en el desarrollo de una solución en base a programación lineal para resolver las asignaciones de espacios en la universidad estatal de Ohio, posteriormente Giannikos [9] en 1995 planteó una solución en base a la optimización por objetivos (Multiobjective Optimization) bajo el método de la programación por metas (Goal programming) para dar una solución al problema en la universidad Westminster UK, mientras que en 1999 Bland J.A [1], abordó el problema de la asignación óptima de actividades a ubicaciones, similar a OSAP, usando el algoritmo ACO (ant colony optimization).

Sin embargo, los mayores avances fueron desarrollados a partir de 1998 con el trabajo de Burke E.K y Varley D.B [6] en el cual catalogaron la asignación de espacios de diversas universidades británicas logrando capturar las principales restricciones que han sido usadas para la definición formal del problema, posteriormente en el mismo año plantean por primera vez el uso de metaheurísticas como Hill-climbing, Simulated Annealing y Genetic Algorithm [7], para la resolución del problema enfocándose en el porcentaje de utilización de espacios, obteniendo

resultados aceptables para los tres algoritmos dependiendo del grado de castigo determinado por romper las restricciones.

Más tarde en 2001 Burke et al., retoman el problema volviendo a usar Hill-climbing, Simulated Annealing y Genetic Algorithm [5], esta vez estableciendo diversas formas para la exploración de los vecindarios correspondientes a una solución candidata, además optó por un vector como representación de dicha solución $x_e = r$, es decir la habitación r es asignada a la entidad e, finalmente plantearon la posibilidad de usar algoritmos híbridos para obtener soluciones con el fin de alcanzar un mejor desempeño.

Posteriormente E. K. Burke, P. Cowling y J.D. Landa Silva propusieron un enfoque híbrido en base mejorar múltiples soluciones candidatas (population-based meta-heuristic) [4], para esto usaron Hill-climbing para buscar buenas soluciones candidatas iniciales, luego Simulated Annealing para mejorar dichas soluciones y finalmente una serie de heurísticas para explorar el espacio de búsqueda, respecto a la representación de las soluciones se usó un hash mapping en cual la llave era la entidad a ser posicionada y el valor de esta la habitación, además las restricciones eran conectadas a la correspondiente entidades o espacios. Por último lograron obtener resultados en la búsqueda de soluciones individuales de buena calidad y múltiples asignaciones de buena calidad.

En 2007 Dario Landa-Silva et al [11], debido a la complejidad computacional intrínseca del problema desarrollan un enfoque de búsqueda local cooperativa asíncrona entre algoritmos los cuales se seleccionó Hill-Climbing (HC), Simulated Annealing (SA), Tabu Search(TS) y una metaheurística híbrida(HMH) , además determinan experimentalmente para la exploración de vecindarios la generación k = n/3 vecinos siendo n el número de entidades, respecto a la exploración de estos vecindarios se plantean tres movimientos. El primero de ellos Relocate que consiste en reasignar una entidad a una nueva habitación, el segundo Swap el cual cambia a dos entidades de habitación entre si e Interchange que a partir de dos habitaciones intercambia todas las entidades asignadas a la habitación opuesta. Finalmente, respecto a las conclusiones obtenidas se logró determinar que el enfoque cooperativo asíncrono logra una mejor performance que algoritmos individuales.

Más tarde Özgür Ülker y Dario Landa-Silva proponen un modelo de programación lineal entera (0/1 Integer Programming Model for the Office Space Allocation Problem) [16] con el fin de ser aplicable a contextos reales, el modelo matemático planteado resultaría ser el más utilizado dado que define tanto restricciones duras como blandas. De forma paralela Rui lopes y Daniela Girimonte [12] buscando resolver la asignación de lugares en la agencia espacial europea usando los cuatro métodos recurrentes (HC, SA, TS y HMH) logran determinar un mejor desempeño en Tabu Search para búsquedas cortas mientras que para búsquedas extensas (suficientes iteraciones) HMH logra un mejor desempeño, por otro lado, se alcanzaron mejores resultados usando búsqueda local al recombinar soluciones usando operadores como swap y rellocate.

Entre 2010 y la actualidad varios trabajos han sido desarrollados con un enfoque de mejora respecto a los algoritmos ya usados en la literatura, en 2011 en base al modelo matemático 0/1 integer programming (IP) se obtuvieron los mejores resultados obtenidos hasta ese momento haciendo uso de un IP solver [15], posteriormente se abordó el problema haciendo uso de algoritmos genéticos híbridos también denominados como Memetic algorithm (MA) que en base a la mejora de un población de soluciones termina logrando mejores resultados que el trabajo previamente mencionado.

En consecuencia, OSAP también fue abordado desde distintos enfoques, los investigadores F. Castillo, M. C. Riff y E. Montero [8] desarrollaron un algoritmo híbrido entre Greedy Algorithm y Tabu search, enfocándose en las etapas de construcción y mejora de la solución, además de determinar los parámetros iniciales seleccionables mediante un software, obteniendo buenos resultados respecto a algoritmos como IP y MA concluyendo la importancia y potencial en combinar métodos y algoritmos para encontrar soluciones.

En la actualidad el problema ha sido abordado desde dos enfoques aplicando nuevos algorit-

mos como Artificial Bee Colony (ABC) [2] o bien proponiendo nuevas variantes de algoritmos ya conocidos como late acceptance Hill-Climbing (LAHC) [3].

4. Modelo Matemático

Continuando con el modelo correspondiente a Office Space Allocation Problem se define de similar manera que en [15] y [8].

4.1. Parámetros

- R Conjunto de habitaciones
- E Conjunto de entidades
- J Conjunto de restricciones, J = al,na,sr,nsr,nsh,ad,gr,aw,cp
- S_e Tamaño de la entidad e, $\forall e \, \epsilon \, E$
- C_r Capacidad de la habitación r, $\forall r \, \epsilon \, R$
- A_r Habitaciones adyacentes a r, $\forall r \, \epsilon \, R$
- N_r Habitaciones cercanas a r, $\forall r \, \epsilon \, R$
- HC^j Conjunto de restricciones duras j, $\forall j \, \epsilon \, J$
- SC^j Conjunto de restricciones blandas j, $\forall j \, \epsilon \, J$

4.2. Variables

Se determinan dos tipos de variables, una para la asignación de habitaciones y una segunda variable auxiliar para la satisfacción de restricciones blandas.

$$x_{er} = \begin{cases} 1 & \text{si la entidad e es asignada a la habitación r,} \forall e \, \epsilon \, E, r \, \epsilon \, R \\ 0 & \text{en otro caso.} \end{cases} \tag{1}$$

$$y^j(i) = \begin{cases} 1 & \text{si la restricción i tipo j no es respetada,} \forall i \, \epsilon \, |SC|^j, j \epsilon J \\ 0 & \text{en otro caso.} \end{cases} \tag{2}$$

4.3. Restricciones Duras

1. Todas asignadas: Cada entidad $e \, \epsilon \, E$ debe ser asignada a exactamente una habitación.

$$\sum_{R \epsilon r} x_{er} = 1 \qquad \forall e \, \epsilon \, E \tag{3}$$

2. Asignación: Una entidad e debe ser asignada a una habitación r.

$$x_{er} = 1 \tag{4}$$

3. Asignación prohibida: Una entidad e no debe ser asignada a una habitación r.

$$x_{er} = 0 \tag{5}$$

4. Misma habitación: Dos entidades e_1 y e_2 deben estar en la misma habitación.

$$x_{e_1 r} - x_{e_2 r} = 0 \qquad \forall r \, \epsilon \, R \tag{6}$$

5. Distinta habitación: Dos entidades e_1 y e_2 no deben estar en la misma habitación.

$$x_{e_1 r} + x_{e_2 r} \leq 1 \qquad \forall r \, \epsilon \, R \tag{7}$$

6. Habitación no compartida: La entidad e no debe compartir la habitación con ninguna otra entidad.

$$\sum_{f \epsilon E - e} x_{fr} \leq (|E| - 1) - (|E| - 1) \cdot x_{er} \qquad \forall r \, \epsilon \, R \tag{8}$$

7. Adyacencia: Entidades e_1 y e_2 deben ser ubicadas en habitaciones adyacentes.

$$x_{e_1 r} \leq \sum_{s \epsilon \lambda_r} x_{e_2 s} \leq 1 \qquad \forall r \, \epsilon \, R \tag{9}$$

8. Agrupadas por: Entidades en un grupo deben ser ubicadas cerca del cabeza de grupo f.

$$x_{er} \leq \sum_{s \epsilon N_r} x_{fs} \leq 1 \qquad \forall r \, \epsilon \, R \tag{10}$$

9. Lejanía: Entidades e_1 y e_2 deben ser ubicadas en habitaciones lejanas entre ellas.

$$0 \leq \sum_{s \epsilon N_r} x_{e_2 s} \leq 1 - x_{e_1 r} \qquad \forall r \, \epsilon \, R \tag{11}$$

10. Capacidad: Habitación r no debe estar en sobre uso.

$$\sum_{e \epsilon E} S_e x_{er} \leq C_r \tag{12}$$

4.4. Restricciones Blandas

1. Asignación: $y^{al}(i)$ es uno si $SC^{al}(i)$ no está satisfecha.

$$y^{al}(i) = 1 - x_{er} \tag{13}$$

2. Asignación prohibida: $y^{na}(i)$ es uno si $SC^{na}(i)$ no está satisfecha.

$$y^{na}(i) = x_{er} \tag{14}$$

3. Misma habitación: $y^{sr}(i)$ es uno si $SC^{sr}(i)$ no está satisfecha.

$$2y_r^{sr}(i) - 1 \leq x_{e_1 r} - x_{e_2 r} \leq 1 - \epsilon + \epsilon \cdot y_r^{sr}(i) \qquad \forall r \, \epsilon \, R \tag{15}$$

$$y^{sr}(i) = \sum_{r \epsilon R} y_r^{sr}(i) \tag{16}$$

4. Distinta habitación: $y^{nsr}(i)$ es uno si $SC^{nsr}(i)$ no está satisfecha.

$$(1 + \epsilon) - (1 + \epsilon) y_r^{nsr}(i) \leq x_{e_1 r} + x_{e_2 r} \leq 2 - y_r^{nsr}(i) \qquad \forall r \, \epsilon \, R \tag{17}$$

$$y^{nsr}(i) = \sum_{r \epsilon R} (1 - y_r^{nsr}(i)) \tag{18}$$

5. Habitación no compartida: $y^{nsh}(i)$ es uno si $SC^{nsh}(i)$ no está satisfecha.

$$(|E|-1)(2-x_{er}-y_r^{nsh}(i)) \leq \sum_{f \epsilon E-e} x_{fr} \qquad \forall r \,\epsilon\, R \tag{19}$$

$$\sum_{f \epsilon E-e} x_{fr} \leq (|E|-1)(1-x_{er}) + \epsilon - (|E|-1+\epsilon) \cdot y_r^{nsh}(i) \qquad \forall r \,\epsilon\, R \tag{20}$$

$$y^{nsh}(i) = \sum_{r \epsilon R} (1 - y_r^{nsh}(i)) \tag{21}$$

6. Adyacencia: $y^{ad}(i)$ es uno si $SC^{ad}(i)$ no está satisfecha.

$$y_r^{ad}(i) + x_{e_1 r} - 1 \leq \sum_{s \epsilon A_r} x_{e_2 s} \leq x_{e_1 r} - \epsilon + (1+\epsilon) \cdot y_r^{ad}(i) \qquad \forall r \,\epsilon\, R \tag{22}$$

$$y^{ad}(i) = \sum_{r \epsilon R} (1 - y_r^{ad}(i)) \tag{23}$$

7. Agrupadas por: $y^{gr}(i)$ es uno si $SC^{gr}(i)$ no está satisfecha.

$$y_r^{gr}(i) + x_{er} - 1 \leq \sum_{s \epsilon N_r} x_{fs} \leq x_{er} - \epsilon + (1+\epsilon) \cdot y_r^{gr}(i) \qquad \forall r \,\epsilon\, R \tag{24}$$

$$y^{gr}(i) = \sum_{r \epsilon R} (1 - y_r^{gr}(i)) \tag{25}$$

8. Lejanía: $y^{aw}(i)$ es uno si $SC^{aw}(i)$ no está satisfecha.

$$1 - x_{e_1 r} + \epsilon - (1+\epsilon) \cdot y_r^{aw}(i) \leq \sum_{s \epsilon N_r} x_{e_2 s} \leq 2 - x_{er} - y_r^{aw}(i) \qquad \forall r \,\epsilon\, R \tag{26}$$

$$y^{aw}(i) = \sum_{r \epsilon R} (1 - y_r^{aw}(i)) \tag{27}$$

9. Capacidad: $y^{cp}(i)$ es uno si $SC^{cp}(i)$ no está satisfecha.

$$\sum_{e \epsilon E} S_e x_{er} + (C_r + \epsilon)(1 - y^{cp}(i)) \geq C_r + \epsilon \tag{28}$$

$$\sum_{e \epsilon E} S_e x_{er} + (\sum_{e \epsilon E} S_e - C_r)(1 - y^{cp}(i)) \leq \sum_{e \epsilon E} S_e \tag{29}$$

4.5. Función objetivo

La función objetivo representa el espacio mal utilizado, es decir por sobre uso y bajo uso, además de considerar una penalización por restricciones blandas no respetadas. Considerando w^j como la penalización correspondiente a la restricción j, $j \epsilon SC^j$. Por lo tanto todas las restricciones duras deben ser satisfechas $\sum_{i=1}^{|HC^j|} y_i^j = 0$.

- Espacio desaprovechado por habitación

$$WP_r = max(0, C_r - \sum_{e \epsilon E} x_{er} \cdot S_e) \qquad \forall r \,\epsilon\, R \tag{30}$$

- Espacio excedido por habitación

$$OP_r = max(0, 2 \cdot (\sum_{e \epsilon E} x_{er} S_e - C_r)) \qquad \forall r \; \epsilon \; R \tag{31}$$

- Espacio mal utilizado

$$f_1(x) = \sum_{r \epsilon R} WP_r + \sum_{r \epsilon R} OP_r \tag{32}$$

- Penalización por restricciones no cumplidas

$$f_2(x) = \sum_{j \epsilon J} w^j \sum_{i=1}^{|SC^j|} y_i^j \tag{33}$$

- Función objetivo

$$min \quad f(x) = f_1(x) + f_2(x) \tag{34}$$

5. Representación

Para la resolución del problema se plantea una lista de enteros como la representación de la solución, dicha lista guarda en cada posición la habitación r_j a la cual una entidad e_i es asignada. De esta manera los movimientos para crear el vecindario en cada iteración (Swap Entities, Relocate Entity e Interchange) son compatibles con la estructura seleccionada. Además es fácil corroborar que las soluciones generadas cumplen con las restricción dura "Todas Asignadas"dado que la lista debe estar completamente asignada.

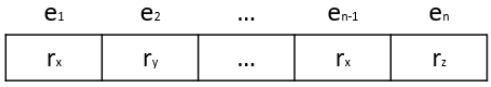

Figura 1: Representación de solución

6. Descripción del algoritmo

Para la resolución de OSAP se plantea un algoritmo de búsqueda incompleta, es decir Hill-Climbing mejor mejora con restart (HCMM+R) el cual mejora soluciones iniciales generadas por un algoritmo greedy con miras de lograr una amplia diversificación y por ende resultados que compitan con algoritmos como LAHC[3] e IIpb[11], por lo que se presentará a continuación la función de evaluación usada para medir la calidad de las soluciones obtenidas, los movimientos que crean los vecindarios de las soluciones, la construcción de la solución inicial factible mediante un algoritmo greedy y finalmente la implementación de HCMM+R.

6.1. Función de evaluación

Como función de evaluación se hace uso de la función objetivo expresada en la ecuación 34, por lo que la calidad de las soluciones será determinada por dos componentes, en primer lugar la suma total de penalizaciones por las restricciones blandas insatisfechas y en segundo lugar el espacio mal usado ya sea por ser desperdiciado o bien sobrecargado.

6.2. Movimientos

Para la creación de vecindarios creado a partir de una solución actual, se ha optado por tres movimientos ampliamente usados en el estado del arte, teniendo especial cuidado en generar soluciones vecinas factibles.

- Relocate, a partir de una entidad e_i esta es reasignada a una habitación aleatoria.
- Swap, dada dos entidades e_i y e_j estás son intercambiadas de habitación.
- Interchange, las entidades pertenecientes a las dos habitaciones r_i y r_j son intercambiadas a la habitación opuesta.

6.3. Solución inicial aleatoria

Como parte de la propuesta para la creación de las soluciones iniciales factibles aleatorias se hace uso de un algoritmo greedy adaptado similar al planteado en [8], en este caso las entidades son separadas en tres grupos, el primer grupo corresponde a las entidades denominadas grandes que cumplen con el siguiente criterio:

$$S_e > \overline{space} + 2 \cdot \sigma_{space} \tag{35}$$

luego se encuentran la entidades relacionadas a restricciones duras y finalmente el resto de entidades, por lo que el algoritmo toma una entidad aleatoria del primer grupo y la asigna en la habitación que minimice la función de evaluación y mantenga la factibilidad de la solución, así hasta que aleatoriamente todas las entidades de dicho grupo son asignadas, posteriormente continúa de manera similar con los siguientes grupos de entidades. Generando de esta manera las soluciones iniciales de manera distinta a la normalmente llevado a cabo en el estado del arte para algoritmos Hill-Climbing, los cuales acostumbran llevar a cabo la inicialización mediante un algoritmo greedy peckish ([3], [11], [12]) que asignan las entidades en la mejor habitación de un subgrupo de k piezas que cumplen con una proporción de espacio entre la entidad y dicho grupo de habitaciones.

6.4. Hill Climbing Mejor mejora + Restart

Para la implementación (Algorithm 1) de Hill-Climbing mejor mejora con restarts (HCMM+R) , se plantea la diversificación mediante dichos restarts además de soluciones iniciales factibles aleatorias buscadas en cada iteración del loop principal, el cual esta determinado por un parámetro de iteraciones. Luego el algoritmo a partir de la solución inicial se dedica a intensificar creando vecindarios, para posteriormente tomar la mejor solución vecina y determinar si convertirla en la solución actual si y solo si muestra una calidad más deseada que la solución presente. Finalmente HCMM+R va guardando la mejor solución obtenida a partir de las soluciones iniciales construidas, para finalmente retornar la mejor solución factible encontrada.

Algorithm 1 Hill Climbing Mejor Mejora + Restart

```
inicializar S_best
while loops budget do
    local = True
    S_c = NuevaSolucionInicial()
    while local do
        move = seleccionar un movimiento random
        S_n = Vecindario (move, S_c);
        if S_n es mejor que S_c then
            S_c = S_n
        else
            local = False
        end if
    end while
    if S_c es mejor que S_best then
        S_best = S_c
    end if
end while
```

6.5. Parámetros

Respecto a los parámetros usados en HCMM+R se encuentra solo uno.

- *n_loops*, este parámetro determina la cantidad de reinicios que lleva a cabo el algoritmo, es importante determinar su valor ya que en HCMM+R, la diversificación es influenciada por el Restart y la aleatoriedad en las soluciones iniciales.

7. Experimentos

7.1. Configuración del experimento

El computador en el cual se llevo a cabo la prueba consta con un procesador Intel Core I3-5005U 2.0 GHz y con 4GB de RAM, teniendo como sistema operativo Ubuntu 18.04.04 (64 bit). Respecto a la configuración para la prueba de las instancias se procedió de determinar la penalización particular para cada restricción.

Restricción	Castigo
al	20
nal	10
sr	10
nsr	10
nsh	50
ad	10
gr	10
aw	10
cp	10

Cuadro 2: Castigos por restricción

Respecto a las instancias utilizadas se usaron dos conjuntos, en primer lugar las instancias PNe150 de [15] conjunto el cual fue creado matemáticamente para generar instancias de OSAP

difíciles, por otro lado también se hizo uso de las instancias Nott de la universidad de Nottingham [14] con el fin de probar HCMM+R en instancias reales.

	Nott1	Nott1b	Nott1c	Nott1d	Nott1e	PNe150
Nro de entidades	158	104	94	56	86	150
Nro de Habitaciones	131	77	94	56	59	92
Nro de pisos	3	3	3	3	3	3
Nro de restricciones	272	136	189	88	100	263
Nro de restricciones duras	111	37	94	49	43	67
Nro de restricciones blandas	161	99	95	39	57	196

Cuadro 3: Data instancia Nott y PNe150

Posteriormente la elección del valor del parametro *n_loops* fue seleccionado haciendo observaciones en el estado del arte probando de esta manera 1000, 5000 y 10000 iteraciones o reinicios, por lo cual se optó por elegir 1000 iteraciones observando en los resultados obtenidos un tiempo promedio de 5 a 10 minutos de ejecución además de no encontrar una mayor variación en los resultados al extender a 5000 iteraciones.

7.2. Diseño

En lo concerniente a la experimentación está se centró en dos enfoques, el primero de ellos determinar la mejor combinación de movimientos que construyen el vecindario a partir de la solución greedy aleatoria utilizada, las combinaciones utilizadas fueron cada operador por si solo y la ejecución de los tres siendo escogido uno de ellos de forma aleatoria, todo esto llevado a cabo en las instancias PNe150 y Nott.

Posteriormente con los resultados ya obtenidos se procede a comparar la idoneidad de los movimientos comparar los resultados del conjunto Nott con los resultados obtenidos por el algoritmo Late Acceptance Hill Climbing (LAHC) [3] e iterative improvement (II_{pb}) [11].

8. Resultados

En primer lugar se revisan los movimientos utilizados en la instancia PNe150 , disponibles en Cuadro 4 ,obteniendo los mejores resultados al usar los 3 movimientos, pero principalmente al usar Interchange, entendiendo HCMM+R como un algoritmo que converge rápidamente a óptimos locales con los movimientos Relocate y Swap, sin embargo al hacer uso de Interchange el algoritmo logra llegar a mejores soluciones, esto se puede explicar debido a que al mover bloques de entidades se mejora el uso de los espacios y se sigue respetando restricciones como sameroom, notsameroom, etc.

Respecto los resultados obtenidos para las instancias Nott se muestra una mayor variación sobre cual movimiento genera mejores soluciones a partir de la solución inicial planteada mostrando mejores resultados en las instancias Nott1b, Nott1d y Nott1e las cuales poseen la menor cantidad de restricciones duras, por lo que el movimiento en los vecindarios no se estanca rápidamente en óptimos locales a HCMM+R.

Considerando los resultados obtenidos para ambos conjuntos de instancias se puede apreciar la influencia que tiene la cantidad de restricciones, entidades y piezas en la búsqueda de una solución, siendo estos tres factores los que determinan y representan en si la dificultad computacional innata de OSAP, es por ello que algoritmos de búsqueda local que de cierta manera implementan el escape de óptimos locales en el estado del arte logran mejores rendimientos ([8], [11], [3]).

	Relocate		Swap		Interchange		All	
	avg	min	avg	min	avg	min	avg	min
p000n000	3176.97	2555	3158.76	2651.5	2728.47	1932	3160.65	2378
p000n005	3209.36	2612.4	3173.96	2563.7	2666.37	1734.1	3166.24	2527.4
p000n010	3179.03	2601.7	3148.86	2560.1	2672.9	1902.8	2671.26	1815.2
p000n015	3217.07	2623.2	3194.48	2563.2	2692.63	1805.7	2716.8	1952.9
p000n020	3154.14	2472.5	3117.08	2423	2688.63	1911.1	3131.77	2527.1
p000n025	3261.34	2724.4	3246.85	2564.4	2755.19	1993.3	2757.94	1969.7
p005n000	3135.18	2406.9	3112.76	2461.7	2661.27	1799.7	3114.01	2389.2
p005n005	3233.11	2595.8	3197.84	2443.4	2586.56	1742.3	3207.9	2544.1
p005n010	3098.95	2501	3066.4	2505	2561.86	1714.7	2554.79	1780.6
p005n015	3211.35	2442.2	3178.92	2563.4	2638.98	1900.2	3180.83	2455.9
p005n020	3250.94	2635.5	3223.6	2486.8	2623.43	1709.2	3218.71	2534.7
p005n025	3206.58	2634	3176.29	2526.8	2658.65	1795.7	3162.36	2497.4
p010n000	3159.75	2511.9	3149.57	2437.1	2660.5	1739.7	2657.65	1838.5
p010n005	3189.43	2642.4	3169.71	2583	2599.33	1761.2	3201.42	2603.2
p010n010	3185.48	2630.8	3162.98	2478	2583.24	1838.9	3198.39	2615.1
p010n015	3352.91	2721.2	3330.23	2758.8	2678.4	1907.6	3361.67	2823.6
p010n020	3237.32	2645.1	3219.1	2673.1	2672.14	1741.2	3215.73	2570.5
p010n025	3196.42	2541.9	3156.22	2541.2	2660.18	1765.3	3173.81	2463.6
p015n000	3152.63	2535.1	3133.44	2392.2	2643.14	1698.6	3143.86	2507.1
p015n005	3150.86	2604.9	3131.28	2578.7	2585.55	1719	3122.03	2453.6
p015n010	3177.15	2594.6	3163.52	2528.4	2584.98	1770.4	2590.3	1797.6
p015n015	3223.4	2668.7	3201.16	2402.9	2618.15	1699	2629.25	1822.7
p015n020	3166.24	2588.7	3138.2	2399.5	2603.09	1790.6	2605.61	1812.1
p015n025	3220.9	2605.8	3187.34	2488.2	2663.75	1881.4	3185.92	2508.3
p020n000	3182.47	2340	3156.39	2560.1	2660.14	1770.3	3184.44	2551
p020n005	3232.48	2599.8	3215.49	2391.9	2628.89	1527.3	3227.13	2450.3
p020n010	3238.14	2589.4	3219.95	2609.1	2625.83	1684.9	3222.02	2542.3
p020n015	3160.44	2466.2	3132.01	2541.2	2613.68	1751.2	3151.86	2548.9
p020n020	3222.18	2538.9	3193.07	2553.1	2632.08	1759	3228.43	2591.5
p020n025	3273.77	2634.4	3249.61	2591.2	2697.2	1930.6	3272.87	2528.5
p025n000	3206.99	2458.8	3186.65	2339.3	2664.98	1800.7	3212.47	2370.1
p025n005	3191.62	2421.3	3155.61	2525.9	2584.71	1749.4	2597.2	1720.1
p025n010	3253.08	2666.6	3233.53	2608	2667.43	1838.8	3220.86	2501.2
p025n015	3225.9	2670.5	3186.74	2468.6	2635.55	1840.1	3197.07	2643.3
p025n020	3205.33	2627.3	3162.81	2580.4	2611.41	1801.8	3270.68	2595
p025n025	3330.46	2588.6	3299.49	2587.3	2698.79	1770.1	3341.6	2755.6

Cuadro 4: Resultados instancia PNe150

	Relocate		Swap		Interchange		All	
	avg	min	avg	min	avg	min	avg	min
Nott1	2184.54	1604.65	2049.4	1441.1	1901.15	1276.45	2064.99	1549.25
Nott1b	898.88	660.95	877.63	705	858	665.45	852.87	674.7
Nott1c	1922.08	1221.15	1709.14	1048.75	1708.37	1011.45	1938.43	1326.45
Nott1d	463.54	348.3	438.32	337.4	440.84	350	438.27	326.5
Nott1e	648.23	354.7	639.13	353.8	642.21	339.9	634.99	344.7

Cuadro 5: Resultados instancia Nott

	HCMM+R	LAHC	IIpb
Nott1	1276.45	393.8	568.1
Nott1b	660.95	332.5	468.4
Nott1c	1011.45	356.6	348.2
Nott1d	326.5	288.3	-
Nott1e	339.9	137.7	-

Cuadro 6: Resultados HCMM+R, IIpb y LAHC

Respecto al cuadro 6, se aprecia como variaciones de Hill Climbing como LAHC e II_{pb} logran buenos resultados, de esta manera al observar Late acceptance Hill climbing, el cual permiten de cierta manera aceptar soluciones de peor calidad comparando la solución actual con una solución antigua que se encuentra dentro de una lista (es por ello que se llama aceptación tardía), logra mejores resultados o bien iterative improvement population based el cual adapta Hill Climbing para mejorar una población de soluciones, implementando de esta manera diversificación. Por lo tanto, siguiendo la linea de lo previamente mencionado se puede entender de cierta manera como el operador Interchange logra mejores resultados que las otras tres variaciones planteadas, dado que la representación de la solución cambia en más de un punto al llevar a cabo el movimiento permitiendo al algoritmo explorar zonas más amplias.

9. Conclusiones

En conclusión, se puede determinar que Office-space allocation problem ha sido abordado con diferentes técnicas para resolver el problema, desde programación lineal hasta algoritmos de búsqueda incompleta, respecto a los algoritmos de búsqueda local como Hill Climbing estos han solido ser implementados siendo modificados con el fin de obtener un mejor rendimiento, es por ello que en este informe se propone hacer uso de la variante Hill climbing mejor mejora aplicando restart más un algoritmo greedy para la creación de soluciones iniciales. En consecuencia se determina la importancia de la diversificación para la familia de algoritmos correspondientes a HC, por lo que se determina que para el algoritmo con mejor mejora el movimiento Interchange posee una mayor aptitud al presentar una mejora notable frente al resto de movimientos, en general para las instancias NPe150 siendo un 40 % mejor que el resto.

Respecto a las desventajas de proponer un enfoque de mejor mejora es la rápida convergencia a óptimos locales que se evidencia al comparar con soluciones que implementan HC. Por lo tanto se remarca la importancia de adaptar el algoritmo para que este realice una mayor exploración del espacio de búsqueda

Por otro lado la elección del algoritmo para la creación de la solución inicial ayuda a precipitar a este a óptimos locales, ya que al construir la solución incial selecciona la mejor habitación posible para la entidad a asignar, volviendo de esta manera al movimiento Relocate poco frúctifero para comenzar la construcción de los primero vecindarios, de manera similar swap no presenta mayor mejora.

Finalmente respecto a los resultados obtenidos se plantea proponer una nueva explotación del vecindario optando por el movimiento Interchange en las etapas tempranas de la iteración para posteriormente intensificar con todos los movimientos planteados.

10. Bibliografía

Indicando toda la información necesaria de acuerdo al tipo de documento revisado. Todas las referencias deben ser citadas en el documento.

Referencias

[1] J. A. Bland. Space planning by ant colony optimisation. *Int. J. Comput. Appl. Technol.*, 12(6):320–328, July 1999.

[2] Asaju La'aro Bolaji, Ikechi Michael, and Peter Bamidele Shola. Optimization of office-space allocation problem using artificial bee colony algorithm. In Ying Tan, Hideyuki Takagi, and Yuhui Shi, editors, *Advances in Swarm Intelligence*, pages 337–346, Cham, 2017. Springer International Publishing.

[3] Asaju La'aro Bolaji, Ikechi Michael, and Peter Bamidele Shola. Adaptation of late acceptance hill climbing algorithm for optimizing the office-space allocation problem. In Maria J. Blesa Aguilera, Christian Blum, Haroldo Gambini Santos, Pedro Pinacho-Davidson, and Julio Godoy del Campo, editors, *Hybrid Metaheuristics*, pages 180–190, Cham, 2019. Springer International Publishing.

[4] E. K. Burke, P. Cowling, and J. D. Landa Silva. Hybrid population-based metaheuristic approaches for the space allocation problem. In *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, volume 1, pages 232–239 vol. 1, May 2001.

[5] E. K. Burke, P. Cowling, J. D. Landa Silva, and Barry McCollum. Three methods to automate the space allocation process in uk universities. In Edmund Burke and Wilhelm Erben, editors, *Practice and Theory of Automated Timetabling III*, pages 254–273, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[6] E. K. Burke and D. B. Varley. Space allocation: An analysis of higher education requirements. In Edmund Burke and Michael Carter, editors, *Practice and Theory of Automated Timetabling II*, pages 20–33, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[7] E. K. Burke and D. B. Varley. Automating space allocation in higher education. In Bob McKay, Xin Yao, Charles S. Newton, Jong-Hwan Kim, and Takeshi Furuhashi, editors, *Simulated Evolution and Learning*, pages 66–73, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[8] Francisco Castillo, Maria-Cristina Riff, and Elizabeth Montero. New bounds for office space allocation using tabu search. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, pages 869–876, New York, NY, USA, 2016. ACM.

[9] Ioannis Giannikos, Elia El-Darzi, and Patrick Lees. An integer goal programming model to allocate offices to staff in an academic institution. *The Journal of the Operational Research Society*, 46(6):713–720, 1995.

[10] Hans Kellerer and Ulrich Pferschy. Cardinality constrained bin-packing problems. *Annals of Operations Research*, 92:335–348, 1999.

[11] Dario Landa-Silva and Edmund K. Burke. Asynchronous cooperative local search for the office-space-allocation problem. *INFORMS J. on Computing*, 19(4):575–587, October 2007.

[12] Rui Lopes and Daniela Girimonte. The office-space-allocation problem in strongly hierarchized organizations. In Peter Cowling and Peter Merz, editors, *Evolutionary Computation in Combinatorial Optimization*, pages 143–153, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[13] Larry Ritzman, John Bradford, and Robert Jacobs. A multiple objective approach to space planning for academic facilities. *Manage. Sci.*, 25(9):895–906, September 1979.

[14] Jesus Dario Landa Silva. Metaheuristic and multiobjective approaches for space allocation. *Doctoral dissertation, University of Nottingham*, 2003.

[15] Özgür Ülker and Dario Landa-Silva. Designing difficult office space allocation problem instances with mathematical programming. In Panos M. Pardalos and Steffen Rebennack, editors, *Experimental Algorithms*, pages 280–291, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[16] Özgür Ülker and Dario Landa-Silva. A 0/1 integer programming model for the office space allocation problem. *Electronic Notes in Discrete Mathematics*, 36:575 – 582, 2010. ISCO 2010 - International Symposium on Combinatorial Optimization.