Estado del avance: Office space allocation

Ignacio Quintana[1,1*]

[1*]Departamento de informática, Universidad técnica Federico Santa Maria .

Corresponding author(s). E-mail(s): ignacio.quintana@usm.cl;

Abstract

Office space allocation problem es un problema de optimización combinatoria cuyo objetivo es asignar un conjunto de entidades a un serie de habitaciones sujeto a un serie de restricciones características de un entorno organizacional como las interdependencias que existen entre grupos de personas. En este documento se explora el arte del problema detallando su origen, restricciones y modelo matemático junto con los distintos métodos de resolución que están siendo utilizados para abordarlo y como estos se comparan entre si.

Keywords: Combinatorial Optimization, Office space allocation, Heuristics, Artificial intelligence

1 Introducción

Office space allocation problem (OSAP), o problema de asignación de espacios de oficina, es un problema de optimización combinatoria con aplicaciones al mundo real que surge a menudo en universidades, empresas u organizaciones gubernamentales [1] de la necesidad de asignar un grupo de entidades ,como personas, a ciertas habitaciones satisfaciendo una serie de restricciones producto de las interacciones que existen entre entidades. *OSAP* pertenece a la clase de problemas de asignación de espacios los cuales pueden ser definidos como: *distribuir entidades a áreas de espacio, tales como habitaciones, asegurando una utilización eficiente y satisfaciendo tantas restricciones como sea posible.* [2]

El problema como tal fue planteado por primera vez en 1979 por Ritzman et al.[3] ante la necesidad de asignar oficinas a miembros de la facultad y

empleados de la Universidad del Estado de Ohio de una forma óptima y considerando una serie de restricciones como lo son la imposibilidad de remover paredes para remodelar los espacios existentes y las interdependencias que existen entre distintas unidades, lo cual implica que cierto personal deba asignarse en oficinas cercanas.

El objetivo de *OSAP* es asignar un conjunto de entidades, como lo pueden ser investigadores o ingenieros, a diferentes espacios dentro de una organización de una forma óptima tal que se minimice el espacio mal utilizado sujeto a una serie de restricciones como que cada entidad sea asignada a una habitación, la proximidad entre ciertas entidades, la compartición de las habitaciones, la asignación o no asignación de una habitación a una entidad en particular y que haya habitaciones que no pueden ser ocupadas sobre su capacidad.

El propósito de este documento es presentar el estado del arte con los avances en los métodos de representación y resolución del problema a modo de realizar una comparativa que permita identificar las mejores técnicas y los lineamientos bajo los cuales realizar futuras investigaciones.

Este documento tiene la siguiente estructura: definición del problema en donde se especifica el origen y restricciones, estado del arte donde se introducen y comparan las distintas técnicas de resolución, modelo matemático donde se formalizan el problema y sus restricciones como un problema de programación binaria y finalmente las conclusiones en donde el autor detalla las diferencias y ventajas de las técnicas actualmente utilizadas.

2 Definición del problema

Office space allocation problem consiste de un conjunto de habitaciones con una cierta capacidad a las cuales se deben asignar entidades, cada una con un tamaño determinado. Existen una serie de restricciones las cuales pueden ser ajustadas como blandas (que son deseables que se satisfagan) o duras (que deben ser satisfechas) de acuerdo a la naturaleza de las interacciones entre las entidades, como el colocar a investigadores de una área en una misma habitación, o las necesidades de la organización, como puede ser que miembros de una misma facultad se encuentren cercanos entre si [1].

Las restricciones a considerar son [4]:

1. Todas las entidades debe ser asignadas a exactamente una habitación, esta restricción siempre debe ser satisfecha (i.e es una restricción dura).
2. Hay entidades que deben ser asignadas a una habitación especifica.
3. Hay entidades que no deben ser asignadas a una habitación especifica.
4. Hay entidades que deben tener una habitación personal (i.e ninguna otra entidad debe ser asignada a la habitación), por ejemplo en el caso de los directores de las facultades.
5. Hay entidades que deben ser asignadas a una misma habitación.
6. Hay entidades que no deben ser asignadas a la misma habitación.
7. Hay entidades que deben estar en habitaciones adyacentes.

8. Hay entidades que deben ser asignadas a habitaciones cercanas, por ejemplo en el caso de un grupo de investigación.
9. Hay entidades que deben ser asignadas en habitaciones lejanas.
10. Hay habitaciones que no pueden ser usadas por sobre su capacidad.

El objetivo de *OSAP* es asignar las entidades de manera tal que se minimice el espacio sobre-utilizado o infrautilizado en las habitaciones, es decir, minimizar el espacio mal utilizado satisfaciendo todas las restricciones duras y la mayor cantidad de restricciones blandas que sea posible.

Los parámetros del problema son:

1. Cantidad de entidades.
2. El espacio ocupado por cada una de las entidades.
3. Cantidad de habitaciones.
4. Capacidad de cada una habitaciones
5. Las habitaciones adyacentes.
6. Las habitaciones cercanas.
7. El conjunto restricciones.
8. El tipo de la restricción (dura o blanda).

Office space alllocation problem es considerado una variante de *bin packing problem*, *knapsack problem* y los problemas de asignación generalizada los cuales se encuentran en la clase de complejidad NP-Hard [5] dentro de la cual también se encuentra *OSAP* [6]. Algunas variantes del problema son:

- The Office-Space-Allocation Problem in Strongly Hierarchized Organizations [7]

  Variante de *OSAP* usada en la Agencia Espacial Europea (*ESA*), esta adaptada a organizaciones dinámicas en donde existe una alta jerarquización en forma de diversos equipos que deben mantenerse trabajando de forma coherente en sus diversas tareas. La variante considera restricciones adicionales como evitar separar a los equipos en edificios o pisos diferentes y evitar zonas compartidas entre diferentes departamentos.
- NASA Office Space Allocation [5]

  Utilizado para la localización de espacios en NASA, en esta variante se extiende el objetivo no solo a minimizar el espacio mal utilizado sino también minimizar la distancia entre los empleados en una misma organización, lo cual incremente su sinergia, y a maximizar el numero de habitaciones vacantes.

3 Estado del arte

Uno de los primeros trabajos en *Office space allocation problem* fue presentado en 1979 por Ritzman et al. [3], en el los autores detallan como durante el invierno de 1976 dos departamentos externos al College de Ciencias Administrativas se trasladarían a otras instalaciones dentro del campus

de la Universidad del Estado de Ohio dejando oficinas libres. Esto ayudaría a descongestionar el College y a reunir funciones que estaban espacialmente separadas. El proceso para asignar las oficinas consistía de una serie de reuniones anuales entre los directos de los departamentos en la cual se negociaban las asignaciones de acuerdo a un único criterio de antigüedad en desmedro de asignar unidades interdependientes en espacios contiguos. Este proceso consumía demasiado tiempo y resultaba en asignaciones poco óptimas que requerían refinamiento razón por la cual se decidió traspasar el trabajo de asignación a una comisión especial. Sin embargo, el comité no tuvo éxito debido a falta de acuerdos y se conforma un segundo comité en donde se propone utilizar un modelo de programación lineal mixta (*MILP*) para asignar de forma equitativa habitaciones a los distintos departamentos de acuerdo a una colección de objetivos conflictivos entre si, como asignar suficientes habitaciones a los distintos departamentos y minimizar las desviaciones del estándar de uso de metros cuadrados, es decir, minimizar el espacio mal utilizado surgiendo así este problema de asignación.

En el año 2000 Burke et al. [2] utilizaron los algoritmos heurísticos *Hill climbing, Simulated annealing* y del tipo genéticos para abordar el problema mediante 3 operaciones de búsqueda local: *allocate* (asignar), *relocate* (reasignar) y *swap* (intercambiar) para recambiar regiones deficientes de la solución. En el caso de que se requiera reorganizar un asignación ya existente *Hill climbing* y *Simulated annealing* se desempeñan mejor que los algoritmos genéticos y logran una asignación mas óptima que la de un experto. Además, *Hill climbing* alcanza los mejores resultados al momento de reasignar pues permite encontrar un óptimo local del problema a partir de la solución ya existente. Por otro lado, cuando se requiere realizar una asignación completa los algoritmos genéticos se desempeñan mejor que *Hill climbing* y *Simulated annealing* pero ninguno de los 3 es capaz de superar la solución del experto. Para representar las soluciones utilizaron una estructura tipo vector en donde cada fila representa la oficina asignada a una entidad.

Al año siguiente [8] los autores proponen una metaheurística híbrida basada en los algoritmos anteriores que toma una solución factible y mejora su calidad de forma iterativa mediante la aplicación de una heurística en cada paso. La metaheurística provee resultados superiores al de las implementaciones estándar de *Hill climbing*, *Simmulated annealing* y algoritmos genéticos por separado.

Por otro lado, [9] Landa-Silva y Burke [9] desarrollaron un enfoque *cooperativo asíncrono de búsqueda local* en los cuales un conjunto de threads locales ejecutan una heurística y cooperan entre si mediante la comunicación de soluciones prometedoras. Por otro lado, se utilizo una heurística de *Tabu search* la cual fue implementada como una matriz nxm, con n la cantidad de entidades y m la cantidad de habitaciones, en donde los movimientos que conllevan a un deterioro de la calidad de la solución son marcados en la matriz para así evitar volver a visitarlos en el futuro. Para la búsqueda de una solución factible inicial se utilizaron las operaciones *swap* (canje de localizaciones), *interchange*

(intercambio de las asignaciones entre entidades) y *relocate* (relocalizar una entidad).

Posteriormente el problema es extendido por Lopes y Girimonte [10] para el caso de organizaciones altamente jerarquizadas. Esta variante fue desarrolladas para la Agencia Espacial Europea y agrega condiciones adicionales como evitar zonas compartidas entre diferentes departamentos y evitar separar equipos. Los autores en su implementación modificaron los operadores de búsqueda local. Por otra parte, Pereira et al. [5] extendió el objetivo agregando minimizar la distancia entre los trabajadores de una misma organización a modo de aumentar su sinergia y maximizar las habitaciones vacantes.

En 2010 Ülker y Landa-Silva [1] proponen un modelo de programación binaria para *OSAP*, las instancias fueron resueltas usando métodos exactos mediante el solver *CPLEX* logrando una diferencia o *gap* de 0.0% con la solución óptima cuando las restricciones de asignar entidades a una misma habitación son duras. Este modelo desplazo a la entonces mejor literatura [9] al entregar soluciones de calidad superior. Los autores notaron que el tiempo de computo para resolver una instancia incrementa de forma desmesurada a medida que se añaden restricciones blandas resultando en tiempos de resolución poco razonables mientras que ajustar todas las restricciones como duras hace el problema intratable, la restricción mas difícil de satisfacer fue la de entidades en una misma habitación del tipo blanda. Mas tarde el modelo es mejorado [11] añadiendo las restricciones de no asignación, capacidad máxima de una habitación y no asignación de dos entidades a una misma habitación.

En [12] Ülker y Landa-Silva proponen un método de resolución basada en algoritmos meméticos. En este método los individuos se representan como arreglos de tamaño fijo con largo igual a la cantidad de entidades. Por otro lado el algoritmo ocupa 3 operadores: *one-point crossover* el cual genera un cruce entre dos individuos y genera dos descendientes los cuales heredan algunas de las características de sus progenitores, *random mutation* el cual aplica una mutación a las características provocando que una entidad sea asignada a otra habitación y uno de *repairing mutation* el cual selecciona aleatoriamente restricciones de asignación especifica que estén siendo violadas y las fuerza a satisfacerse. Este método provee soluciones de mejor calidad en la mitad de los casos respecto a Ülker y Landa-Silva en [1].

Posteriormente Castillo et al. [4] proponen una combinación de algoritmos *greedy* y *Tabu search*. El enfoque propuesto tiene dos fases, una fase de construcción en donde se utiliza el algoritmo *greedy* para encontrar una solución factible y una fase de reparación en la cual se mejora la calidad de la solución. La fase de reparación utilizado dos operadores: *interchange* que intercambia dos entidades y *relocate* que cambia la habitación asignada a una entidad. En la fase de reparación se analiza una lista *Tabu* para seleccionar el mejor movimiento. Este método de resolución permitió encontrar soluciones con un *gap* de 0% en instancias donde [1] y [12] no lograron la solución óptima lo cual indica que visito regiones del espacio de búsqueda los otros algoritmos no.

Además de ello los tiempos de computación resultaron ser son mucho menores a los de la mejor literatura [12] en ese entonces.

En [6] Bolaji et. al plantean una metaheurística basada en una variación de *Hill Climbing* llamada *Late acceptance Hill climbing* que mejora la solución de forma iterativa comparando en cada paso la nueva solución con las soluciones generadas en las soluciones generadas anteriormente. Una nueva solución es aceptada si y solo si su calidad es superior a las anteriores. Este método logra los mejores resultados en las instancias *Nott1*, *Nott1E* y *Wolver1* y soluciones competitivas con otros métodos del estado del arte.

El modelo con mejor desempeño es la variante binaria planteada por Ülker y Landa-Silva[1]. Del estado del arte uno de los algoritmos mas competitivos es el planteado por Bolaji et al. [6] el cual logra 2 mejores resultados, descubre 1 nuevo resultado y presenta soluciones competitivas para el resto de instancias. En tanto el enfoque de Castillo et al. [4] permite generar soluciones competitivas para algunas instancias en un tiempo muy reducido.

Por otro lado, debido a que el problema pertenece a la clase de complejidad NP-Hard, lo cual implica que encontrar la solución puede ser demasiado costoso para algunas instancias, entran en juego las heurísticas y metaheurísticas pues estas pueden ayudar a reducir el espacio o a encontrar soluciones que a pesar de no ser óptimas poseen suficiente calidad.

4 Modelo matemático

Para el modelamiento matemático seguimos el enfoque de Castillo et al. [4]

- **Constantes**:
  R: conjunto de habitaciones
  E: conjunto de entidades
  S_e: tamaño de la entidad e, $\forall\ e \in E$
  C_r: capacidad de la habitación r, $\forall\ r \in R$
  A_r: lista de habitaciones adyacentes a r, $\forall\ r \in R$
  N_r: lista de habitaciones cercanas a r, $\forall\ r \in R$
  J: conjunto de restricciones $J = \{al,\ na,\ sr,\ nsr,\ nsh,\ ad,\ gr\ ,aw,\ cp\}$
  HC^j: conjunto de restricciones duras del tipo j, $\forall\ j \in J$
  SC^j: conjunto de restricciones suaves del tipo j, $\forall\ j \in J$

- **Variables**:

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

$$y_i^j = \begin{cases} 1, \text{si la restricción } i \text{ tipo } j \text{ es violada } \forall i \in \mathbf{card}(SC^j)\ \forall j \in J \\ 0, \text{en otro caso} \end{cases}$$

- **Función objetivo**:

$Min \quad z \quad = \quad \sum_{r \in R} max(C_r \; - \; \sum_{e \in E} x_{er} S_e, \quad 2 \sum_{e \in E} x_{er} S_e \; - \; C_r) \quad + \sum_{j \in J} w^j \sum_{i=1}^{\mathbf{card}(SC^j)} y_i^j$

El primer termino se utiliza para minimizar el sobre uso o infrauso de una habitación, ambos son mutuamente excluyentes. Sobre uso se penaliza el doble. El segundo termino es utilizado para minimizar la cantidad de restricciones suaves violadas donde el termino w^j es el peso asociado a quebrar una restricción de tipo j.

- **Modelamiento de restricciones duras**:

1. Todas asignadas: todas las entidades son asignadas a una habitación

   $\sum_{r \in R} x_{er} = 1 \; \forall e \in E$

2. Asignación: la entidad e debe ser asignada (específicamente) a la habitación r

   $x_{er} = 1$

3. No asignación: la entidad e no debe ser asignada a la habitación r

   $x_{er} = 0$

4. No compartimiento: la entidad e no debe compartir su habitación con otros

   $\sum_{f \in E-e} x_{fr} \leq (\mathbf{card}(E) - 1) - (\mathbf{card}(E) - 1) x_{er} \; \forall r \in R$

5. Misma habitación: las entidades e_1 y e_2 deben ser asignadas a la misma habitación

   $x_{e_1 r} - x_{e_2 r} = 0 \; \forall r \in R$

6. Habitación distinta: las entidades e_1 y e_2 deben ser asignadas en la misma habitación r

   $x_{e_1 r} + x_{e_2 r} \leq 1 \; \forall r \in R$

7. Adyacencia: las entidades e_1 y e_2 deben ser asignadas en habitaciones adyacentes

   $x_{e_1 r} \leq \; \sum_{s \in A_r} x_{e_2 s} \leq 1 \; \forall r \in R$

8. Cercanía: las entidades e_1 y e_2 que deben ser asignadas en habitaciones cercanas

   $x_{e_1 r} \leq \sum_{s \in N_r} x_{e_2 s} \leq 1 \; \forall r \in R$

9. Lejanía: las entidades e_1 y e_2 que deben ser asignadas en habitaciones lejanas

   $0 \leq \sum_{s \in N_r} x_{e_2 s} \leq 1 - x_{e_1 r} \; \forall r \in R$

10. Capacidad: la habitación r no puede ser usada por sobre su capacidad

    $\sum_{e \in E} S_e x_{er} \leq C_r$

- **Modelamiento de restricciones blandas**: Las variables auxiliares tomaran un valor 1 si la restricción no es satisfecha

1. Asignación:

   $y_i^{al} = 1 - x_{er}$

2. No asignación: la entidad e no debe ser asignada a la habitación r

   $y_{na} = x_{er}$

3. No compartimiento:

   $(\mathbf{card}(E) - 1)(2 - x_{er} - y_{ir}^{nsh}) \geq \sum_{f \in E-e} x_{fr} \; \forall r \in R$

   $\sum_{f \in E-e} x_{fr} \geq (\mathbf{card}(E) - 1)(1 - x_{er}) + \epsilon - (\mathbf{card}(E) - 1 + \epsilon) y_{ir}^{nsh} \; \forall r \in R$

   $y_i^{nsh} = \sum_{r \in R} (y_{ir}^{nsh})$

4. Misma habitación:

   $2y_{ir}^{sr} - 1 \leq x_{e_1 r} - x_{e_2 r} \leq 1 - \epsilon + \epsilon y_{ir}^{sr} \; \forall r \in R$

   $y_i^{sr} = \sum_{r \in R} (1 - y_{ir}^{sr})$

5. Habitación distinta:

   $(1 + \epsilon) - (1 + \epsilon) y_{ir}^{nsr} \leq x_{e_1 r} - x_{e_2 r} \leq 2 - y_{ir}^{nsr}$

   $y_i^{nsr} = \sum_{r \in R} (1 - y_{ir}^{nsr})$

6. Adyacencia:

   $y_{ir}^{ad} + x_{e_1 r} - 1 \leq \sum_{s \in A_r} x_{e_2 s} \leq x_{e_1 r} - \epsilon + (1+\epsilon) y_{ir}^{ad} \ \forall r \in R$

   $y_i^{ad} = \sum_{r \in R} (1 - y_{ir}^{ad})$

7. Cercanía:

   $y_{ir}^{gr} + x_{er} - 1 \leq \sum_{s \in N_r} x_{fs} \leq x_{er} - \epsilon + (1+\epsilon) y_{ir}^{gr} \ \forall r \in R$

   $y_i^{gr} = \sum_{r \in R} (1 - y_{ir}^{gr})$

8. Lejanía:

   $1 - x_{e_1 r} + \epsilon - (1+\epsilon) y_{ir}^{aw} \leq \sum_{s \in N_r} x_{e_2 s} \leq 2 - x_{e_1 r} - yir^{aw} \ \forall r \in R$

   $y_i^{aw} = \sum_{r \in R} (1 - y_{ir}^{aw})$

9. Capacidad:

   $\sum_{e \in E} S_e x_{er} + (C_r + \epsilon)(1 - y_i^{cp}) \geq C_r + \epsilon$

   $\sum_{e \in E} S_e x_{er} + (\sum_{e \in E} S_e - C_r)(1 - y_i^{cp}) \leq \sum_{e \in E} S_e$

5 Representación

Para la representación de soluciones candidatas del problema seguimos el enfoque de Burke et al. [2]. Una solución sera representada como un vector v de enteros de tamaño $n = |E|$ con E el conjunto de entidades. La entrada i del vector v corresponderá al recurso (habitación) asignado a la entidad con identificador i, esto es, v_i = recurso asignado a la entidad i con $v_i \in R$. La estructura que representa la asignación se muestra en la Figura 1.

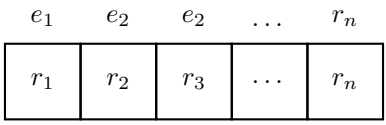

Fig. 1: Vector representación de solución candidata

6 Descripción del algoritmo

Debido a que el problema pertenece a la clase de complejidad NP-Hard encontrar la solución utilizando técnicas de resolución completa resulta demasiado

costoso, en consecuencia usaremos técnicas de búsqueda incompleta. El proceso de búsqueda de soluciones se divide en dos fases, una de construcción y una de reparación. En la fase de construcción se utiliza un algoritmo de búsqueda heurística informada $greedy\,best-first\,search$ para construir una solución inicial factible. Esta clase de algoritmos utilizan conocimiento del dominio para ir seleccionando localmente los valores que parecen estar mas cercanos a la meta de la función de evaluación. Luego, en la fase de reparación utilizamos un algoritmo reparador $Hill-climbing\,MM$ (mejor mejora).

En el caso del algoritmo *greedy* la función de evaluación corresponderá a la función objetivo, es decir,

$$Min \quad z \quad = \quad \sum_{r\in R} max(C_r \; - \; \sum_{e\in E} x_{er}S_e, \quad 2\sum_{e\in E} x_{er}S_e \; - \; C_r) \quad + \sum_{j\in J} w^j \sum_{i=1}^{\mathbf{card}(SC^j)} y_i^j$$

En la solución parcial se asignaran primero aquellas entidades que deben asignarse a una habitación especifica, luego de manera secuencial se asignara a las entidades restantes a las habitaciones remanentes de manera tal que cada entidad se asigne a la mejor habitación posible, es decir, aquella habitación en la que el costo asociado de asignación en la función objetivo sea mínimo satisfaciendo todas las restricciones. El costo que produce la asignación en la función de evaluación esta asociado al sobre uso o infrauso de una habitación y la insatisfacción de restricciones blandas. En consecuencia la función heurística es

$h(i)$ = asignar entidad i a la habitación disponible que produzca el mínimo aporte a la función de evaluación.

Algorithm 1 Algoritmo greedy OSAP

```
s ← vector solución parcial
while entidades sin asignar do
    e ← e_i ∈ E sin asignar
    r ← r_j ∈ R tal que la penalización es mínima
    s_e = r
end while
```

7 Conclusiones

De las técnicas estudiadas no todas están enfocadas a exactamente el mismo problema, en el modelo de programación lineal mixta planteado por Ritzman et al. [3] la función objetivo a minimizar difiere de la forma estándar que se planteo en el modelo e incluye términos como distancias desde oficinas a departamentos. Esto se debe en sus orígenes el problema no estaba generalizado

para ser aplicado en otras organizaciones por lo cual los objetivos respondían mas un problema de organización académica, posteriormente el problema fue generalizado dando como resultado el modelo binario de Ülker y Landa-Silva [1].

Debido a que el problema pertenece a la clase de complejidad NP-Hard [6] encontrar soluciones es un proceso costoso por lo cual las técnicas de resolución están enfocadas al uso de heurísticas y meta heurísticas que permitan acelerar el proceso de búsqueda. Usualmente el proceso de búsqueda se divide en dos etapas, una etapa de construcción y una etapa de reparación [4]. En la etapa de construcción se utiliza un procedimiento para encontrar una solución factible, para ello se realiza una búsqueda local la cual puede ser realizada mediante una serie de operaciones, como *reallocate*, que alteran las porciones de peor calidad, otra forma de construir la solución inicial es mediante algoritmos evolutivos que realicen mutaciones aleatorias. Posteriormente en la etapa de reparación se utilizan heurísticas para mejorar iterativamente la calidad de la solución. Las técnicas generalmente difieren en los algoritmos que se utilizan en las fases.

El enfoque utilizado corresponderá a uno de dos fases utilizando técnicas de búsqueda incompleta lo cual permite utilizar heurísticas que aprovechan el dominio del problema para generar una solución inicial factible mediante un algoritmo *greedy* para luego mejorarla iterativamente mediante un algoritmo de reparación $Hill - Climbing$. Esto permite reducir el tamaño del espacio de búsqueda y reducir la complejidad asociada que tiene debido a ser un problema NP-Hard. Por otro lado, debido la cantidad de restricciones que tienen las instancias al momento de asignar una entidad a una habitación en la solución parcial existe un costo elevado asociada a revisar la factibilidad de la asignación. Estas dificultades hacen que el problema sea difícil de tratar incluso mediante el uso de heurísticas por lo cual la resolución de instancias reales mediante el uso de técnicas completas no es factible.

La desventaja del uso de heurísticas es que estas pueden nunca converger a la solución óptima o incluso llegar a ella en el mismo tiempo que lo haría un algoritmo *greedy*. Una investigación a futuro podría considerar los efectos que sufren las distintas heurísticas al relajar algunas restricciones del problema y como esto afecta la factibilidad y calidad de la solución.

Para la siguiente entrega se debe finalizar la implementación del algoritmo *greedy* y empezar a desarrollar la fase de reparación mediante $Hill - climbing$.

References

[1] Ülker, Ö., Landa-Silva, D.: A 0/1 integer programming model for the office space allocation problem. Electronic Notes in Discrete Mathematics **36**, 575–582 (2010)

[2] Burke, E.K., Cowling, P., Landa Silva, J.D., McCollum, B.: Three methods to automate the space allocation process in uk universities. Proceedings of the 3rd International Conference on the Practice and Theory of Automated Timetabling, 374–393 (2000)

[3] Ritzman, L., *et al.*: Multiple objective approach to space planning for academic facilities. Management Science **25**(9), 895–906 (1979)

[4] Castillo, F., Riff, M.C., Montero, E.: New bounds for office space allocation using tabu search. Proceedings of the 2016 on Genetic and Evolutionary Computation Conference - GECCO '16 (2016)

[5] Pereira, R., Cummiskey, K., Kincaid, R.: Office space allocation optimization. 2010 IEEE Systems and Information Engineering Design Symposium (2010)

[6] Bolaji, A.L., Michael, I., Shola, P.: Adaptation of late acceptance hill climbing algorithm for optimizing the office-space allocation problem. International Workshop on Hybrid Metaheuristics, 180–190 (2019)

[7] Lopes, R., Girimonte, D.: The office space allocation problem in strongly hierarchized organizations. Evolutionary Computation in Combinatorial Optimization **6022**, 143–153 (2010)

[8] Burke, E.K., Cowling, P., Landa Silva, J.D.: Hybrid population-based metaheuristic approaches for the space allocation problem. Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546), 232–239 (2001)

[9] Landa-Silva, D., Burke, E.K.: Asynchronous cooperative local search for the office-space-allocation problem. INFORMS Journal on Computing **19**(4), 575–587 (2007)

[10] Lopes, R., Girimonte, D.: The office space allocation problem in strongly hierarchized organizations. Evolutionary Computation in Combinatorial Optimization **6022**, 143–153 (2010)

[11] Landa-Silva, D., Ülker, Ö.: Designing difficult office space allocation problem instances with mathematical programming. Proceedings of the 10th international conference on Experimental algorithms, 280–291 (2011)

[12] Ülker, Ö., Landa-Silva, D.: Evolutionary local search for solving the office space allocation problem. EEE Congress on Evolutionary Computation(CEC 2012), 1–8 (2012)