UNIVERSIDAD DE CONCEPCIÓN - CHILE
FACULTAD DE INGENIERIA
DEPARTAMENTO DE INGENIERÍA INDUSTRIAL

HEURÍSTICA BASADA EN ENTROPÍA PARA RESOLVER EL PROBLEMA DE ASIGNACIÓN NO BALANCEADO

por
MARCELO IGNACIO GALLEGOS ORTIZ

Profesora Guía:
Dott.ssa Rosa Medina D.

Concepción, abril de 2021

Tesis presentada a la

DIRECCIÓN DE POSTGRADO
UNIVERSIDAD DE CONCEPCIÓN

Para optar al Grado de
Magíster en Ingeniería Industrial

Tabla de Contenidos

Índice de Tablas

Índice de Figuras

Capítulo 1 : Introducción

La necesidad de mantener un nivel competitivo dentro del actual mercado, incrementalmente exigente en la calidad de los productos y servicios demandados, empuja a las industrias y empresas a refinar constantemente las herramientas y métodos implementados en su gestión de operaciones y administración de recursos. Las organizaciones industriales, históricamente, han abordado el desafío de cómo responder a las crecientes y cambiantes demandas del mercado, analizando e implementando técnicas y prácticas que le permitan dar solución a sus desafíos operativos y perfeccionar sus sistemas productivos. Algunos de los más frecuentes problemas abordados por las empresas se centran en la administración y gestión de activos, materiales y productos.

El campo de estudio de los problemas de asignación (*Assignment Problem*, AP) permite conocer cuál es la distribución óptima de diferentes tareas, trabajos o actividades que requieren ser ejecutadas en un sistema, entre los diferentes agentes, operadores o máquinas disponibles para su asignación, de manera tal que se minimicen los costos monetarios o temporales asociados a dicha distribución. Toma especial interés en este tipo de problemas no sólo buscar como solución un programa de asignación que entregue la mayor eficiencia en términos de recursos consumidos (sean monetarios o temporales), sino también que la propuesta construida sea robusta y de interés de implementación práctica para abordar problemas reales.

Generalmente, se reconoce como el inicio del desarrollo de métodos exactos para la resolución del problema de asignación clásico (y sus variantes) a la publicación en 1955 del artículo referido al Método Húngaro presentado por Kuhn (Kuhn, 1955). Más de 60 años después, modificaciones de la solución inicial han disminuido la complejidad temporal original (Edmonds y Karp, 1972), así como han extendido su aplicación a problemas de flujo en redes (Ford y Fulkerson, 1956). A su vez, la aplicación práctica de variantes de estos métodos ha permitido abarcar una extensa cantidad de problemas de aplicación prácticos. La distribución de vendedores dentro de diferentes zonas regionales, la asignación de trabajos a máquinas en un sistema productivo y la asignación de aviones a las mangas de un aeropuerto son ejemplos de aplicación del campo de estudio de los problemas de asignación (Rabbani et al., 2019; Sena Das et al., 2020).

En términos generales, y en referencia a las dimensiones del AP, podemos encontrar una clasificación del problema basada en dos enfoques. El primer enfoque, y denominado Problema de Asignación Clásico o Balanceado (*Classical Assignment Problem*, CAP), supone que existe igual número de tareas y número de agentes. De esta forma, se procede a asignar en una correspondencia de uno a uno cada tarea a los agentes disponibles, hasta completar la distribución. El segundo enfoque, denominado Problema de Asignación No Balanceado (*Unbalanced Assignment Problem*, UAP) tiene la capacidad de abordar situaciones en donde no existe una equivalencia entre tareas y

agentes, pudiendo abarcar tanto un mayor número de tareas que de agentes, así como viceversa (Bhunia et al., 2017). Se tiene, por tanto, que el UAP corresponde a un problema más general que el CAP. Esto es, un algoritmo capaz de resolver el UAP es también capaz de resolver instancias de CAP.

En el presente proyecto se desarrolla un algoritmo heurístico para resolver el UAP, bajo un enfoque basado en el concepto de entropía. Entenderemos la entropía de una determinada tarea como el desorden potencial, en términos de costo, asociado a las combinaciones de asignación de la misma a los agentes disponibles. De esta forma, se establece un criterio de priorización entre tareas para un procedimiento de asignación secuencial de las mismas. El objetivo del algoritmo, finalmente, es minimizar los costos de asignación de tareas a un conjunto de agentes que conforman un sistema.

1.1. Hipótesis

Si se desarrolla un Algoritmo Heurístico, basado en el impacto de las diferencias de costos de asignación de cada tarea, entonces es posible resolver el Problema de Asignación No Balanceado con una solución que contemple una distribución equitativa de tareas entre agentes.

1.2. Objetivo general

Desarrollar un algoritmo heurístico basado en Entropía para resolver el Problema de Asignación No Balanceado (UAP).

1.3. Objetivos específicos

1. Caracterizar el problema de asignación, en entornos Balanceado y No Balanceado (UAP).
2. Analizar las características, condiciones y restricciones presentadas en la literatura para abordar el UAP.
3. Diseñar e implementar un algoritmo basado en el concepto de entropía para resolver el UAP.
4. Evaluar la efectividad y el desempeño del algoritmo desarrollado en un conjunto de casos de estudio extraídos desde la literatura, comparando sus resultados con los alcanzados en publicaciones anteriores.

El siguiente trabajo está dividido en capítulos en una secuencia lógica que, a medida que se desarrolla, permite entender la relevancia del UAP dentro de la literatura relacionada y trabajos realizados, presentar el diseño y desarrollo del algoritmo propuesto para resolver el problema bajo un enfoque basado en el concepto de Entropía, y establecer un conjunto de casos de estudio para los cuales será aplicado para realizar un análisis y proporcionar conclusiones sobre el método desarrollado.

En detalle, en el Capítulo 2 se exponen antecedentes teóricos sobre el problema de asignación, realizando una revisión de la literatura relacionada con la variante no balanceada y los diferentes métodos existentes para su resolución. Además, se discute sobre las consideraciones o definiciones presentes en trabajos anteriores respecto a las restricciones o limitaciones del número de tareas a asignar a un mismo agente.

En el Capítulo 3 se presenta la formulación del algoritmo propuesto para la resolución del UAP, definiendo el concepto de entropía en el cual se basa el algoritmo, la construcción de la estructura de la solución a partir del cálculo de la Capacidad de Asignación y la metodología de aplicación del algoritmo.

El Capítulo 4 presenta los casos de estudio utilizados para la experimentación, así como la recopilación y análisis de los resultados obtenidos. Se presenta una comparativa y discusión del desempeño del algoritmo respecto soluciones alcanzadas por otros métodos presentes en la literatura, así como un análisis estadístico sobre la calidad de las soluciones y tiempos de ejecución del algoritmo propuesto.

Finalmente, en el Capítulo 5 se presentan las principales conclusiones de este trabajo, además de algunas propuestas de interés para guiar futuras investigaciones.

Capítulo 2 : Revisión bibliográfica

En este capítulo se presenta una revisión bibliográfica sobre el Problema de Asignación No Balanceado, sus características, sus principales variantes y metodologías para su resolución. Además, se presenta una discusión sobre la restricción de capacidad de asignación de tareas a un agente.

2.1. Sobre el problema de asignación no balanceado

El problema de asignación es uno de los más destacados y frecuentemente abordados tópicos dentro de las ciencias de administración y gestión industrial, siendo un problema significativamente relevante en el campo de las matemáticas por su capacidad de modelar situaciones de interés real. El problema de asignación (*Assignment Problem*, AP) es un problema de optimización lineal, y corresponde a un caso particular de los problemas de transporte (Ahmed y Ahmad, 2014). En su forma más general, puede ser descrito como el problema para asignar de forma óptima ciertos recursos disponibles, denominados agentes, para la ejecución de determinadas tareas, incurriendo en un costo asociado que puede variar según la combinación agente-tarea. Este modelo se puede aplicar a todo campo que requiera generar programas de asignación, tales como trabajos a máquinas, personal a proyectos, contratos a oferentes, vendedores a territorios, vehículos a rutas o productos a fábricas son ejemplos de aplicación de este tipo de problemas (Kabiru et al., 2017). Es, por tanto, un factor esencial dentro de las actividades de la planificación de la producción, gestión de recursos y gestión de activos.

En el AP se contempla la posibilidad de que cualquier agente puede ser asignado para desarrollar cualquier tarea, considerando que esta asignación contrae un costo que puede variar dependiendo de la combinación agente-tarea generada. Una restricción importante en un AP es que cada tarea puede ser asignada a un solo agente (Pentico, 2007). Por último, se tiene la necesidad de desarrollar todas las tareas, de forma tal que el costo total de las asignaciones sea el mínimo.

El Problema de Asignación considera múltiples tipos dentro de su clasificación, dependiendo del tipo de problema a abordar, teniendo como una de sus variantes al Problema de Asignación Lineal (Burkard et al., 2009). La principal característica de esta variante es que contempla un emparejamiento entre agentes y tareas en el que se asume que una tarea puede ser asignada sólo a un agente, y cada agente puede tener asignada sólo una tarea. El objetivo del problema es minimizar (o maximizar) la sumatoria de los costos de asignación (o utilidades) asociados al emparejamiento de cada tarea a agente. En el caso de minimización, los costos de asignación se consolidan en una matriz de costos de asignación. A su vez, dentro de esta categoría se pueden considerar casos en los que exista igual o diferente número de tareas y agentes, considerando al AP

como balanceado o no balanceado, respectivamente (Wang et al., 2015). El AP Balanceado describe a un problema de asignación equilibrado. Esto es, existe un número idéntico de agentes y tareas y, por tanto, las tareas son asignadas a los agentes en una correspondencia uno-a-uno (1-1), con un costo de asignación fijo (Munapo, 2020).

En Votaw y Orden (1952) se introduce formalmente el AP, y se define como objetivo del problema el establecer una asignación uno-a-uno entre tareas y agentes, así como minimizar el costo monetario o temporal de dicha asignación. Un extenso número de métodos se han desarrollado o implementado para resolver el AP en su versión balanceada, algunos de los cuales corresponden al Método Húngaro (Kuhn, 1955), Algoritmo Genético (Chu y Beasley, 1997), y Recocido Simulado (Sahu y Tapadar, 2006).

En contraste al enfoque de asignación uno-a-uno, el enfoque de asignación uno-a-muchos (1-M) contempla la posibilidad de que un mismo agente ejecute más de un trabajo. Este tipo de asignación captura un mayor potencial de aplicación a situaciones reales, especialmente en el modelamiento de sistemas con recursos escasos y/o de baja disponibilidad (Bhunia et al., 2017). En Thenepalle y Singamsetty (2019) se señala que en muchos casos prácticos existe una dificultad latente para describir una relación balanceada entre el número de agentes y el número de tareas. Así, la mayoría de los casos de estudio pueden ser abordados como un problema de asignación no balanceado (*Unbalanced Assignment Problem*, UAP).

De acuerdo a Biswas et al. (2018), esta rama de los problemas de asignación se puede categorizar en dos tipos: Tipo 1, si el número de tareas excede al número de agentes disponibles; y Tipo 2, si el número de agentes es mayor que el de tareas. En particular, los UAP de Tipo 2 tienen soluciones que demandan que algunos agentes queden libres u ociosos. En contraste, los UAP de Tipo 1 tienen soluciones en las que ciertas tareas quedan sin realizarse o, alternativamente, se extienden a la interesante decisión de qué agentes realizarán más de una tarea.

De forma ilustrativa, en la Figura 1 se tiene una representación gráfica del Problema de Asignación No Balanceado. En la parte izquierda de la figura, se ilustra el UAP con m tareas y n agentes ($n < m$), y en donde C_{ij} denota el costo de asignación de la tarea i al agente j. En la parte derecha de la figura, se ilustra una solución para el UAP, en donde cada asignación comprende su costo asociado. Además, se asume que las tareas son ejecutadas de forma independiente, y no existe ninguna dependencia lógica entre ellas.

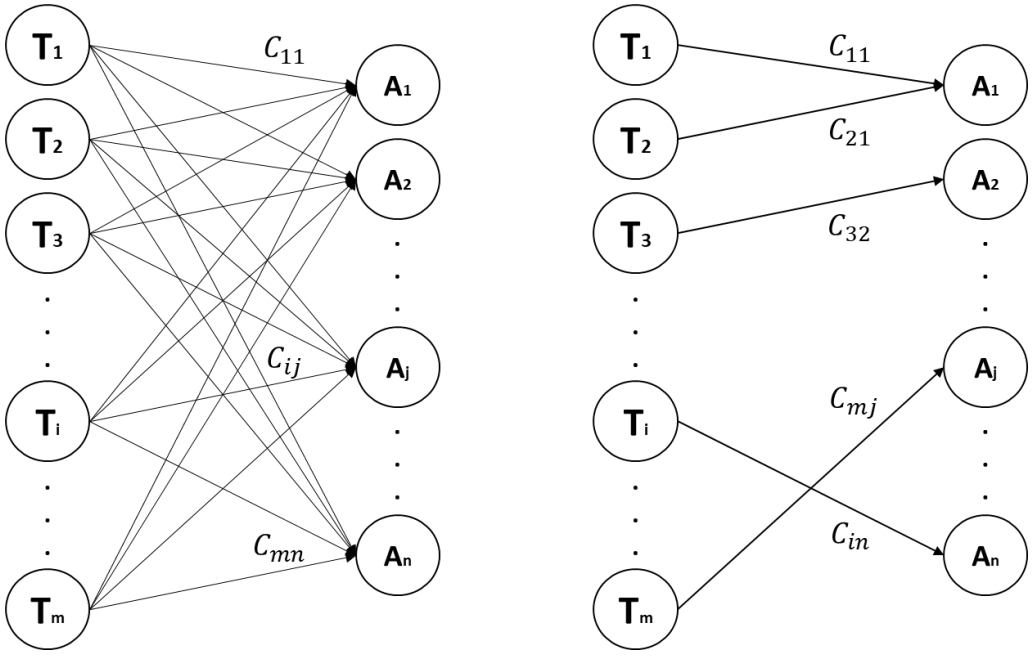

Figura 2.1. Representación del Problema de Asignación No Balanceado
Fuente: Elaboración propia.

La estructura de estos modelos, en efecto, representa gran parte de los escenarios de aplicación práctica dentro de los problemas de asignación. Así, por ejemplo, se puede analizar un caso en donde un operador (agente) se dispone a ejecutar una serie de trabajos (tareas), las cuales no puede ejecutar de forma simultánea, sino los procede a ejecutar sucesivamente siguiendo una secuencia ordenada. La acción de determinar las tareas a ejecutar por el operador, así como su secuencia, estará condicionada por la búsqueda de una reducción en costos de ejecución, así como en el incremento de la productividad.

El creciente desarrollo de los métodos de producción y las técnicas de gestión de producción reconocen el papel esencial de abordar modelos de asignación 1-M. En la literatura se tienen algunos campos de aplicación variados de este tipo de modelos, abarcando la asignación de pacientes a médicos disponibles en la gestión de desastres climáticos (Amaliah et al., 2013), la distribución de carga de procesamiento en computación en la nube (Mondal et al., 2019) y casos generales en Administración y Manufactura (Ramesh et al., 2020).

2.2. Sobre las técnicas de resolución del UAP

El UAP puede ser modelado como un Problema de Flujo de Costo Mínimo (*Minimum Cost Flow Problem*, MCFP). Esto es, considerar una representación del UAP a través de una red, en donde se tienen un número de $m + n$ nodos (sumatoria de m Tareas y n Agentes asociados al problema), y un número de mn arcos (posibles combinaciones entre la asignación de Tareas y Agentes). Luego, por tanto, el UAP puede definirse como un problema perteneciente a la clase de complejidad P.

Múltiples algoritmos han sido propuestos para resolver el MCFP, con diferencias de interés en la complejidad temporal asociada a sus implementaciones. Así, se tiene el algoritmo *Cycle-Canceling* (Sokkalingam et al., 2000), el algoritmo *Successive Shortest Path* (Engquist, 1982; Brunsch et al., 2015), el algoritmo Primal-Dual (Williamson, 2002), y el algoritmo *Out-of-Kilter* (Ciupala, 2005). En la Tabla 2.1 se detalla la complejidad temporal asociada a cada implementación al contemplar la nomenclatura asociada al UAP, y considerando un problema de m Tareas y n Agentes.

Tabla 2.1. Algoritmos para la resolución del MCFP y su complejidad temporal
Fuente. Elaboración propia.

Algoritmo	Complejidad Temporal
Algoritmo *Cycle-Canceling*	$\mathcal{O}(mn * (mn + (m+n)\log(mn)) * \log(m+n))$
Algoritmo *Successive Shortest Path*	$\mathcal{O}(mn(m+n) * (mn + (m+n)\log(m+n))\log(m+n))$
Algoritmo Primal-Dual	$\mathcal{O}((m+n)^2 * \log(m+n))$
Algoritmo *Out-of-Kilter*	$\mathcal{O}(mn * (mn + (m+n)\log(m+n)))$

Para el problema de asignación, múltiples enfoques que han sido desarrollados para encontrar una política de asignación óptima de tareas a agentes. El método más popular disponible y extendido en la literatura para obtener una solución óptima en la distribución de tareas para el Problema de Asignación Balanceado es el Método Húngaro, el cual cuenta con una complejidad temporal de $\mathcal{O}(n^3)$, donde n corresponde al número de tareas (Wu et al., 2019). Fue publicado en 1955 por Harold Kuhn (Kuhn, 1955), quien entregó el nombre al método en reconocimiento a los trabajos realizados por los matemáticos húngaros Dénes König y Jenö Egerváry, y en los cuales se basa el algoritmo desarrollado. Al abordar el UAP, existen variantes del método que consideran la asignación del excedente de tareas a agentes ficticios. Estas tareas son, finalmente, ignoradas en su ejecución. Adicionalmente, añadir nuevas columnas o filas para representar estos elementos ficticios en la matriz de costos de asignación puede aumentar significativamente el tamaño, considerando la complejidad espacial $\mathcal{O}(n^2)$ del problema (Iampang et al., 2010). En Kumar et al. (2018) se plantea

que abordar el UAP haciendo uso del Método Húngaro repercute considerablemente en el consumo de memoria y tiempo computacional invertido, especialmente cuando se incrementa el tamaño del problema (esto es, una matriz de costos con mayor número de elementos). Por último, en Rabbani et al. (2019) se describe que la falta de aplicación práctica de variantes del Método Húngaro para el UAP, al dejar tareas sin ejecución, lo vuelven un método poco atractivo para abordar escenarios reales.

Algunos autores han presentado algoritmos basados en el Método Húngaro para resolver el UAP, de manera tal que la solución óptima contemple la totalidad de tareas a ser distribuidas y ejecutadas por los agentes disponibles. En Kumar (2006) se propone una variante modificada del Método Húngaro para el UAP, y cuyo algoritmo permite la decisión de asignar más de una tarea a un mismo agente. En Betts y Vasko (2016) se propone una reformulación teórica del UAP para balancear el problema, y luego resolverlo aplicando el Método Húngaro. El algoritmo propuesto se aplica al problema presentado por Kumar (2006), obteniendo una solución con un menor costo total de asignación. En Rabbani et al. (2019) se presenta una nueva modificación al Método Húngaro, y cuyo algoritmo desarrollado resuelve el UAP, incluso entregando una solución con un costo total de asignación inferior al alcanzado por la propuesta por Betts y Vasko (2016).

Otros autores se han inclinado por implementar metodologías y algoritmos heurísticos alternativos al Método Húngaro. Heurística puede ser definida como un enfoque de resolución de problemas, bajo una estrategia que examina un número limitado de alternativas, para encontrar soluciones de buena calidad en tiempos computacionales razonables (Burkard et al., 2009). En términos generales, pueden encontrarse dentro de los tipos de heurísticas los métodos constructivos (procedimientos que construyen la solución al problema paso a paso), métodos de mejora (procedimientos que toman una solución inicial y se enfocan en su refinamiento), y metaheurísticas (Silver, 2004). Iampang et al. (2010) proponen un método heurístico constructivo para resolver el UAP. El algoritmo propuesto se enfoca en minimizar los costos de asignación de m tareas a los n agentes disponibles, así como establecer una complejidad espacial $\mathcal{O}(nm)$ (en contraste con la complejidad espacial del Método Húngaro, igual a $\mathcal{O}(m^2)$). En Yadaiah y Haragopal (2016) se presenta un método basado en búsqueda lexicográfica para resolver problemas de asignación no balanceados. El algoritmo propuesto permite dar una solución que contemple todas las tareas involucradas a un UAP. El método propuesto es aplicado al caso de estudio proporcionado por Kumar (2006), alcanzando un valor óptimo igual al propuesto en dicho trabajo.

Se tienen, a su vez, varios algoritmos enfocados en refinar estrategias de búsqueda de soluciones sobre la matriz de costos de asignación. En Fang et al. (2011) se aborda un UAP en donde el número de agentes supera al de tareas a ejecutar. Los autores modelan el problema de forma idéntica a los casos anteriormente presentados, con la salvedad pertinente que todas las tareas deben ser ejecutadas, y que un mismo agente puede ejecutar una o ninguna tarea. Se propone, entonces, un

algoritmo de búsqueda codiciosa, el cual presenta mejores resultados que el enfoque basado en el Método Húngaro. Khandelwal (2018) propone resolver el UAP a través de un nuevo método basado en la búsqueda exhaustiva de soluciones a través de optimización con límites parciales. El interés del autor se basa en abordar situaciones reales en donde típicamente el número de tareas supera al de agentes disponibles. El algoritmo desarrollado se aplica a un ejemplo numérico para mostrar la eficiencia de la técnica propuesta. En Jain et al. (2019) se aborda un UAP en donde las tareas a ejecutar se encuentran distribuidas en dos etapas consecutivas y excluyentes. El objetivo del problema es minimizar el tiempo total de ejecución del proyecto, es decir, la suma de los tiempos de término de la primera y segunda etapas. Los autores presentan un algoritmo iterativo que genera soluciones con una estructura pareada para primera y segunda etapa.

Por último, dentro de la literatura es posible encontrar métodos metaheurísticos o inteligentes para abordar el UAP. El principal interés en la propuesta de estos métodos es el desarrollo de enfoques alternativos, capaces de entregar una solución de buena calidad, haciendo uso de un menor tiempo y espacio computacional. Además, se puede considerar la solución proporcionada por estos métodos como la solución inicial a un *solver* para determinar una solución exacta de forma más eficiente en términos temporales. Majumdar y Bhunia (2012) desarrollan un Algoritmo Genético (*Genetic Algorithm*, GA) elitista para resolver un UAP. Los autores presentan operadores de inicialización, cruzamiento y mutación nuevos de forma tal que el algoritmo propuesto es capaz de asignar de manera óptima la totalidad de tareas a los agentes disponibles. En Bhunia et al. (2017) se propone un enfoque basado en un GA, pero el interés del modelo recae en considerar costos de asignación definidos como intervalos de valores. En el trabajo presentado se desarrollan dos versiones del GA, con diferencias en los operadores de cruzamiento. Por último, en Biswas et al. (2018) se presenta un GA para resolver un UAP multiobjetivo. El modelo propuesto aborda ciertas restricciones en los agentes en ejecutar ciertas tareas, lo que se entiende como la escasa o pobre eficiencia de determinado agente en realizar ciertos procesos específicos.

Sin embargo, y pese a la potencial aplicación de modelos de asignación para abordar problemas de aplicación real, gran parte de los trabajos existentes en la literatura replican y/o modifican técnicas anteriormente presentadas, no proponiendo y desarrollando nuevos enfoques o métodos para abordar el UAP de forma efectiva y robusta.

2.3. Sobre la capacidad de asignación de tareas por agente

Respecto a la distribución de las tareas a asignar para cada agente, y directamente relacionado con la capacidad de modelar y resolver problemas de aplicación real, en la literatura se encuentran algunas referencias respecto al número mínimo y máximo de tareas a asignar a un mismo agente.
En primer lugar, y respecto al número mínimo de tareas a asignar a un agente, los trabajos relacionados parecen enfocarse en evitar que un agente quede ocioso, teniendo en los supuestos del modelo que cada agente debe ejecutar a lo menos una tarea dentro de la asignación generada en la solución (Majumdar y Bhunia, 2012; Bhunia et al., 2017; Betts y Vasko, 2016).

En segundo lugar, y respecto al número máximo de tareas a asignar a un mismo agente, una primera aproximación a esta consideración es realizada por Majumdar y Bhunia (2012). En este trabajo, los autores establecen como restricción del modelo una capacidad máxima de tareas que un mismo agente puede ejecutar. Así, para un problema con m tareas y n agentes, la capacidad límite por agente es igual a $(m - n + 1)$ tareas, esto es, la capacidad límite por agente está definida en función del tamaño del problema. Además, esta restricción se complementa con la restricción de que no quedan agentes ociosos en la asignación, esto es, que cada agente ejecuta al menos una tarea. Este mismo enfoque es abordado como supuesto del modelo presentado en Bhunia et al. (2017), y luego en Biswas et al. (2018). Sin embargo, al definir este límite superior se entrega un amplio, y posiblemente excesivo, intervalo de valores posibles para el número máximo de tareas a asignar a un mismo agente. Por ejemplo, para un problema no balanceado que considere 8 tareas y 5 agentes, se tiene una capacidad máxima de hasta 4 tareas por agente y, de tomarse una solución de esta dimensión, se distribuirían las restantes 4 tareas entre los 4 agentes ociosos. En la Figura 2.2 se representa una comparativa de dos versiones de la estructura de la solución que podrían generarse como resultado para este problema, considerando un máximo de cuatro y dos tareas a asignar a un mismo agente. Se hace evidente, según esto, que la solución de mayor tolerancia (cuatro tareas a un mismo agente) es poco atractiva si se considera la intención de una aplicación práctica de esta solución. Por supuesto, este último comentario omite las excepciones en donde un agente fuese considerablemente más eficiente (en términos de costo) que los restantes, lo que mostraría que la ejecución de múltiples tareas por parte de dicho agente podría tener un costo total cercano (o incluso menor) a los demás agentes, aun cuando éstos últimos tengan asignados un menor número de tareas.

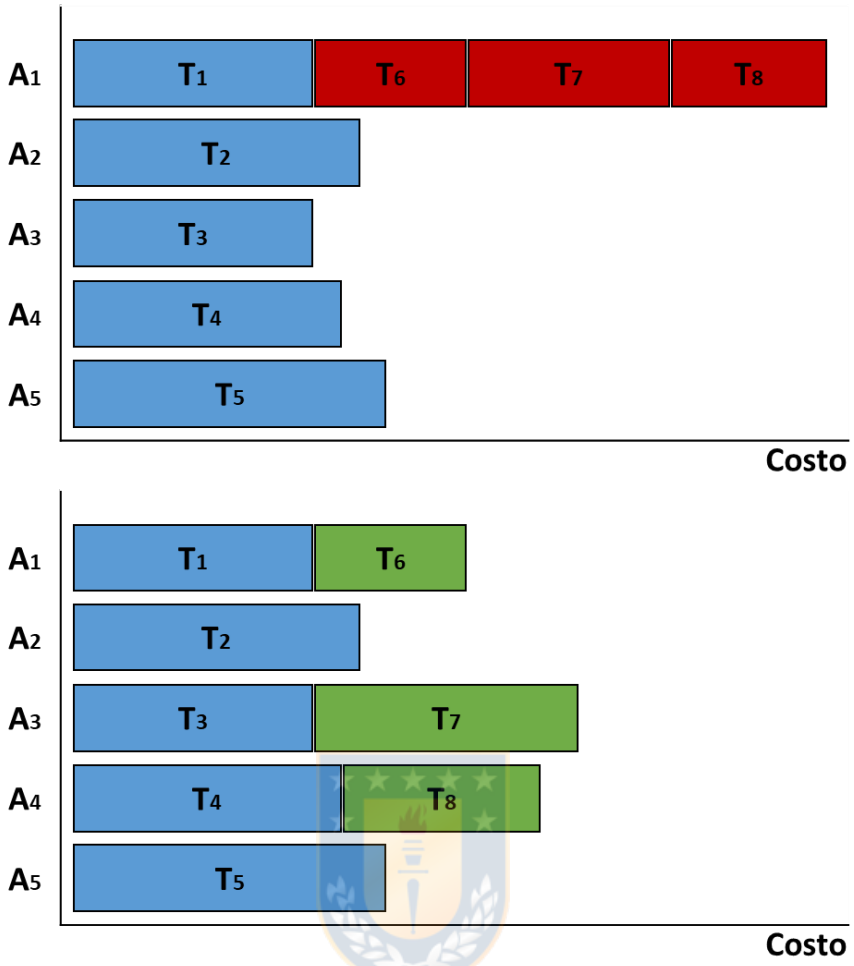

Figura 2.2. Comparativa entre soluciones con capacidad de 4 tareas (arriba) y 2 tareas (abajo) por agente
Fuente. Elaboración propia

En Betts y Vasko (2016), se aborda el caso de estudio propuesto por Kumar (2006), en donde se tiene 8 tareas y 5 agentes. Los autores, dentro de los supuestos de su modelo, parecen aludir este problema al definir en la estructura de la solución que la asignación debe realizarse de tal forma que cada máquina no puede ser utilizada más de dos veces. Con esto, y de forma implícita, se apunta a una estructura en la solución que no sólo apunte a minimizar los costos de asignación de tareas, sino también a proporcionar un mayor balance en la carga de trabajo de cada agente. Sin embargo, el supuesto presentado carece de justificación en su motivación, o una ecuación formal que defina el cómo determinar esta restricción en problemas de diferente tamaño. Sin embargo, y a pesar de la relevancia de este elemento en modelos que abordan problemas reales, son pocos los métodos presentes en la literatura que plantean un supuesto o restricción sobre el número mínimo y máximo de tareas que puede ejecutar un agente (Majumdar y Bhunia, 2012; Betts y Vasko, 2016; Bhunia et al., 2017; Biswas et al., 2018). Más aún, en ninguno de estos trabajos se entrega una motivación o justificación a su propuesta desde la perspectiva práctica, o apuntando a generar soluciones de mejor calidad o de una mayor robustez al momento de implementarlas en un ambiente real.

Capítulo 3 : Algoritmo de Entropía para el UAP

En este capítulo se presenta el Algoritmo de Entropía propuesto para resolver el UAP. A modo introductorio se describen y detallan las características del algoritmo.

En primer lugar, se presenta el modelo matemático y las restricciones del UAP, así como los conjuntos y subconjuntos contemplados por las diferentes variables a abordar en la resolución del problema. Posteriormente, en la segunda sección se exponen los supuestos y consideraciones del problema, previas a la formulación del algoritmo. Luego, en la tercera sección se presenta el Algoritmo desarrollado en este trabajo. Para ello, se inicia con la descripción del concepto de Entropía abordado por el algoritmo, así como la definición de una función matemática para calcular este valor en un UAP. Luego, se definen conceptos relacionados con el algoritmo para, posteriormente, mostrar su metodología de aplicación. Finalmente, y de forma ilustrativa, se resuelve un UAP a través del algoritmo.

El algoritmo presenta una formulación para abordar el Problema de Asignación No Balanceado, y corresponde a un método heurístico constructivo para generar asignación de tareas a agentes disponibles, de forma tal que se proporcione prioridad en la secuencia de asignación a aquellas tareas que tienen un mayor valor de entropía en el sistema.

El objetivo de este método es proporcionar una secuencia lógica de asignación de tareas, de forma tal que se minimice el costo total de asignación. Al mismo tiempo, se considera una distribución equilibrada de la carga de procesos a ejecutar entre los diferentes agentes.

El algoritmo presentado en este capítulo presenta características, metodología y un enfoque no abordado en ninguna investigación realizada hasta el momento, pero que, sin embargo, representa un caso de interés respecto a su objetivo y la capacidad de resolver el UAP con un potencial de aplicación práctico.

3.1. Del modelo matemático del problema

Siendo i una tarea a ser desarrollada, se denota al conjunto total de tareas a ejecutar por un sistema determinado por $I = \{1,2, \dots, M\}$. Toda tarea puede ser asignada a un determinado agente j, que puede hacer referencia a una máquina, servidor u operador en donde se ejecutarán dichas tareas. Se denota dicho conjunto de agentes como $J = \{1,2, \dots, N\}$. Además, y según el enfoque de interés de este trabajo, se tiene que $M > N$.

Sea C_{ij}, $i = 1,2, \dots, M$; $j = 1,2, \dots, N$ el costo de asignación de la tarea i al agente j. El UAP consiste en determinar la política de asignación de tareas a agentes de forma óptima, de manera tal que la totalidad de tareas sean ejecutadas y el costo de asignación de dichas tareas sea mínimo.

Sea la variable binaria X_{ij}, definida como:

$$X_{ij} = \begin{cases} 1, & \text{si la tarea } i \text{ es asignada al agente } j \\ 0, & \text{si la tarea } i \text{ no es asignada al agente } j \end{cases}$$

El modelo matemático para el Problema de Asignación No Balanceado con distribución equitativa de tareas, entonces, puede ser definido como:

$$\text{Minimizar: } Z = \sum_{i \in I} \sum_{j \in J} C_{ij} X_{ij} \qquad (3.1)$$

Sujeto a:

$$\sum_{i \in I} X_{ij} \geq \left\lfloor \frac{M}{N} \right\rfloor, \quad \forall j \in J \qquad (3.2)$$

$$\sum_{i \in I} X_{ij} \leq \left\lceil \frac{M}{N} \right\rceil, \quad \forall j \in J \qquad (3.3)$$

$$\sum_{j \in J} X_{ij} = 1, \quad \forall i \in I \qquad (3.4)$$

$$X_{ij} = \{0,1\}, \ \forall i \in I, \forall j \in J \qquad (3.5)$$

La función objetivo (3.1) minimiza el costo total de asignación de las tareas. Las restricciones (3.2) y (3.3) indican el límite el número mínimo y máximo de tareas a ser asignadas a cada agente. Las restricciones (3.4) establecen que ninguna tarea puede ser asignada a más de un agente. Finalmente, las restricciones (3.5) definen el dominio de las variables de decisión.

El UAP definido anteriormente es lineal, dado que la función de costo a optimizar, así como las restricciones asociadas al modelo, contienen sólo términos lineales. Además, la matriz de restricciones del UAP es totalmente unimodular y, por tanto, el problema puede ser resuelto relajando la binariedad de las variables.

3.2. De los supuestos y consideraciones del problema

En primer lugar, se define como sistema a un conjunto de agentes determinado y capacitados para ejecutar una serie de tareas y, por tanto, sin restricciones en la combinación de asignación entre ellas. Además, y desde una perspectiva computacional, se tiene que un programa de asignación a completar implica que la totalidad de tareas han sido distribuidas entre los agentes del sistema.

Como primer supuesto, se considera que el sistema está compuesto por un total de N agentes, los cuales están capacitados para ejecutar un conjunto de M tareas, donde $M > N$ (Kumar, 2006). Este supuesto corresponde a la definición de un Problema de Asignación No Balanceado y, para este caso, cae en la categoría de UAP de Tipo 1, en donde hay un mayor número de tareas que de agentes (Bhunia et al., 2017).

Como segundo supuesto, se considera que toda combinación entre tarea y el agente al cual puede ser asignado tiene un costo de asignación asociado (Rabbani et al., 2019), el cual es conocido, positivo y no cambia.

Como tercer supuesto, se considera que todos los agentes comienzan a ejecutar sus tareas asignadas de forma simultánea. Esto es, bajo un enfoque de aplicación práctica, que ninguna tarea tiene requerimientos de preprocesamiento, secuenciamiento o preparación previos a su ejecución.

A su vez, como primera restricción, se asume que cada tarea debe ser asignada a exactamente un agente (Majumdar y Bhunia, 2012; Rabbani et al., 2019).

Como segunda restricción, y según el tipo de UAP abordado en el presente trabajo, se tiene que ningún agente se deja ocioso. Esto es, a cada agente le es asignado a lo menos una tarea a ejecutar (Bhunia et al., 2017; Rabbani et al., 2019).

3.3. Del algoritmo de Entropía propuesto

A continuación, se presenta la propuesta de Algoritmo de Entropía para la resolución del UAP. La formulación del problema a abordar contempla un sistema de procesamiento o ejecución de un conjunto de tareas, haciendo uso de un número determinado de agentes disponibles. El problema recae en determinar la asignación de las tareas de manera tal que todas sean ejecutadas, al mismo tiempo que se minimizan los costos de procesamiento asociados. Los parámetros expuestos en la Tabla 3.1 corresponden a los parámetros que se consideran en la aplicación del algoritmo.

Tabla 3.1. Parámetros del Algoritmo propuesto
Fuente. Elaboración propia

Parámetro	Descripción
M	Número de Tareas a ejecutar por el sistema
N	Número de Agentes disponibles en el sistema
J_u	Máxima Asignación de Tareas a un mismo Agente
J_l	Mínima Asignación de Tareas a un Agente
M_u	Número de Agentes con Máxima Asignación de Tareas
M_l	Número de Agentes con Mínima Asignación de Tareas
C_{ij}	Costo asociado a la asignación de una Tarea i a un Agente j

3.3.1. Definición del concepto de Entropía

Entropía se define como la medida del desorden de un sistema. Definiciones similares se pueden encontrar en trabajos vinculados a Problemas de Asignación Cuadrática (González-Cruz y Gómez-Senent, 2011; Ning y Li, 2018). En este estudio, la utilidad de este concepto es la de crear una medida numérica para evaluar cuantitativamente una serie de tareas, de forma tal que pueda generarse una secuencia ordenada de las mismas para el procedimiento de asignación. En particular, el interés recae en evaluar la selección de un determinado agente para una cierta tarea, en contraste con cualquier otro agente disponible para dicha tarea. Dado que existe un costo asociado a dicha asignación, la selección de uno u otro agente puede repercutir en un incremento o reducción de costo asociado a ejecutar dicha tarea.

Ahora, para describir el concepto de Entropía utilizado, se considera, a modo de ejemplo, para una cierta tarea el menor costo de asignación posible entre los agentes disponibles. Si bajo un determinado criterio no se selecciona este agente, y en su lugar se selecciona la segunda mejor opción, es posible cuantificar esta diferencia generada en el costo de asignación como una penalización o desorden proyectado por la decisión de asignación tomada. Luego, dado que esta idea puede generalizarse a todas las alternativas de asignación de una tarea respecto a los agentes disponibles, se puede cuantificar la totalidad del desorden potencial asociado a una determinada tarea.

Por otra parte, si consideramos que es de mayor relevancia la diferencia existente entre las primeras dos opciones respecto a, por ejemplo, la cuarta y quinta opción, se debe establecer una medida capaz de representar estas diferencias en el cálculo del desorden potencial asociado a la tarea. En otras palabras, genera un mayor desorden la diferencia entre las opciones más atractivas (o más probables) para realizar la asignación, en contraste a aquellas que lo son menos.

El procedimiento para el cálculo del valor de Entropía de una Tarea i se describe a continuación:

Paso 1: Crear un vector ordenado de forma ascendente, O_i, con los costos de asignación para cada Agente j, C_{ij}, asociados a la Tarea i. Este vector estará compuesto por N elementos, siendo los elementos $O_{i,1}$ y $O_{i,N}$ iguales al menor y mayor costo, respectivamente.
Paso 2: Calcular el valor de Entropía para la Tarea i haciendo uso de los elementos del vector O_i, y a través de una función de entropía, S_i, definida como:

Para $i = 1, \dots, M$.

$$S_i = \sum_{j=1}^{N-1} \frac{(O_{i,j+1} - O_{i,j})}{j} \qquad (3.6)$$

En la Función de Entropía (Ecuación 3.6) se explica, para una determinada Tarea i, la motivación de generar un cociente entre la diferencia entre los elementos $O_{i,j+1} - O_{i,j}$ (como dividendo) y el valor j (como divisor), de forma tal que se pueda representar con un mayor valor en la sumatoria a aquellas diferencias entre los menores costos de asignación y, por tanto, ubicados en las primeras posiciones del vector de costos ordenado O_i (con un valor j reducido). Por el contrario, para aquellas diferencias de costos que se encuentren en posiciones posteriores del vector de costos ordenado O_i, el divisor j correspondiente tiene un mayor valor y, por tanto, entrega un cociente con menor valor a la sumatoria total que resulta en el valor de Entropía de la Tarea i.

Así, por tanto, el valor de entropía asociado a una Tarea i, S_i, permite no sólo representar el nivel de desorden asociado a sus posibles asignaciones, sino que también proporciona una medida cuantitativa como criterio de ordenamiento o relevancia de las diferentes tareas.

3.3.2. Capacidad de asignación

En esta sección se define la Capacidad de Asignación para los agentes del sistema. Esto es, el número de tareas (máximo o mínimo) que pueden ser asignadas a un mismo agente en la solución del UAP proporcionada por el Algoritmo. A partir de esto, se determina la Estructura de la solución según el tamaño del problema abordado (número de agentes y número de tareas). Dicha Estructura tiene como objetivo distribuir equitativamente el total de tareas entre los agentes.

En primer lugar, se tiene que la Máxima Asignación, J_u, corresponde al entero superior de la razón entre el número de tareas y el número de agentes del problema (Ecuación 3.7). De forma directa, la Mínima Asignación de tareas a asignar a un agente, J_l, se calcula según la Ecuación 3.8.

$$J_u = \left\lceil \frac{M}{N} \right\rceil \tag{3.7}$$

$$J_l = J_u - 1 \tag{3.8}$$

Luego, en base a 3.7. y 3.8, se define la Estructura de la solución. Dicha Estructura se entiende como el formato de asignación de tareas a agentes que se pretende construir en la solución al problema, sin conocer la asignación propiamente tal. Así, la Estructura comprende el número de agentes con un número de tareas asignadas igual a J_u, correspondiente a M_u, el número de agentes con un número de tareas asignadas igual a J_l, correspondiente a M_l. M_u y M_l se calculan como se muestra en las Ecuaciones 3.9 y 3.10, respectivamente.

$$M_u = (M - J_l * N) \tag{3.9}$$

$$M_l = N - M_u \tag{3.10}$$

3.4.1. Metodología de aplicación

En esta sección se considera, como paso inicial, retomar el trabajo realizado en la sección 3.3.2, en donde fue construida la estructura de la solución. Esto es, ya se ha determinado el número de agentes que tendrán asignados J_u tareas (M_u agentes) y el número de agentes que tendrán asignados J_l tareas (M_l agentes).

Como segundo paso, se procede a calcular los valores de Entropía asociados a cada tarea, haciendo uso de la Función de Entropía definida en la sección 3.3.1. Luego, se crea una lista E, cuyos elementos corresponden a las M tareas ordenadas de forma descendente según el valor de Entropía asociado a cada una. En el caso que dos o más tareas tengan el mismo valor de Entropía, se procede a ordenar de forma arbitraria las tareas en cuestión dentro de la lista E.

Luego, y como tercer paso, siguiendo el orden anteriormente generado, se considera la primera tarea de la lista para ser asignada a los agentes disponibles (que inicialmente es la totalidad de los agentes), y se procede a seleccionar aquel para el cual se tiene el menor costo de asignación. De esta forma, esta primera tarea se remueve de la lista ordenada, y queda asignada al agente.

A continuación, se procede a revisar el registro de utilización del agente en cuestión, es decir, el número de tareas que le han sido asignadas (en este caso, pasará de cero tareas a una tarea). En el caso que el registro haya alcanzado la Capacidad de Asignación (inicialmente definida como igual a J_u), se procede a removerlo de los agentes disponibles. Esto es, dichos agentes dejan de estar disponibles para la asignación de las próximas tareas en la secuencia y, por tanto, la selección entre los menores costos de asignación se reduce a los restantes agentes disponibles.

Posteriormente, si el número de agentes removidos alcanza el valor de M_u (Número de Agentes con Capacidad de Asignación J_u utilizada), entonces se cambia el valor de la Capacidad de Asignación, de J_u a J_l, y se remueve cualquier agente cuyo registro de utilización ya tenga dicho valor (igual a J_l). Esto se entiende como la utilización completa de aquellos agentes que podían tener una Capacidad de Asignación igual a J_u y, por tanto, se dispone a distribuir tareas sólo entre agentes que pueden ejecutar hasta un máximo de J_l tareas cada uno.

Luego de lo anterior, el procedimiento vuelve a repetirse desde el tercer paso, ahora considerando la siguiente tarea en la lista ordenada. Esta iteración es ejecutada hasta que todas las tareas hayan sido asignadas a algún agente.

Finalmente, y con la totalidad de tareas asignadas, se procede a calcular el costo total de asignación asociado a la solución alcanzada. Esto es, sumar cada uno de los costos de asignación de cada tarea respecto al agente correspondiente. La Figura 3.1 ilustra secuencialmente las etapas que comprenden el algoritmo presentado.

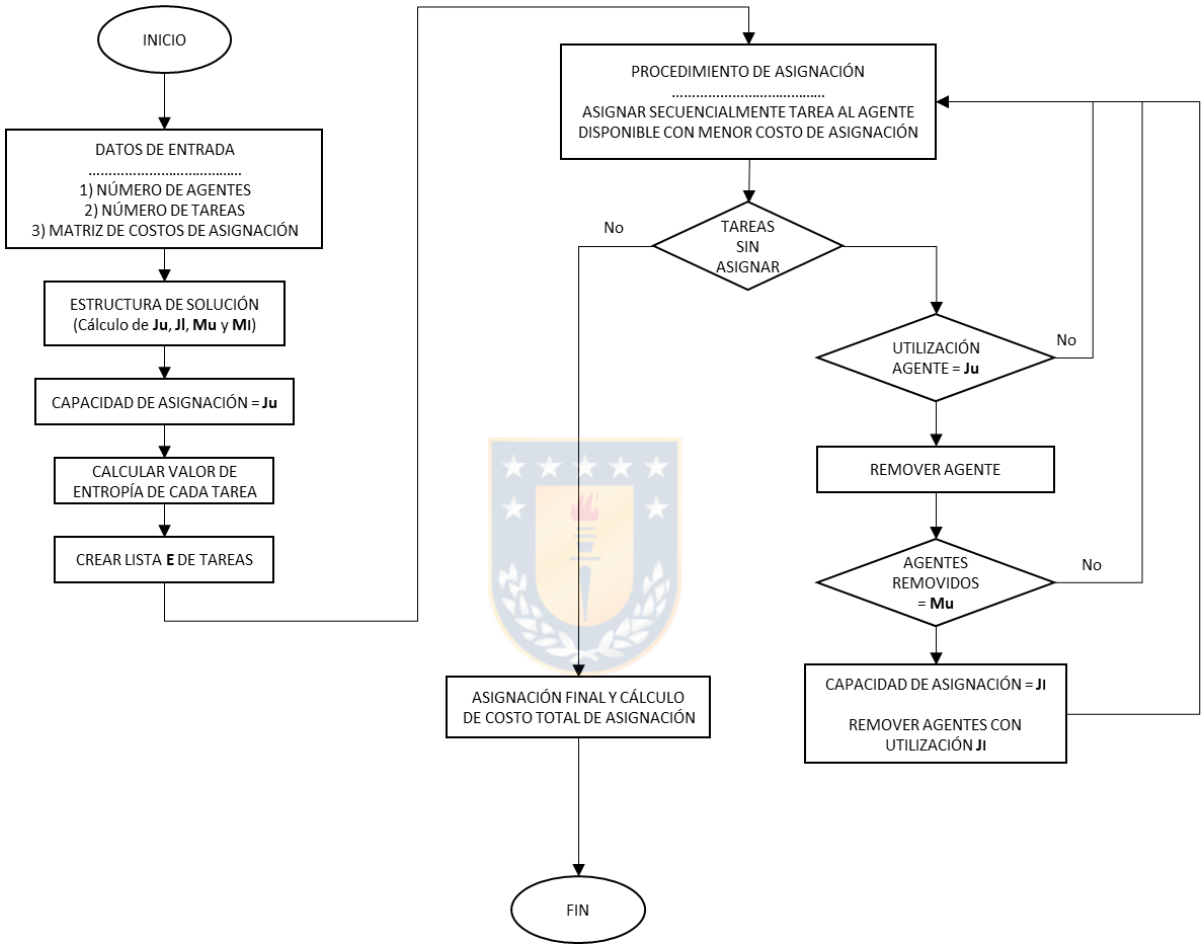

Figura 3.1. Diagrama de Flujo para el Algoritmo propuesto
Fuente. Elaboración propia

En la implementación del algoritmo presentado en este trabajo, para un problema de M tareas y N agentes, se tiene el registro de una complejidad temporal de orden $\mathcal{O}(M * N * log(N))$, considerando que la operación de mayor complejidad corresponde al ordenamiento ascendente de los Costos de Asignación relacionados a cada uno de los N agentes (complejidad de orden $N * log(N)$), y esto se ejecuta para cada una de las M Tareas. Más aún, la complejidad espacial del algoritmo propuesto es de orden $\mathcal{O}(MN)$, al ser la matriz de costos de asignación el elemento de mayor tamaño a considerar en la memoria.

3.4.5. Ejemplo ilustrativo de aplicación del algoritmo

En esta sección se procede a ilustrar con un ejemplo práctico la metodología de aplicación del algoritmo propuesto. Considere una instalación productiva, en donde se requiere la ejecución de 5 trabajos y se tienen 3 estaciones de trabajo disponibles. La Matriz de Costos de Asignación del Problema se presenta en la Tabla 3.2.

Tabla 3.2. Matriz de Costos de Asignación para el problema ilustrativo
Fuente. Elaboración propia

	A_1	A_2	A_3
T_1	90	70	60
T_2	70	50	70
T_3	60	50	30
T_4	60	80	70
T_5	70	90	50

En primer lugar, se procede a determinar la Estructura de la Solución. Para el problema en cuestión, respecto a la asignación se tiene una Máxima Asignación $J_u = \left\lceil \frac{5}{3} \right\rceil = 2$ tareas, y una Mínima Asignación $J_l = J_u - 1 = 1$ tarea. Al mismo tiempo, se calcula el número de agentes con Máxima Asignación como $M_u = (5 - 1 * 3) = 2$ agentes, y el número de agentes con Mínima Asignación como $M_l = 3 - 2 = 1$ agente. La Capacidad de Asignación inicial para aplicar el algoritmo es igual a $J_u = 2$.

En segundo lugar, se procede a calcular el valor de Entropía asociado a cada tarea. Para ello, se procede a ordenar de forma ascendente los costos de asignación de cada tarea, generando para cada tarea el vector O_i correspondiente. Luego, se aplica la Función de Entropía definida en la sección 3.3.1. De forma ilustrativa, en la Figura 3.2 se muestra el cálculo del valor de Entropía para la Tarea 1, igual a 20 en este caso. Para las Tareas 2, 3, 4 y 5 se tienen los valores de Entropía 20, 25, 15 y 30, respectivamente. El orden de tareas, en base al valor de entropía, corresponde a T_5, T_3, T_1, T_2 y T_4.

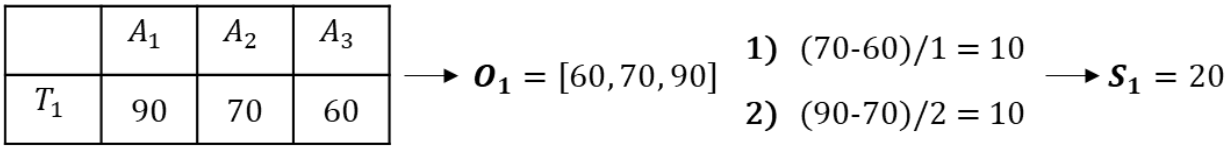

Figura 3.2. Ilustración del cálculo del valor de Entropía para la Tarea 1
Fuente. Elaboración propia

En tercer lugar, se procede a asignar secuencialmente las tareas al agente correspondiente, según el menor costo de asignación, y siguiendo el orden definido en el paso anterior. Así, para este caso, se considera asignar la Tarea T_5 al Agente A_3, combinación que tiene el menor costo de asignación dentro de los agentes disponibles (igual a 50).

Como existen aún tareas sin asignar, se procede a revisar la utilización del Agente respecto a la Capacidad de Asignación. En este caso, la utilización de A_3 asciende a 1 tarea, y como $J_u = 2$, se procede a la asignación de la segunda tarea en la lista ordenada.

En este caso, la segunda tarea es T_3, la cual también será asignada al Agente A_3 por tener el menor costo de asignación. Al revisar la utilización del Agente A_3, se tiene que esta asciende a 2 tareas, igual a la Capacidad de Asignación J_u. Luego, se procede a remover este agente de la lista de agentes disponibles. Seguidamente, se revisa que el número de agentes removidos asciende a 1 agente, siendo menor al Número de Agentes con Asignación Máxima ($M_u = 2$), por lo que se procede a la asignación de la tercera tarea en la lista ordenada.

El procedimiento contempla asignar las dos tareas siguientes, T_1 y T_2. Como ambas tienen igual valor de Entropía, arbitrariamente se decide establecer en la secuencia de asignación como tercera y cuarta tareas a T_1 y T_2, respectivamente. Ambas son asignadas al Agente A_2, según el menor costo de asignación dentro de los agentes disponibles (sólo A_1 y A_2 en esta etapa). Al revisar la utilización del agente, luego de asignar la cuarta tarea, se tiene que la utilización de A_2 asciende a 2 tareas, lo que es igual a la Capacidad de Asignación J_u. Así, se procede a remover este agente de la lista de agentes disponibles. A continuación, al revisar que el número de agentes removidos es igual a 2 agentes (igual al Número de Agentes con Asignación Máxima M_u), se establece como nueva Capacidad de Asignación a $J_l = 1$. Además, se procede a revisar si ya existen agentes con una utilización igual a 1 tarea para ser removidos, lo que no se cumple en este caso.

Luego, se procede a asignar la última tarea, T_4, al único agente aún disponible, correspondiente al Agente A_1. Como no hay más tareas sin asignar, se continua a la última etapa del algoritmo.

Finalmente, se procede a calcular el costo total de asignación de la distribución generada. En la Figura 3.3 se presenta la solución generada en este ejemplo. En este caso, se tiene que el costo total de asignación de la solución generada alcanza 270.

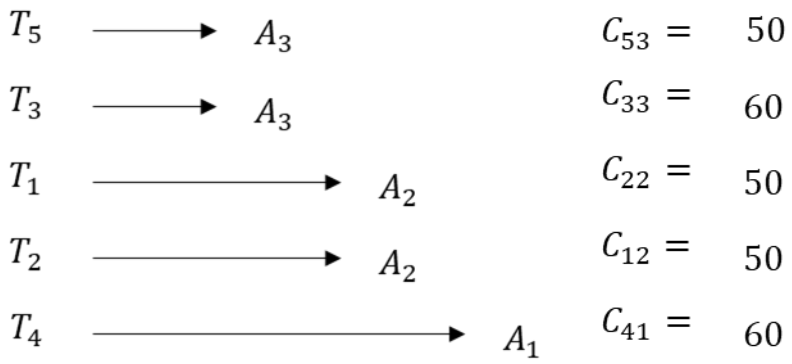

Figura 3.3. Representación del método de asignación y solución alcanzada por el algoritmo propuesto
Fuente. Elaboración propia

Capítulo 4 : Experimentación y resultados

Este capítulo se divide en dos secciones. En la primera sección, se presentan las instancias utilizadas para la aplicación y evaluación del algoritmo propuesto para resolver el UAP.

En la segunda sección, se presentan y analizan los resultados obtenidos luego de la aplicación del algoritmo propuesto. Además, los resultados obtenidos son comparados con los resultados de otros métodos aplicados en trabajos previos, así como el resultado óptimo para cada caso.

4.1. Instancias de prueba

A continuación, se presentan las instancias consideradas para la aplicación del algoritmo propuesto. Estas instancias corresponden, por una parte, a un conjunto de problemas recopilados de la literatura, lo que es de interés para evaluar la flexibilidad y robustez del algoritmo propuesto. Además, se añade un conjunto de instancias de prueba adicionales, generadas de forma aleatoria y con la intención de evaluar estadísticamente el desempeño del algoritmo propuesto, respecto a calidad de la solución y tiempo de ejecución del algoritmo propuesto.

4.1.1. Caso de estudio Kumar (2006)

El caso a presentar a continuación corresponde a un problema de asignación no balanceado, el cual tiene como objetivo determinar la asignación y ejecución óptima de un conjunto de 8 trabajos considerando un sistema productivo que comprende 5 máquinas disponibles. En detalle, los parámetros del problema abordado se presentan en la Tabla 4.1. En la Tabla 4.2 se presenta la Matriz de Costos de Asignación del problema. Este caso de estudio es presentado en Kumar (2006), y citado sucesivamente por Yadaiah y Haragopal (2016), Betts y Vasko (2016) para evaluar la efectividad y rendimiento de diferentes métodos para resolver el UAP.

Tabla 4.1. Parámetros generales para Caso de Estudio Kumar (2006)

Fuente. Elaboración propia

Parámetro	Descripción	Valor
M	Conjunto de Tareas a ejecutar en el sistema	8
N	Conjunto de Agentes disponibles en el sistema	5
J_u	Máxima Asignación de tareas a un mismo Agente	2
J_l	Mínima Asignación de tareas a un Agente	1
M_u	Número de Agentes con Máxima Asignación	3
M_l	Número de Agentes con Mínima Asignación	2

Tabla 4.2. Matriz de Costos de Asignación del Caso de Estudio Kumar (2006)

Fuente. Kumar (2006)

	A_1	A_2	A_3	A_4	A_5
T_1	300	290	280	290	210
T_2	250	310	290	300	200
T_3	180	190	300	190	180
T_4	320	180	190	240	170
T_5	270	210	190	250	160
T_6	190	200	220	190	140
T_7	220	300	230	180	160
T_8	260	190	260	210	180

4.1.2. Caso de estudio Majumdar y Bhunia (2012)-A

El segundo caso de estudio corresponde a un problema de asignación no balanceado, presentado como el primer problema de prueba en Majumdar y Bhunia (2012), y el cual tiene como objetivo determinar la asignación y ejecución óptima de un conjunto de 7 tareas, considerando un sistema que comprende 5 agentes disponibles. En detalle, los parámetros del problema abordado se presentan en la Tabla 4.3. En la Tabla 4.4 se presenta la Matriz de Costos de Asignación del problema.

Tabla 4.3. Parámetros generales para Caso de Estudio Majumdar y Bhunia (2012)-A
Fuente. Elaboración propia

Parámetro	Descripción	Valor
M	Conjunto de Tareas a ejecutar en el sistema	7
N	Conjunto de Agentes disponibles en el sistema	5
J_u	Máxima Asignación de tareas a un mismo Agente	2
J_l	Mínima Asignación de tareas a un Agente	1
M_u	Número de Agentes con Máxima Asignación	2
M_l	Número de Agentes con Mínima Asignación	3

Tabla 4.4. Matriz de Costos de Asignación del Caso de Estudio Majumdar y Bhunia (2012)-A
Fuente. Majumdar y Bhunia (2012)

	A_1	A_2	A_3	A_4	A_5
T_1	30	29	28	29	21
T_2	25	31	29	30	20
T_3	18	19	30	19	18
T_4	32	18	19	24	17
T_5	27	21	19	25	16
T_6	19	20	22	19	14
T_7	22	30	23	18	16

4.1.3. Caso de estudio Majumdar y Bhunia (2012)-B

El tercer caso de estudio corresponde a un problema de asignación no balanceado, presentado como el tercer problema de prueba en Majumdar y Bhunia (2012), y el cual tiene como objetivo determinar la asignación y ejecución óptima de un conjunto de 10 tareas, considerando un sistema que comprende 6 agentes disponibles. En detalle, los parámetros del problema abordado se presentan en la Tabla 4.5. En la Tabla 4.6 se presenta la Matriz de Costos de Asignación del problema.

Tabla 4.5. Parámetros generales para Caso de Estudio Majumdar y Bhunia (2012)-B
Fuente. Elaboración propia

Parámetro	Descripción	Valor
M	Conjunto de Tareas a ejecutar en el sistema	10
N	Conjunto de Agentes disponibles en el sistema	6
J_u	Máxima Asignación de tareas a un mismo Agente	2
J_l	Mínima Asignación de tareas a un Agente	1
M_u	Número de Agentes con Máxima Asignación	4
M_l	Número de Agentes con Mínima Asignación	2

Tabla 4.6. Matriz de Costos de Asignación del Caso de Estudio Majumdar y Bhunia (2012)-B
Fuente. Majumdar y Bhunia (2012)

	A_1	A_2	A_3	A_4	A_5	A_6
T_1	10	7	4	21	10	15
T_2	2	12	8	9	12	7
T_3	14	9	6	12	30	34
T_4	9	3	12	9	15	17
T_5	6	5	21	32	12	7
T_6	7	6	9	10	17	16
T_7	21	9	21	19	30	14
T_8	32	16	14	25	12	17
T_9	18	54	45	16	12	9
T_{10}	11	12	13	10	9	5

4.1.4. Caso de estudio Majumdar y Bhunia (2012)-C

El cuarto caso de estudio corresponde a un problema de asignación no balanceado, presentado como el cuarto problema de prueba en Majumdar y Bhunia (2012), y el cual tiene como objetivo determinar la asignación y ejecución óptima de un conjunto de 10 tareas, considerando un sistema que comprende 7 agentes disponibles. En detalle, los parámetros del problema abordado se presentan en la Tabla 4.7. En la Tabla 4.8 se presenta la Matriz de Costos de Asignación del problema.

Tabla 4.7. Parámetros generales para Caso de Estudio Majumdar y Bhunia (2012)-C
Fuente. Elaboración propia

Parámetro	Descripción	Valor
M	Conjunto de Tareas a ejecutar en el sistema	10
N	Conjunto de Agentes disponibles en el sistema	7
J_u	Máxima Asignación de tareas a un mismo Agente	2
J_l	Mínima Asignación de tareas a un Agente	1
M_u	Número de Agentes con Máxima Asignación	3
M_l	Número de Agentes con Mínima Asignación	4

Tabla 4.8. Matriz de Costos de Asignación del Caso de Estudio Majumdar y Bhunia (2012)-C
Fuente. Majumdar y Bhunia (2012)

	A_1	A_2	A_3	A_4	A_5	A_6	A_7
T_1	21	14	9	16	12	8	21
T_2	11	15	17	23	40	18	9
T_3	16	20	11	8	14	9	12
T_4	9	10	31	15	36	42	9
T_5	15	16	21	10	9	8	32
T_6	10	3	16	3	21	11	10
T_7	12	6	7	6	14	19	19
T_8	32	9	9	3	19	9	25
T_9	26	21	10	20	4	32	16
T_{10}	16	14	11	23	13	20	10

4.1.5. Caso de estudio Mondal et al. (2017)

El quinto caso de estudio corresponde a un problema de asignación no balanceado, presentado en Mondal et al. (2017), y el cual tiene como objetivo determinar la asignación y ejecución óptima de un conjunto de 8 tareas, considerando un sistema que comprende 5 agentes disponibles. En detalle, los parámetros del problema abordado se presentan en la Tabla 4.9. En la Tabla 4.10 se presenta la Matriz de Costos de Asignación del problema.

Tabla 4.9. Parámetros generales para Caso de Estudio Mondal et al. (2017)
Fuente. Elaboración propia

Parámetro	Descripción	Valor
M	Conjunto de Tareas a ejecutar en el sistema	8
N	Conjunto de Agentes disponibles en el sistema	5
J_u	Máxima Asignación de tareas a un mismo Agente	2
J_l	Mínima Asignación de tareas a un Agente	1
M_u	Número de Agentes con Máxima Asignación	3
M_l	Número de Agentes con Mínima Asignación	2

Tabla 4.10. Matriz de Costos de Asignación del Caso de Estudio Mondal et al. (2017)
Fuente. Mondal et al. (2017)

	A_1	A_2	A_3	A_4	A_5
T_1	151	277	185	276	321
T_2	245	286	256	264	402
T_3	246	245	412	423	257
T_4	269	175	145	125	156
T_5	421	178	185	425	235
T_6	257	257	125	325	362
T_7	159	268	412	256	286
T_8	365	286	236	314	279

4.1.6. Caso de estudio Khandelwal (2018)

El sexto caso de estudio corresponde a un problema de asignación no balanceado, presentado en Khandelwal (2018), y el cual tiene como objetivo determinar la asignación y ejecución óptima de un conjunto de 6 trabajos, considerando un sistema productivo que comprende 4 máquinas disponibles. En detalle, los parámetros del problema abordado se presentan en la Tabla 4.11. En la Tabla 4.12 se presenta la Matriz de Costos de Asignación del problema.

Tabla 4.11. Parámetros generales para Caso de Estudio Kumar (2006)
Fuente. Elaboración propia

Parámetro	Descripción	Valor
M	Conjunto de Tareas a ejecutar en el sistema	6
N	Conjunto de Agentes disponibles en el sistema	4
J_u	Máxima Asignación de tareas a un mismo Agente	2
J_l	Mínima Asignación de tareas a un Agente	1
M_u	Número de Agentes con Máxima Asignación	2
M_l	Número de Agentes con Mínima Asignación	2

Tabla 4.12. Matriz de Costos de Asignación del Caso de Estudio Khandelwal (2018)
Fuente. Khandelwal (2018)

	A_1	A_2	A_3	A_4
T_1	6	5	1	6
T_2	2	5	3	7
T_3	3	7	2	8
T_4	7	7	5	9
T_5	12	8	8	6
T_6	6	9	5	10

4.1.7. Instancias de prueba adicionales

En esta sección se presenta un experimento estadístico desarrollado para evaluar el desempeño del algoritmo propuesto. Específicamente, el interés de este análisis recae en comparar los valores de las soluciones alcanzadas por el algoritmo propuesto y GAMS, así como los tiempos de ejecución asociados a cada método.

Para el experimento, consideramos un conjunto total de 200 instancias de prueba presentes en un repositorio abierto (https://github.com/marceloigallegos/UAP200). Además, se procede a replicar los parámetros establecidos en Campelo y Wanner (2020) para la comparación de dos métodos de resolución, considerando como parámetros un tamaño del efecto una d de Cohen igual a $d = 0.5$ (lo que alude al interés en detectar diferencias iguales o mayores a la mitad de la desviación estándar), un nivel de significancia de $\alpha = 0.05$ y un poder estadístico de $\pi = 0.8$ (para $\beta^* = 0.2$).

Luego, según los parámetros definidos, se tiene un número de $I^* = 57$ instancias como la menor cantidad de casos necesarios para obtener un experimento con las propiedades deseadas. Este número de instancias definen la muestra de prueba, siendo seleccionadas de forma aleatoria y sin reemplazo del conjunto total de instancias (ver Tabla 4.13).

Tabla 4.13. Descripción de Instancias seleccionadas
Fuente. Elaboración propia.

Instancia	Tareas	Agentes	Instancia	Tareas	Agentes	Instancia	Tareas	Agentes
2	8	4	74	96	10	136	256	14
8	8	6	76	96	12	143	320	8
9	8	6	81	128	6	148	320	10
10	8	6	89	128	8	152	320	12
12	16	4	92	128	10	153	320	12
15	16	4	93	128	10	154	320	12
21	32	4	94	128	10	157	320	14
25	32	4	96	128	12	159	320	14
33	48	6	102	192	8	165	384	8
36	48	8	103	192	8	172	384	12
39	48	8	105	192	8	176	512	5
42	48	10	107	192	10	177	512	5
44	48	10	112	192	12	180	512	5
49	64	6	115	192	12	183	512	7
52	64	8	122	256	8	186	512	9
60	64	10	123	256	8	188	512	9
63	96	6	129	256	10	197	24	6
70	96	8	131	256	12	199	24	6
73	96	10	135	256	12	200	24	6

4.2. Análisis de resultados

A continuación, se tienen los resultados obtenidos luego de la aplicación del algoritmo propuesto. En primer lugar, se presenta un análisis comparativo éstos con los resultados alcanzados por otros métodos aplicados en la literatura para las mismas instancias. Las soluciones alcanzadas por el algoritmo propuesto para estas instancias se presentan en el Anexo I. Luego, se describe el análisis estadístico aplicado para la evaluación del algoritmo propuesto, respecto a la calidad de las soluciones alcanzadas y el tiempo de ejecución del mismo.

El algoritmo propuesto fue programado en el lenguaje Python, utilizando la versión 3.8.3 del software, haciendo uso del entorno de desarrollo integrado Spyder (versión 4.1.4). Además, se instalaron las librerías NumPy (versión 1.18.5) y Pandas (versión 1.0.5). El modelo matemático del UAP fue implementado en GAMS para resolver cada instancia, y con propósitos comparativos. Además, el análisis estadístico fue desarrollado en el lenguaje y entorno R. Por último, la ejecución del algoritmo propuesto y la ejecución del modelo matemático en GAMS, así como la implementación del análisis estadístico en R, fue realizada en un computador con procesador AMD Ryzen 5, 3.20 GHz (6 CPU) y 8 GB de memoria RAM.

El modelo matemático corresponde al presentado en la sección 3.1, es decir, contempla una distribución equitativa del número de tareas entre los agentes del sistema.

En primer lugar, en la Tabla 4.14 se presentan las soluciones alcanzadas por el algoritmo propuesto, además de los tiempos de ejecución del mismo, para todas las instancias presentadas en la sección anterior. La distribución de tareas por agente asociadas a cada solución se presenta en el Apéndice B. La más inmediata observación generada de esta información es que el algoritmo propuesto resuelve el UAP, considerando sus supuestos y haciendo uso del enfoque basado en entropía propuesto. Más aún, los tiempos de ejecución son razonablemente pequeños en cada instancia abordada (tiempos inferiores a 1 segundo), lo que apoya la premisa de que se trata de un algoritmo eficiente.

Tabla 4.14. Resultados de las soluciones obtenidas por el algoritmo propuesto
Fuente. Elaboración propia

Instancia	Costo de Asignación	Tiempo de Ejecución (s)
Kumar (2006)	1520	0.009
Majumdar and Bhunia (2012)-A	133	0.007
Majumdar and Bhunia (2012)-B	66	0.009
Majumdar and Bhunia (2012)-C	69	0.011
Mondal et al. (2017)	1533	0.007
Khandelwal (2018)	27	0.006
Problema 1	1350	0.008
Problema 2	1690	0.009
Problema 3	2850	0.015
Problema 4	5510	0.030
Problema 5	7240	0.038
Problema 6	7960	0.079

La Tabla 4.15 se presenta una comparativa de los resultados obtenidos por el algoritmo propuesto y diferentes métodos presentados en la literatura, además de la solución óptima obtenida por GAMS. El gap entre el costo total de la solución óptima y la obtenida por el algoritmo fue calculado a través de la relación $(HEU - OPT)/OPT$, donde HEU y OPT corresponden a los costos totales de las soluciones del algoritmo y óptima, respectivamente. De esta información, en primer lugar, se aprecia que el algoritmo propuesto alcanza la solución óptima para el problema presentado en la subsección 4.1.1., igualando o superando a los métodos anteriormente aplicados para la misma instancia. Luego, para las instancias presentadas en las subsecciones 4.1.2, 4.1.3 y 4.1.4, el algoritmo propuesto es capaz de alcanzar la solución óptima en los tres casos, en contraste con las dos variantes del GA propuesto por Majumdar y Bhunia (2012). Para las instancias presentada en la subsección 4.1.5, el algoritmo propuesto obtiene una mejor solución que la presentada en Mondal et al. (2017), y además alcanzando un gap considerablemente pequeño (2.5%). Por último, para la instancia presentada en la subsección 4.1.6, el algoritmo propuesto obtiene una mejor solución que la presentada por Khandelwal (2018), alcanzando el mayor gap (12.5%). Esto puede ser explicado a través del estrecho rango de valores de la matriz de costos del problema, lo que implicaría una mayor similitud entre los valores de Entropía asociados a cada tarea.

Tabla 4.15. Comparativa entre el algoritmo propuesto, otros métodos y la solución óptima
Fuente. Elaboración propia

Instancia	Método									
	Kumar (2006)	Yadaiah y Haragopal (2016)	Betts y Vasko (2016)	Majumdar y Bhunia (2012) GA-1	Majumdar y Bhunia (2012) GA-2	Mondal et al. (2017)	Khandelwal (2018)	Algoritmo propuesto	Solución óptima	Gap
Kumar (2006)	1550	1550	**1520**	**1520**	**1520**	**-**	**-**	**1520**	**1520**	0%
Majumdar y Bhunia (2012)-A	-	-	-	**133**	**133**	-	**-**	**133**	**133**	0%
Majumdar y Bhunia (2012)-B	-	-	**-**	69	**66**	-	**-**	**66**	**66**	0%
Majumdar y Bhunia (2012)-C	-	-	-	**69**	74	-	**-**	**69**	**69**	0%
Mondal et al. (2017)	-	-	-	-	-	1591	-	1533	**1459**	2.5%
Khandelwal (2018)	-	-	-	-	-	-	30	27	**24**	12.5%

Por otra parte, en la Tabla 4.16 se detallan los valores de las soluciones alcanzadas por GAMS y el algoritmo propuesto. En detalle, se denota como **c-G** y **c-A** a los valores de las soluciones alcanzadas por GAMS y el algoritmo propuesto, respectivamente. De igual forma, se denota como **t-G** y **t-A** a los tiempos de ejecución asociados al modelo implementado en GAMS y el algoritmo propuesto, respectivamente. Estos datos serán agrupados en muestras, según el método aplicado (GAMS y algoritmo aplicado) y la variable de estudio (valor de la solución y tiempo de ejecución).

Tabla 4.16. Resumen de los resultados para cada Instancia seleccionada
Fuente. Elaboración propia.

Instancia	c-G	t-G (s)	c-A	t-A (s)	Instancia	c-G	t-G (s)	c-A	t-A (s)
2	1990	0,065	2090	0	105	38350	0,271	39370	0,078
8	1590	0,091	1590	0	107	36840	0,443	38350	0,094
9	2020	0,053	2120	0	112	30060	0,304	31150	0,094
10	1440	0,067	1440	0	115	31220	0,189	32650	0,097
12	4240	0,053	4430	0	122	50200	0,464	51690	0,094
15	4590	0,055	4730	0	123	53790	0,337	54620	0,093
21	8160	0,064	8410	0	129	45180	0,503	46790	0,109
25	8200	0,07	8480	0	131	42620	0,516	45770	0,125
33	11210	0,084	12360	0,015	135	42720	0,446	43850	0,125
36	9970	0,128	10790	0,015	136	41750	0,285	40610	0,141
39	9610	0,174	9990	0,015	143	60770	0,553	63460	0,126
42	9200	0,14	9620	0,016	148	55880	0,82	60000	0,108
44	8110	0,117	8460	0,015	152	52470	0,309	55140	0,156
49	13870	0,107	14340	0,015	153	53000	0,448	53800	0,156
52	13390	0,223	13620	0,015	154	53330	0,35	55800	0,168
60	12250	0,17	13050	0,031	157	49440	0,308	50520	0,149
63	20170	0,197	21900	0,031	159	49280	0,405	50360	0,178
70	19940	0,153	20260	0,030	165	74600	0,434	77140	0,137
73	17650	0,316	18410	0,047	172	62600	0,607	64280	0,191
74	17520	0,213	18310	0,044	176	125080	0,358	126880	0,173
76	16040	0,282	17560	0,047	177	123510	0,315	125720	0,136
81	28590	0,351	29390	0,047	180	124090	0,551	124460	0,141
89	27340	0,23	28840	0,047	183	102670	0,509	103690	0,162
92	23370	0,127	24580	0,061	186	95490	0,483	96790	0,203
93	21730	0,402	22570	0,062	188	95560	0,904	98300	0,206
94	23080	0,195	23740	0,062	197	5170	0,101	5270	0
96	20290	0,566	21230	0,062	199	4950	0,08	5060	0
102	37680	0,447	38680	0,078	200	6260	0,057	6460	0
103	38280	0,13	40570	0,060					

En primer lugar, para el análisis estadístico de los valores de la solución fue ejecutada una prueba de hipótesis asociada a la diferencia de medias respecto a los valores de las soluciones, esto es, evaluar si existen o no diferencias significativas entre las medias de las soluciones alcanzadas por el algoritmo propuesto y GAMS. Dado que la prueba de varianzas indica que no existen diferencias significativas entre ambas muestras, fue aplicada una Prueba *t* de Student. Como resultado, se tiene que estadísticamente no existen diferencias significativas entre las medias de los valores de costo alcanzados por GAMS y el algoritmo propuesto, lo que permite señalar que el gap entre ambas soluciones es a lo sumo significativamente pequeño.

En segundo lugar, el análisis estadístico de los tiempos de ejecución fue ejecutada una prueba de hipótesis asociada a la diferencia de medias respecto a los tiempos, esto es, evaluar si existen o no diferencias significativas entre las medias del tiempo de ejecución requerido por el algoritmo propuesto y GAMS. Dado que la prueba de varianzas indica que existen diferencias significativas entre ambas muestras, fue aplicada una Prueba *t* de Welch. Como resultado, se tiene que estadísticamente existen diferencias significativas entre las medias de tiempos de ejecución requeridos por GAMS y por el algoritmo propuesto, lo que permite señalar que el algoritmo propuesto es significativamente más eficiente en ejecución que GAMS.

Capítulo 5 : Conclusiones

En este trabajo se estudia la formulación de algoritmo basado en entropía para resolver el problema de asignación no balanceado. El problema de asignación, entendido como la distribución de un conjunto de tareas entre un conjunto de agentes o elementos que constituyen un sistema, corresponde a un problema esencial para apoyar la toma de decisiones en la gestión de recursos, procesos y logística, así como para garantizar la calidad y optimalidad en los procesos de una organización, tanto bajo perspectivas operativas como estratégicas.

Mientras la mayoría de los métodos para abordar el problema de asignación presentados en la literatura e investigaciones abordan estrategias basadas directamente en el costo de asignación, en este trabajo se aplica un enfoque basado en cuantificar el nivel de desorden potencial que puede generar la selección de una determinada tarea, por sobre las demás, hacia un agente disponible. Hasta el momento no se conocen otras investigaciones sobre la aplicación de un algoritmo basado en un enfoque de entropía para resolver un UAP.

Los principales factores considerados para la aplicación y evaluación del algoritmo se remiten a las diferencias entre los costos de asignación de una tarea con respecto a los agentes candidatos, así como la ponderación de estas alternativas en cuanto a su atractivo. Por último, el algoritmo ordena de forma descendente las tareas a asignar según el valor de la entropía, entendiendo que dicho valor muestra la relevancia de una determinada tarea respecto a las demás.

Mediante la programación del algoritmo, haciendo uso del lenguaje Python, fue posible resolver y validar el método en un conjunto amplio de problemas presentados en la literatura. Los resultados alcanzados revelan que este nuevo enfoque de algoritmo basado en entropía para solucionar el UAP cumple con el objetivo de minimizar el costo total de asignación en un sistema, a la vez que iguala o supera a otros métodos anteriormente desarrollados. Más aún, el análisis estadístico presentado en este trabajo evidencia que las soluciones alcanzadas por el algoritmo no son significativamente diferentes a las soluciones óptimas (esto es, el gap es significativamente pequeño), así como también evidencia que los tiempos de ejecución del algoritmo son significativamente inferiores a los consumidos por un solver exacto. En base a lo anterior, es posible validar la Hipótesis presentada en este trabajo, estableciendo que este algoritmo heurístico, basado en el impacto de las diferencias de costos de asignación de cada tarea, puede resolver el UAP alcanzando soluciones que contemplan la distribución equitativa de tareas entre los agentes del sistema. El propósito de la distribución equitativa de tareas es definir al algoritmo como una herramienta capaz de apoyar la toma de decisiones en entornos reales, y proporcionar soluciones robustas para su aplicación práctica.

Como trabajos futuros se propone, en primer lugar, modificar y aplicar el algoritmo a problemas que contemplen costos de asignación definidos por intervalos, es decir, que no están definidos de forma explícita. Lo anterior en base al incremental interés presente en la literatura por abordar este tipo de problemas, así como su potencial aplicación a contextos prácticos en donde no se conoce con certeza el costo de asignación de una determinada tarea a un agente, pero existe una noción sobre un determinado rango en donde dicho costo puede fluctuar.

En segundo lugar, podría ser de interés considerar modificar las etapas de iteración del algoritmo y/o la aplicación de la función de entropía para establecer el orden de asignación de tareas. Esto puede entenderse como, luego de asignar la primera tarea a un agente, recalcular la Entropía para las tareas restantes para ver el orden de asignación resultante. De forma alternativa, este proceso también podría realizarse cada vez que un agente haya alcanzado su Máxima Asignación, de manera tal que en el cálculo de Entropía los costos de asignación vinculados a este agente no estén considerados.

En tercer lugar, se propone modelar un problema que contemple capacidades de asignación no determinadas según el tamaño del problema, sino definidas para cada agente. Esto puede tener interés práctico para sistemas de producción en los cuales los agentes difieren en sus características técnicas para atender un número o tiempos de proceso vinculados a las tareas a ejecutar.

Por último, y vinculado con la anterior propuesta, se sugiere evaluar modificaciones y la aplicación del algoritmo a problemas en donde existan restricciones respecto a la configuración de asignación de tareas a determinados agentes. Esto es, contemplar la posibilidad en la que determinadas tareas no puedan ser ejecutadas por ciertos agentes, siguiendo la idea en la que en un contexto industrial o de servicios existan operadores (o máquinas) que no cuenten con las capacidades formativas (o técnicas) para ejecutar determinadas órdenes (o trabajos).

Referencias

Ahmed, A. y Ahmad, A. (2014). A new method for finding an optimal solution of assignment problems. *International Journal of Modern Mathematical Sciences 12(1)*: 10-15.

Amaliah, B. Widyant, M., Purwanto, D., Armand, F., Igbal, M., y Verdianto, S. (2013). A modified hungarian method for unbalanced assignment doctor problem in disaster management. *The 3rd International Workshop on Soft Computing and Disaster Control*: 61-67.

Betts, N., y Vasko, F. (2016). Solving the unbalanced assignment problem: simple is better. *American Journal of Operations Research 6*: 296-299.

Bhunia, A., Biswas, A., y Samanta, S. (2017). A genetic algorithm-based approach for unbalanced assignment problem in interval environment. *International Journal of Logistics Systems and Management 27(1)*: 62–77.

Biswas, A., Bhunia, A., y Shaikh, A. (2018). Multi-objective unbalanced assignment problem with restriction of jobs to agents via NSGA-II. *International Journal of Mathematics in Operational Research 13(1)*: 107.

Brunsch, T., Cornelissen, K., Manthey, B., Roglin, H., y Rosner, C. (2015). Smoothed analysis of the successive shortest path algorithm. *SIAM Journal on Computing 44(6):* 1798–1819.

Burkard, R., Dell'Amico, M., y Martello, S. (2009). Assignment problems. Philadelphia, USA: *SIAM, Society for Industrial and Applied Mathematics*.

Campelo, F., y Wanner, E. (2020). Sample size calculations for the experimental comparison of multiple algorithms on multiple problem instances. *Journal of Heuristics 26:* 851–883.

Chu, P., y Beasley, J. (1997). A genetic algorithm for the generalised assignment problem. *Computers and Operations Research 24(1)*: 17–23.

Ciupala, L. (2005). Scaling out-of-kilter algorithm for minimum cost flow. *Control and Cybernetics 34(4)*: 1169-1174.

Edmonds, J., y Karp, R. (1972). Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems. *Journal of the ACM 19:* 248-264.

Engquist, M. (1982). A Successive Shortest Path algorithm for the Assignment Problem. *INFOR: Information Systems and Operational Research 20:* 370-384.

Fang, M., Wang, M., y Bi, Y. (2011). Searching Best Strategies Algorithm for the No Balance Assignment Problem. *Applied Mechanics and Materials 50-51*: 386–390.

Ford, L., y Fulkerson, D. (1956). Maximal flow through a network. *Canadian Journal of Mathematics 8:* 399-404.

González-Cruz, M., y Gómez-Senent, E. (2011). An entropy-based algorithm to solve the facility layout design problem. *Robotics and Computer-Integrated Manufacturing 27 (1)*, 88–100.

Iampang, A., Boonjing, V., y Chanvarasuth, P. (2010). A cost and space efficient method for unbalanced assignment problems. *2010 IEEE International Conference on Industrial Engineering and Engineering Management*, 985-988.

Jain, E., Dahiya, K., y Verma, V. (2019). A priority based unbalanced time minimization assignment problem. *OPSEARCH 57*: 13-45.

Kabiru, S., Saidu, B., Abdul, Z., y Ali, U. (2017). An optimal assignment Schedule of staff-subject allocation. *Journal of Mathematical Finance 7*: 805-820.

Khandelwal, A. (2018) An Amalgamated Approach for solving the unbalanced assignment problem. *Malaya Journal of Matematik 6(2)*: 321-325.

Kuhn, H. (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quaterly 5*: 83-97.

Kumar, A. (2006). A modified method for solving the unbalanced assignment problems. *Applied Mathematics and Computation 176*: 76-82.

Kumar, S., Kumar, A., y Upadhyay, V. (2018). The Average Sum Method for the Unbalanced Assignment Problems. *International Journal of Mathematics Trends and Technology 55(2)*: 89-100.

Majumdar, J. y Bhunia, A. (2012). An alternative approach for unbalanced assignment problem via genetic algorithm. *Applied Mathematics and Computation 218*: 6934-6941.

Mondal, R., Ray, P., Nandi, E., Biswas, B., Sanyal, M., y Sarddar, D. (2019). Load Balancing of Unbalanced Assignment Problem With Hungarian Method. *International Journal of Ambient Computing and Intelligence 10(1)*: 46-60.

Munapo, E. (2020). Development of an accelerating hungarian method for assignment problems. *Eastern-European Journal of Enterprise Technologies 4*: 6-13.

Ning, X., y Li, P. (2018). A cross-entropy approach to the single row facility layout problem. *International Journal of Production Research 56(11)*: 3781-3794.

Pentico, D. (2007). Assignment problems: A golden anniversary survey. *European Journal of Operational Research 176(2)*: 774-793.

Rabbani, Q., Khan, A., y Quddoos, A. (2019). Modified hungarian method for unbalanced assignment problem with multiples jobs. *Applied Mathematics and Computation 361*: 493-498.

Ramesh, G., Sudha, G., y Ganesan, K. (2020). Method of finding an optimal solution for interval balanced and unbalanced assignment problem. *IOP Conference Series: Materials and Science and Engineering 912*: 062031.

Sahu, A., y Tapadar, R. (2006). Solving the assignment problem using genetic algorithm and simulated annealing. *International Journal of Applied Mathematics 36*: 762–765.

Sena Das, G., Gzara, F., y Stützle, T. (2020) A Review on Airport Gate Assignment Problems: Single versus Multi Objective Approaches. *Omega 92*: 102-146.

Silver, E. (2004). An Overview of Heuristic Solution Methods. *The Journal of the Operational Research Society 55(9)*: 936-956.

Sokkalingam, P., Ahuja, R., y Orlin, J. (2000). New polynomial-time cycle-canceling algorithms for minimum-cost flows. *Networks 36(1)*: 53–63.

Thenepalle, J., y Singamsetty, P. (2019). Lexi-search algorithm for one to many multidimensional bi-criteria unbalanced assignment problem. *International Journal of Bio-Inspired Computation 14(3)*: 151-170.

Votaw, D. y Orden, A. (1952). The personnel assignment problem. *Symposium on Linear Inequalities and Programming, Scoop 10, US Air Force*: 155-163.

Wang, Z., Pu, J., Cao, L. y Tan, J. (2015). A parallel biological optimization algorithm to solve the unbalanced assignment problem based on DNA molecular computing. *International Journal Molecular Sciences 16(10)*: 25338–25352.

Williamson, D. (2002). The primal-dual method for approximating algorithms. *Mathematical Programming 91(3)*:447-478.

Wu, L., Zhang, Z., y Zhang, J. (2019). A Hybrid Vehicle Dispatching Approach for Unified Automated Material Handling System in 300mm Semiconductor Wafer Fabrication System. *IEEE Access 7*: 174028-174041.

Yadaiah, V., y Haragopal, V. (2016). A new approach of solving single objective unbalanced assignment problem. *American Journal of Operations Research 6*: 81-89.

Anexos

Anexo I. Soluciones alcanzadas por el algoritmo para instancias de la literatura

Tabla A.1. Resultado y asignación obtenida por el Algoritmo para Kumar (2006)

Fuente. Elaboración propia

Asignación			Costo de Asignación
T_3, T_6	$\rightarrow$	A_1	**1520**
T_4, T_8	$\rightarrow$	A_2	
T_5	$\rightarrow$	A_3	
T_7	$\rightarrow$	A_4	
T_1, T_2	$\rightarrow$	A_5	

Tabla A.2. Resultado y asignación obtenida por el Algoritmo para Majumdar y Bhunia (2012)-A

Fuente. Elaboración propia

Asignación			Costo de Asignación
T_3, T_6	$\rightarrow$	A_1	**133**
T_4	$\rightarrow$	A_2	
T_5	$\rightarrow$	A_3	
T_7	$\rightarrow$	A_4	
T_1, T_2	$\rightarrow$	A_5	

Tabla A.3. Resultado y asignación obtenida por el Algoritmo para Majumdar y Bhunia (2012)-B

Fuente. Elaboración propia

Asignación			Costo de Asignación
T_2, T_5	$\rightarrow$	A_1	**66**
T_4, T_7	$\rightarrow$	A_2	
T_1, T_3	$\rightarrow$	A_3	
T_6	$\rightarrow$	A_4	
T_8	$\rightarrow$	A_5	
T_9, T_{10}	$\rightarrow$	A_6	

Tabla A.4. Resultado y asignación obtenida por el Algoritmo para Majumdar y Bhunia (2012)-C

Fuente. Elaboración propia

Asignación			Costo de Asignación
T_4	$\rightarrow$	A_1	**69**
T_6	$\rightarrow$	A_2	
T_7	$\rightarrow$	A_3	
T_3, T_8	$\rightarrow$	A_4	
T_9	$\rightarrow$	A_5	
T_1, T_5	$\rightarrow$	A_6	
T_2, T_{10}	$\rightarrow$	A_7	

Tabla A.5. Resultado y asignación obtenida por el Algoritmo para Mondal et al. (2017)

Fuente. Elaboración propia

Asignación			Costo de Asignación
T_1, T_7	$\rightarrow$	A_1	**1533**
T_3	$\rightarrow$	A_2	
T_5, T_6	$\rightarrow$	A_3	
T_2, T_4	$\rightarrow$	A_4	
T_8	$\rightarrow$	A_5	

Tabla A.6. Resultado y asignación obtenida por el Algoritmo para Khandelwal (2018)

Fuente. Elaboración propia

Asignación			Costo de Asignación
T_4, T_6	$\rightarrow$	A_1	**27**
T_2	$\rightarrow$	A_2	
T_1, T_3	$\rightarrow$	A_3	
T_5	$\rightarrow$	A_4	