UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Máster Universitario en Ingeniería de Análisis de Datos, Mejora de Procesos y Toma de Decisiones

Departamento de Estadística e Investigación Operativa Aplicadas y Calidad

RESOLUCIÓN DE UN PROBLEMA REAL DE ASIGNACIÓN DE PERSONAL A PROYECTOS DE AUDITORÍA

Rodrigo Teodoro Lima

València, Septiembre 2019

Directoras:

María Fulgencia Villa Julià y Eva Vallada Regalado

RESUMEN

En este trabajo se aplican métodos de programación de la producción para modelizar y resolver un problema real de asignación y secuenciación de proyectos de auditoría al que se hace frente en el órgano de control interno del gobierno federal brasileño. En este problema, un gestor debe determinar, a partir de un conjunto predefinido de proyectos (obligatorios y optativos), el calendario laboral anual de cada uno de los auditores de su equipo, teniendo por objetivo maximizar la suma de beneficios asociados a la realización de los proyectos de naturaleza optativa. Los proyectos poseen restricciones de *release times* y *deadlines,* mientras los auditores presentan restricciones de indisponibilidades deterministas. Se propone un modelo de programación entera mixta (MIP) para la comprensión matemática del problema y también para intentar resolverlo óptimamente, pero no se encuentran soluciones óptimas exactas en un tiempo de computación razonable para los propósitos de la investigación. Solamente para las instancias pequeñas y medianas (hasta 10 auditores y 110 proyectos) aleatoriamente generadas para el problema se consigue obtener buenas soluciones factibles. Para las instancias de tamaño más grande, el modelo matemático solo es capaz de encontrar una cota superior relativa a la relajación lineal del problema. De este modo, se desarrollan dos algoritmos heurísticos constructivos con el lenguaje de programación *Julia*, los cuales logran encontrar, en un tiempo computacional muy pequeño, soluciones de alta calidad para el conjunto total de instancias probadas. Se hace una modificación en el segundo heurístico con el objetivo de incrementar su rendimiento y, al final, mediante un análisis de variancia (ANOVA), se verifica si las medias de los valores de las soluciones encontradas por cada uno de los heurísticos presentan diferencias estadísticamente significativas.

Palabras clave: asignación y secuenciación de proyectos de auditoría, modelizado matemático de problemas reales, programación entera mixta, algoritmos heurísticos constructivos, lenguaje de programación Julia.

ABSTRACT

In this work, production scheduling methods are applied to model and solve a real problem of audit projects assignment and scheduling, which is faced in the internal control body of the Brazilian federal government. In this problem, a manager must determine, from a predefined set of projects (required and optional ones), the annual work calendar of each auditor of his team, with the objective of maximizing the sum of benefits associated with the execution of the optional projects. There are restrictions of release times and deadlines concerning the projects, while auditors have restrictions related to deterministic unavailabilities. A mixed integer linear programming model (MIP) is proposed for the mathematical understanding of the problem and still for trying to solve it optimally, but no exact optimal solutions are found in a reasonable computational time for the purposes of the investigation. Only for small and medium-sized instances (up to 10 auditors and 110 projects) randomly generated for the problem, good feasible solutions can be obtained. For larger instances, the mathematical model is only able to find a best bound related to the linear relaxation of the problem. Then, two constructive heuristic algorithms are developed with Julia programming language, which succeed to find, in a very small computational time, high quality solutions for all randomly generated instances of the problem. A modification is introduced in the second heuristic with the aim of increase its performance and, finally, through a variance analysis (ANOVA), it is verified if the means of the solutions found by each heuristic differ amongst themselves in a statistically significant way.

Keywords: audit projects assignment and scheduling, real problems mathematical modelling, mixed integer programming, constructive heuristic algorithms, Julia programming language.

ÍNDICE

LISTADO DE SIGLAS

AHP – *Analytic Hierarchy Process*

ANOVA – *Analysis of Variance*

CGU – *Controladoria-Geral da União* (Contraloría-General de la Unión)

CP – *Constraint Programming*

CRG – *Corregedoria-Geral da União* (Corregiduría-General de la Unión)

HSD (Tukey) – *Honestly Significant Difference*

IBM – *International Business Machines*

IIA – *Institute of Internal Auditors*

MIP – *Mixed Integer Programming*

MIT – *Massachussets Institute of Technology*

NAC – *Núcleo de Ações de Controle* (Núcleo de Acciones de Control)

OGU – *Ouvidoria-Geral da União* (Oidoría-General de la Unión)

PSO – *Particle Swarm Optimization*

RCPSP – *Resource-Constrained Project Scheduling Problem*

RPD – *Relative Percentage Deviation*

SCC – *Secretaria de Combate à Corrupção* (Secretaría de Combate a la Corrupción)

SFC – *Secretaria Federal de Controle Interno* (Secretaría Federal de Control Interno)

STPC – *Secretaria de Transparência e Prevenção da Corrupção* (Secretaría de Transparencia y Prevención de la Corrupción)

TCU – *Tribunal de Contas da União* (Tribunal de Cuentas de la Unión)

UPMSP (o UPMS) – *Unrelated Parallel Machines Scheduling Problem*

VBA – *Visual Basic for Applications*

1. Introducción

La programación (o secuenciación) de la producción – del inglés, *scheduling* – es, según Pinedo (2016), un proceso de toma de decisiones que se utiliza regularmente en muchas industrias manufactureras y de servicios, ocupándose de la asignación de recursos para tareas en períodos de tiempos determinados, con la finalidad de optimizar uno o más objetivos. Es una de las áreas de la investigación operativa más ampliamente estudiadas, en gran parte debido a su aplicación en una rica variedad de problemas. Su origen se remonta a principios del siglo XX (en 1916) con los pioneros trabajos del ingeniero mecánico norteamericano Henry Gantt, sin embargo, solo a mediados de la década de los 50 comienza a aparecer una colección sostenida de publicaciones sobre programación de la producción (Potts y Strusevich, 2009).

Uno de los posibles problemas que puede ser abordado mediante la investigación operativa es la planificación y programación de los trabajos de la unidad de auditoría interna de una organización o de una empresa de auditoría externa. Si se consideran los trabajos como actividades (tareas) de un proyecto, se tiene típicamente un *Resource-Constrained Project Scheduling Problem* (RCPSP), pero este no es el caso de la presente investigación, pues los trabajos a programar son los propios proyectos, es decir, son los todos y no las partes. Por lo tanto, como se detallará en los capítulos 2 y 3, en este trabajo se considera el problema de programación de los proyectos de una unidad de auditoría interna para el periodo de un año, lo cual puede ser visto como una variante del problema de programación de máquinas paralelas no relacionadas – del inglés, *Unrelated Parallel Machines Scheduling Problem* (UPMSP o UPMS) – que posee una función objetivo distinta de las que suelen ser utilizadas, así como restricciones específicas de *release times* de los trabajos y de indisponibilidades programadas de los auditores.

1.1. Motivación

La Contraloría-General de la Unión (CGU) es el órgano de control interno del Poder Ejecutivo Federal de Brasil, responsable de realizar actividades relacionadas con la defensa del patrimonio público y el incremento de la transparencia de la gestión, por medio de acciones de auditoría pública, corrección disciplinar, prevención y combate a la corrupción.

La CGU está estructurada en cinco unidades finalistas, actuando de forma articulada: Secretaría de Transparencia y Prevención de la Corrupción (STPC), Secretaría Federal de Control Interno (SFC), Corregiduría-General de la Unión (CRG), Secretaría de Combate a la Corrupción (SCC) y Oidoría-General de la Unión (OGU). El órgano central está ubicado en la capital federal del país, Brasilia, pero en la capital de cada Unidad Federativa (Estado) existe también una representación regional de la CGU.

La SFC, donde actualmente trabaja el autor de este trabajo, es la unidad organizacional responsable de ejecutar las acciones de fiscalización y evaluación de los programas de políticas públicas del gobierno federal brasileño. En esta unidad – y también en los núcleos de acciones de control (NAC) de las CGU Regionales – es común la necesidad de planificación, por los gestores, de los trabajos de auditoría a realizar a lo largo de un periodo específico (normalmente de un año, pero pudiendo ser subdivido en semestres o trimestres). Esto está en conformidad con las normas internacionales para el ejercicio profesional de la auditoría interna, del Instituto de Auditores Internos (IIA), más específicamente, con la norma "2010 – Planificación" (*The Institute of Internal Auditors*, 2017), la cual enuncia que se debe establecer un plan de auditoría basado en los riesgos, a fin de determinar las prioridades de la actividad de auditoría interna, siendo que dichos planes deberán ser consistentes con las metas de la organización.

La SFC está estructurada en seis direcciones y un gabinete de asesoramiento, siendo que en cada dirección trabajan entre 30 y 70 auditores. Las direcciones son subdividas en coordinaciones-generales, que suelen comprender entre 10 y 20 auditores y que, a su vez, son compuestas por 2 o 3 divisiones de auditoría, trabajando en cada división entre 4 y 8 auditores aproximadamente. En las CGU Regionales suele existir un escenario similar, dependiendo del tamaño de la representación de la CGU en ese Estado. En las Regionales medianas, un NAC (similar a una división de auditoría del órgano central) suele comprender entre 5 y 10 auditores.

Los trabajos en la SFC y en toda la CGU normalmente tienen la naturaleza de proyectos, con las siguientes características: i) se realizan en un periodo limitado; ii) poseen alcance y objetivos muy concretos; iii) dependen de la asignación de recursos (de personal, financieros, etc.) para su realización; y iv) tienen como un resultado un producto de

características únicas, lo cual, en el caso específico de los proyectos de auditoría, es el informe final de la auditoría.

En este trabajo académico se ha buscado construir un modelo matemático capaz de ayudar a los gestores de unidades de auditoría de la CGU en la toma de decisiones relativa a la planificación y programación de los proyectos de su equipo para el periodo de un año. Con el término "unidad de auditoría" o "equipo de auditoría" se pretende referenciar, de modo más usual, un núcleo de acción de control (NAC) de una CGU Regional o una división de auditoría del órgano central. Esta estructura administrativa suele estar formada por equipos de trabajo de entre 5 y 10 auditores. Es dentro de esta estructura donde se aplicaría la metodología de la presente investigación. No obstante, consideramos que los modelos aquí desarrollados podrían ser aplicados en una unidad de trabajo un poco más amplia y con más auditores, como una coordinación-general o dirección del órgano central o una CGU Regional entera.

Se habla específicamente de proyectos de auditoría debido al hecho de ser la auditoría interna el campo de trabajo del autor, pero los trabajos que se desea programar podrían también, por supuesto, corresponder a las otras unidades organizacionales de la CGU ya mencionadas.

En la SFC existen, todos los años, proyectos de auditoría de naturaleza obligatoria, destacándose, de entre estos, los relativos a las auditorías de rendición de cuentas de los órganos y entidades de la Administración Pública Federal, las cuales son demandadas anualmente por el Tribunal de Cuentas de la Unión (TCU), órgano de control externo de los tres poderes de la República brasileña que está vinculado al Congreso Nacional.

Además de la realización de los proyectos obligatorios, los gestores de equipos deben seleccionar otros proyectos de auditoría de naturaleza optativa en función de su relevancia para los resultados pretendidos por la alta dirección del órgano. No existe actualmente una metodología general definida para esta selección, es decir, se realiza de modo *ad hoc*: cada gestor, en cada unidad administrativa, realiza la selección de los proyectos de auditoría optativos y su respectiva asignación a los miembros de su equipo de modo intuitivo, es decir, de acuerdo con su experiencia, preferencias e *insights*. Con esto se quiere enfatizar que la creación de un calendario anual de trabajos para cada auditor no sigue un método plenamente

estructurado, estando sujeta a fallos de dimensionamiento, tanto por exceso como por defecto.

Por tanto, partiendo del contexto explicado anteriormente, este trabajo propone la definición de una metodología estructurada de selección de proyectos optativos para la ejecución en el periodo de un año, de modo que se asignen a los auditores disponibles, se secuencien en el tiempo de acuerdo con las características generales y las restricciones operativas que serán presentadas en el capítulo 2 y teniendo por objetivo maximizar la suma de beneficios obtenida por los proyectos optativos seleccionados.

1.2. Estructura y objetivos

La estructura de este trabajo ha sido determinada teniendo en cuenta los objetivos en distintos capítulos, tal y como se presenta a continuación.

En el capítulo 2 (descripción del problema), se describe el problema real de asignación y secuenciación de proyectos de auditoría que ha motivado la investigación emprendida en este trabajo.

En el capítulo 3 (revisión bibliográfica), se presenta un breve resumen de los trabajos previos que han aplicado los métodos de programación de la producción en la resolución de problemas relacionados con la asignación y secuenciación de proyectos de auditoría. También se hace un análisis de los trabajos de secuenciación de máquinas únicas o paralelas (idénticas o no relacionadas) que han considerado las restricciones de *release times* o de indisponibilidades programadas, las cuales son importantes para la resolución del problema tratado en este trabajo.

En el capítulo 4 (simulación y modelización), se expone, en la primera sección, el diseño de una metodología de colección, organización y simulación de los datos de entrada necesarios para la resolución del problema planteado en el capítulo 2. Esta metodología ha servido para la generación del conjunto de instancias de prueba empleado en los experimentos computacionales del capítulo 6. En la segunda sección, se presenta un modelo de programación entera mixta – del inglés, *mixed integer programming* (MIP) – que es capaz de incorporar todas las características del problema, posibilitando, así, su resolución exacta por medio de algoritmos de optimización matemática ampliamente conocidos, como es el caso del método *branch and bound*.

En el capítulo 5 (algoritmos heurísticos), se explica y detalla el diseño de algoritmos heurísticos constructivos que han logrado encontrar buenas soluciones con un tiempo de procesamiento computacional muy inferior al utilizado por el modelo MIP presentado en el capítulo anterior.

En el capítulo 6 (experimentos computacionales), se resumen los principales resultados encontrados, tanto por el modelo MIP como por los algoritmos heurísticos, para la resolución exhaustiva de un conjunto de instancias de prueba generadas de modo aleatorio. Además, se realiza un análisis de varianza (ANOVA) de los cinco algoritmos heurísticos experimentados para verificar la existencia de diferencias estadísticamente significativas entre las medias obtenidas por cada uno de ellos para la desviación relativa porcentual – respecto a una cota superior determinada por el *solver* del CPLEX en la resolución del modelo MIP tras un *timeout* de 30 minutos – del valor encontrado para la función objetivo del problema.

En el capítulo 7 (conclusiones y futuras líneas de investigación), como el propio título ya lo indica muy claramente, se exponen las principales conclusiones del presente trabajo, así como posibles líneas futuras de investigación que no han sido contempladas en ello por limitaciones de tiempo y alcance.

En el apartado "Anexos" han sido listados algunos ficheros que sirven para demostrar, aunque apenas parcialmente, la generación de los datos, el desarrollo de los modelos, la resolución de las instancias del problema y el análisis de los resultados.

2. Descripción del problema

Como ya se ha introducido en la motivación del trabajo, los gestores de equipos de auditoría – divisiones de auditoría de la Secretaría Federal de Control Interno del órgano central de la CGU o núcleos de acciones de control de las CGU Regionales – deben, para cada ejercicio anual, realizar la planificación y programación de los proyectos de auditoría a realizar en ese periodo.

Los proyectos pueden ser de dos tipos: obligatorios u optativos. Los proyectos de naturaleza obligatoria, como su denominación ya indica, deben realizarse obligatoriamente y, por lo tanto, el gestor no tiene que tomar ningún tipo de decisión al respecto. Además de estos proyectos, cada gestor dispone de un conjunto previamente determinado de proyectos

optativos, de entre los que tiene que decidir cuáles ejecutar. En general, se seleccionan aquellos proyectos que proporcionan resultados más relevantes tanto para la organización como para la sociedad brasileña, la cual es el cliente final de los trabajos que entrega la CGU.

Respecto a los proyectos optativos, el equipo al que se asignarán dichos proyectos siempre tendrá horas disponibles para ejecutarlos. La cantidad de proyectos optativos seleccionados para ejecución es importante, pero no se debería medir el éxito del equipo de auditoría solamente por este criterio. Se debe también tener en consideración el nivel de importancia que tiene cada proyecto de auditoría, según su grado de alineamiento con el plan táctico de la organización, su capacidad de generar resultados financieros y no financieros relevantes para la Administración Pública del país y otros criterios que puedan existir dentro de cada unidad organizacional.

Además, los proyectos optativos seleccionados deben ser asignados a cada auditor para completar su calendario de actividades durante todo el periodo sin excederlo y también, por supuesto, sin infrautilizarlo. Es decir, se realiza, a la vez, la asignación del proyecto a un auditor y su secuenciación en la ventana de tiempo disponible.

Esta asignación y secuenciación conjunta, la cual denominaremos secuenciación (o programación), tiene por objetivo maximizar la suma de beneficios obtenidos con la realización de los proyectos optativos seleccionados. Cualquier proyecto (obligatorio u optativo) presenta las siguientes características:

1. Se considera que cada proyecto de auditoría es realizado solamente por un único auditor. En la realidad no es lo que siempre ocurre, pero se ha considerado esta premisa para simplificar el problema.
2. Algunos auditores no son aptos para realizar determinados proyectos (por inexperiencia, cuestiones legales, decisión del gestor, etc.).
3. Los tiempos de ejecución de los proyectos para cada auditor son distintos, es decir, se considera la eficiencia del auditor en la realización del tipo de proyecto a que se refiere.
4. No todos los proyectos están disponibles al inicio del periodo, es decir, el primer día del año. Los proyectos poseen distintas fechas estimadas para su disponibilidad inicial, ya que ésta depende de factores internos y externos a la unidad de auditoría. Estos tiempos se denotan como *release times*.

5. Los proyectos obligatorios poseen fechas de entrega específicas, las cuales deben ser estrictamente respetadas, es decir, no se admite la programación de trabajos con retraso. Estos tiempos se denotan como *deadlines*. Para los proyectos optativos se considera el *deadline* como el último día del año por programar.
6. Se considera que una vez que un auditor inicia la ejecución de un proyecto de auditoría debe seguir con él hasta su término, es decir, no se debe cambiar a otro proyecto sin que haya completado al que estaba asignado inicialmente.
7. Los auditores no están disponibles en todos los días útiles del calendario anual, ya que existe una programación previa de vacaciones y periodos de licencia para la realización de acciones de capacitación (u otras finalidades). En este trabajo se consideran solo periodos de indisponibilidad que puedan ser determinados con anterioridad a la programación de los trabajos, o sea, las indisponibilidades aleatorias que puedan ocurrir posteriormente durante el año laboral no son tomadas en cuenta. Se considera que la cantidad máxima de periodos de indisponibilidad para cada auditor durante el año es de 3 y la cantidad mínima es de 1.
8. Se considera que, una vez que un auditor inicia la realización de un proyecto, si, por algún motivo, la ejecución debe ser interrumpida por indisponibilidad del auditor, puede retomarla al final de la interrupción sin aumento del tiempo útil de procesamiento del proyecto por el auditor (solo aumenta el tiempo total de ejecución, que es la suma del tiempo útil de procesamiento y el tiempo de indisponibilidad). Es decir, se considera el *resumable case* (*preemption allowed*), de acuerdo con la definición de Lee (1996).
9. Se considera que un proyecto de auditoría solo puede ser interrumpido por un único periodo de indisponibilidad. De este modo, los datos de las instancias del problema deben garantizar que la ejecución de un proyecto nunca se alarga por más de un periodo de indisponibilidad.

El problema descrito en este trabajo se puede enfocar como un problema de máquinas en paralelo no relacionadas, si se considera cada auditor como una máquina y cada proyecto de auditoría como un trabajo. En este punto es necesario recordar que en el problema de máquinas en paralelo no relacionadas existe un conjunto $N = \{1, \ldots, n\}$ de n trabajos que deben ser procesados por exactamente una máquina de un conjunto $M = \{1, \ldots, m\}$ de m

máquinas paralelas (Vallada y Ruiz, 2011). Cada trabajo exige un determinado tiempo de procesamiento y las máquinas son consideradas no relacionadas cuando el tiempo de procesamiento de los trabajos depende de la máquina a la cual son asignados. Además, cuando la función objetivo del problema se refiere a la minimización del tiempo máximo de finalización de una secuenciación, criterio conocido como *makespan* o *Cmax*, tenemos el problema *R* | *Cmax*, según la notación de Pinedo (2016).

Volviendo al comentario acerca del posible enfoque del problema, las "máquinas" (auditores) serían no relacionadas porque el tiempo de procesamiento de los proyectos depende del auditor al cual son asignados. Sin embargo, este problema no se trata de un *R* | *Cmax* sencillo, ya que existen otras restricciones y la función objetivo es distinta.

Mientras en el problema *R* | *Cmax* – ampliamente estudiado en la literatura académica – la función objetivo consiste en minimizar el tiempo máximo de finalización de los trabajos, en el problema de este trabajo, la función objetivo consiste en maximizar la suma de beneficios asociados a la realización de los proyectos optativos seleccionados.

Además, existen restricciones relativas a los *release times* y *deadlines* de los proyectos, así como indisponibilidades temporales por parte de los auditores. El problema que se aborda en este trabajo no se ha tratado previamente en la literatura académica, pero se ha podido identificar problemas con algunas similitudes. En la revisión bibliográfica realizada se han recogido diversos artículos que reúnen una u otra característica del problema aquí estudiado. Se ha hecho un aprovechamiento de la estructura común de estos problemas ya estudiados previamente para, a partir de ahí, desarrollar por cuenta propia las modificaciones y adaptaciones necesarias para la modelización matemática de la estructura total del problema real de secuenciación tratado en este trabajo.

3. Revisión bibliográfica

En el artículo de Mohamed (2015) se presenta un *survey* exhaustivo de los modelos desarrollados para la planificación y programación de auditorías. De acuerdo con este autor, existen dos enfoques principales en la literatura académica: (1) el relativo a la identificación de la frecuencia óptima de auditorías – es decir, el tiempo transcurrido óptimo (o el número óptimo de transacciones) después del cual una auditoría debería ser realizada – y (2) el relativo a la determinación de la asignación óptima de los recursos de auditoría. Respecto al

segundo enfoque – en el que se clasifica este trabajo – la investigación pionera ha sido la de Summer (1972), sucedida por otros trabajos como los de Balachandran y Zoltners (1981), Chan y Dodin (1986), Dodin y Chan (1991) y Dodin, Elimam y Rolland (1998). Sin embargo, se debe destacar que el artículo de Chan y Dodin (1986) ha sido el primero en utilizar un modelo de programación entera para encontrar valores óptimos en las programaciones de auditorías, pues las publicaciones anteriores no han tratado de las programaciones propiamente dichas, sino apenas de *loadings* (sin considerar los tiempos de inicio y fin de las tareas).

En los tres artículos de Dodin citados se han utilizado las técnicas de gestión de proyectos para modelizar el problema de programación de las auditorías, ya que lo ha planteado a nivel de actividades (o tareas), entre las cuales suelen existir relaciones de precedencia. Sin embargo, en la presente investigación se enfocan los trabajos en un nivel superior, es decir, se consideran los proyectos de auditoría como un todo a programar, no como una de las varias tareas que se interrelacionan para componer un conjunto programable. Es por esta razón que no se modeliza el problema de este trabajo como un *Resource-Constrained Project Scheduling Problem* (RCPSP).

Por lo expuesto y como ya se ha introducido en el capítulo anterior, el problema planteado en este trabajo puede ser visto como una versión muy peculiar del problema de máquinas paralelas no relacionadas, con restricciones deterministas de disponibilidad, *release times* y *deadlines* y una función objetivo muy específica, que consiste en maximizar la suma de beneficios obtenidos con la selección de proyectos optativos en la programación del calendario laboral anual de los auditores de una unidad de auditoría de la CGU.

A partir de la investigación bibliográfica emprendida, no se ha identificado ningún trabajo académico que haya abordado el problema de máquinas paralelas no relacionadas con restricciones de disponibilidad de las máquinas, *release times* y *deadlines*. Se han encontrado trabajos que comprenden alguna(s) característica(s) del problema aquí estudiado, pero no todas. Hay artículos, por ejemplo, que tratan de indisponibilidades, pero solamente para una única máquina y sin considerar *release times*, como es el caso de Wang, Sun y Chu (2005), Low, Li y Wu (2016) y Bülbül, Kedad-Sidhoum y Şen (2017). Otros enfocan el problema de máquinas paralelas (idénticas o no relacionadas) con indisponibilidades y con todos los proyectos disponibles en el mismo instante (sin *release times*), de los cuales

podemos citar: Sevindik (2006), Kurt (2012), Hashemian, Diallo y Vizvári (2014), Gedik, Rainwater, Nachtmann y Pohl (2016) y Beaton, Diallo y Gunn (2016). Hay también los relativos al problema de máquinas paralelas no relacionadas con *release times*, pero sin considerar indisponibilidades, como Lin (2013) y Avdeenko y Mesentsev (2016).

Por claridad, se ha partido primero de los problemas con máquinas paralelas que consideran *release times*, para después comentar los problemas con máquinas paralelas que se centran en la restricción de indisponibilidades, resaltando que este segundo caso ha sido más discutido e investigado por la comunidad académica que el primero.

Empezando por el enfoque de los *release times*, se destaca el artículo de Lin (2013) sobre el problema de máquinas paralelas no relacionadas con distintos *release times* para los trabajos, referido en la literatura como $R \mid r_j \mid Cmax$, según la notación de Pinedo (2016). Este problema es *NP-hard*. Según el autor del artículo, ninguna investigación había sido publicada hasta entonces que desarrollara un algoritmo eficiente para minimizar el *makespan* para máquinas paralelas no relacionadas con *release times*. Se propone un heurístico y un algoritmo del tipo *particle swarm optimization* (PSO) para resolver el problema.

También abordan el mismo problema los autores Avdeenko y Mesentsev (2016), haciendo hincapié en que la bibliografía sobre este problema es muy escasa y proponiendo un enfoque de resolución exacta basado en un método de programación dinámica. Se han conseguido resultados eficaces para problemas grandes de 100 trabajos y 30 máquinas.

Pasando al enfoque de las indisponibilidades de las máquinas, Sevindik (2006) ha estudiado el problema de secuenciación de máquinas paralelas sujeto a restricciones de indisponibilidad en cada máquina, tanto para el objetivo de minimización del tiempo total de finalización como para lo de minimización del tiempo máximo de finalización. En ambos casos el problema es *NP-hard*. En realidad, cuando se consideran restricciones de indisponibilidad, incluso los problemas de secuenciación para una única máquina son clasificados como *NP-hard*. En el problema de Sevindik (2006) todos los trabajos tienen el mismo *release time*, los tiempos de procesamiento son conocidos y deterministas (no cambian según la máquina a la cual son asignados) y no se admite interrupción. Existe solamente un periodo de indisponibilidad para cada máquina, siendo conocidos sus tiempos de inicio y duración. Además, los tiempos de indisponibilidad de las máquinas no se pueden solapar. En este trabajo se ha desarrollado un procedimiento exacto del tipo *branch and*

bound y tres algoritmos heurísticos (un constructivo, uno de búsqueda local y un *simmulated annealing*) para la obtención de soluciones aproximadas para el problema con el objetivo de minimización del tiempo total de finalización.

Kurt (2012) considera el problema de secuenciar *n* trabajos independientes en *m* máquinas paralelas no relacionadas sujetas a restricciones de indisponibilidad y elegibilidad, con el objetivo de minimizar el *makespan*. Son analizados tanto los trabajos que se pueden continuar tras una interrupción (*resumable jobs*) como los que no (*non-resumable*). Se desarrollan tanto modelos matemáticos exactos como un algoritmo heurístico de múltiples fases para la obtención de soluciones casi-óptimas para ambos tipos de trabajos, con múltiples periodos de indisponibilidad. Se concluye que los modelos de programación lineal entera no son capaces de resolver instancias grandes del problema, mientras el algoritmo heurístico ha demostrado ser capaz de resolver óptimamente instancias pequeñas y medianas, así como de obtener buenas soluciones para las instancias grandes, en un pequeño tiempo de computación.

Kaabi y Harrath (2014) han realizado un *survey* del problema de secuenciación de máquinas paralelas bajo restricciones de indisponibilidad. Se han investigado las tres variantes del problema de máquinas paralelas: idénticas (todas las máquinas tienen las mismas velocidades de procesamiento para los trabajos), uniformes (las máquinas tienen velocidades de procesamiento distintas, pero que se diferencian según una razón uniforme) y no relacionadas (los tiempos de procesamiento de los trabajos por las máquinas son distintos, no siguiendo ninguna regla específica). Se destaca al final que los problemas de máquinas paralelas no relacionadas con restricciones de indisponibilidad han sido muy poco estudiados y, por lo tanto, son raros los resultados publicados.

En Hashemian, Diallo y Vizvári (2014) se aborda el problema de minimización del *makespan* para la secuenciación de máquinas paralelas idénticas con múltiples periodos de indisponibilidad y con trabajos del tipo *resumable*. Sus principales contribuciones son una nueva formulación MIP para el problema denotado en la literatura como $Pm \mid r - a_{m,q} \mid Cmax$ (la primera formulación para la versión general del problema, según los autores) y un algoritmo de enumeración exacta implícita para resolverlo. Los autores destacan que en el estado del arte había un número muy limitado de métodos y algoritmos para hacer frente al problema considerado y solamente para instancias pequeñas y periodos de indisponibilidad

restrictivos a un subconjunto de máquinas. El modelo MIP ha sido capaz de resolver instancias pequeñas y medianas del problema en CPLEX. Para las instancias más grandes, ha sido desarrollado un algoritmo de enumeración implícita que hace uso del orden lexicográfico y de lemas que reducen su espacio de búsqueda. Varios experimentos han demostrado el incremento de la eficacia en la resolución del problema tanto por el modelo MIP como por el algoritmo de enumeración.

En Gedik, Rainwater, Nachtmann y Pohl (2016) se propone un modelo de *constraint programming* (CP) y algoritmos basados en la lógica de descomposición de Bender para tomarse las mejores decisiones en la secuenciación de trabajos no idénticos con intervalos de indisponibilidad y tiempos de *setup* dependientes de la secuencia en máquinas paralelas no relacionadas con un horizonte de planificación fijo. Se demuestra que el modelo CP obtiene soluciones factibles de buena calidad, pero no es capaz de conseguir la solución óptima para la mayoría de las instancias. Por otro lado, los dos algoritmos basados en la lógica de descomposición de Bender son capaces de obtener soluciones casi-óptimas para 86 de las 90 instancias examinadas y de proporcionar una solución factible para las demás.

En Beaton, Diallo y Gunn (2016) se considera el problema de minimización del *makespan* para la secuenciación de máquinas paralelas idénticas con múltiples periodos de indisponibilidad para tres tipos de trabajos: *non-resumable*, *semi-resumable* y *resumable jobs*. Se destaca que en el estado del arte las formulaciones son separadas para los tres tipos de trabajos mencionados. Se propone un modelo MIP que es el primero que integra los tres casos en una única formulación, sin imponer restricciones al número de máquinas o de periodos de indisponibilidad. La formulación MIP de Beaton et al (2016) ha resuelto óptimamente las instancias pequeñas del problema con la utilización del CPLEX en un tiempo razonable. Para las instancias grandes se han utilizado cuatro métodos heurísticos para obtener buenas soluciones, con significativa reducción del tiempo de procesamiento computacional. Sin embargo, debido a la elevada complejidad del problema, incluso algunos heurísticos resultaron lentos en la resolución de las instancias grandes.

La tabla 1 muestra un resumen de los trabajos citados en este capítulo que poseen al menos una de las características estructurales del problema real investigado en este trabajo.

Autor	Máquinas	*Release Times*	Indispon.	Función Objetivo
Wang, Sun y Chu (2005)	Única	No	Si	Minimizar la suma ponderada de los tiempos de finalización.
Low, Li y Wu (2016)	Única	No	Si	Minimizar el adelanto y tardanza totales.
Bülbül, Kedad-Sidhoum y Şen (2017)	Única	No	Si	Minimizar la desviación total de los tiempos máximos de finalización de los trabajos respecto a una fecha de entrega común.
Lin (2013)	Paralelas no relacionadas	Si	No	Minimizar el *makespan*.
Avdeenko y Mesentsev (2016)	Paralelas no relacionadas	Si	No	Minimizar el *makespan*.
Sevindik (2006)	Paralelas idénticas	No	Si	Minimizar el *makespan*. Minimizar el tiempo total de finalización.
Kurt (2012)	Paralelas no relacionadas	No	Si	Minimizar el *makespan*.
Hashemian, Diallo y Vizvári (2014)	Paralelas idénticas	No	Si	Minimizar el *makespan*.
Gedik, Rainwater, Nachtmann y Pohl (2016)	Paralelas no relacionadas	No	Si	Maximizar la ganancia total con la ejecución de los trabajos.
Beaton, Diallo y Gunn (2016)	Paralelas idénticas	No	Si	Minimizar el *makespan*.

Tabla 1 – Resumen de la revisión bibliográfica acerca de los trabajos que abordan *release times* o indisponibilidades en problemas de máquinas paralelas o de una única máquina.

Se puede comprobar que aún no ha sido publicado algún artículo que haya abordado simultáneamente las restricciones de *release times* e indisponibilidades programadas para el problema de máquinas paralelas, ya sean ellas idénticas o no relacionadas.

Como ya se ha comentado, los problemas de máquinas paralelas no relacionadas con el objetivo de minimizar el *makespan* son *NP-hard*, sea con la restricción de *release times* o con la de indisponibilidades, así como lo es el problema *R* | *Cmax* original, sin tales restricciones. Con esto se quiere decir que es poco probable (a menos que se probara que P = NP) que exista un algoritmo de tiempo polinómico capaz de encontrar una solución óptima exacta para estos problemas (Fanjul Peyró, 2011). El problema de este trabajo posee una función objetivo muy específica, distinta de la minimización del *Cmax*, por lo tanto, no se

puede afirmar, sin comprobarlo formalmente, que se trate de un problema del tipo *NP-hard*, sin embargo, todo indica que es muy probable que sí.

Por lo expuesto, es esperado que se requiera un tiempo de computación considerable para encontrarse una solución óptima para el problema abordado en este trabajo (Potts y Strusevich, 2009), lo que será demostrado por los experimentos computacionales relativos al rendimiento del modelo MIP propuesto en el próximo capítulo. Según Potts y Strusevich (2009), para la mayoría de las necesidades prácticas es suficiente adoptar una solución aproximada cuando se sepa, por anticipación, que esta se encuentre relativamente cercana al valor óptimo.

De entre los algoritmos responsables por obtener soluciones aproximadas para los problemas del tipo *NP-hard*, se distinguen dos tipos: los de aproximación (del inglés, *approximation*) y los heurísticos. El rendimiento de un algoritmo de aproximación es evaluado por el análisis del peor caso o por un análisis probabilístico de su comportamiento. Respecto a los heurísticos, no hay una evaluación analítica, es decir, su calidad es determinada únicamente por su rendimiento en la resolución de instancias conocidas (*benchmarks*) o aleatoriamente generadas (Potts y Strusevich, 2009). En este trabajo, se ha optado por la creación de algoritmos heurísticos para la obtención de soluciones relativamente próximas del valor óptimo, los cuales son presentados en el capítulo 5.

4. Simulación y modelización

Este capítulo se inicia por la presentación, en la primera sección, de la metodología construida para recoger los datos de entrada necesarios para la resolución del problema planteado en este trabajo. En la segunda sección, se formula un modelo de programación entera mixta para su resolución exacta.

4.1. Simulación de los datos de entrada con Excel

Para que se pueda determinar el calendario laboral anual de los auditores de una unidad, son necesarios varios datos de entrada, pero estos no se encuentran plenamente disponibles en la CGU. Como ya ha sido mencionado antes, no existe actualmente en esta organización una metodología estructurada para la programación de los proyectos de auditoría y tampoco se tiene una colección suficiente de datos para alimentar un modelo con

esta finalidad. Por ello, se han diseñado inicialmente, por medio del software Microsoft Excel y de su lenguaje de programación *Visual Basic for Applications* (VBA), algunas herramientas que sirven para generar los datos de entrada requeridos por los modelos que se construyen en este trabajo.

Al principio, se ha pensado en diseñar las hojas Excel de tal manera que el gestor de auditoría pudiera entrar con algunas de las informaciones necesarias de modo manual, para que se pudieran colectar los datos reales de su unidad de trabajo en la CGU. Sin embargo, para los propósitos de este trabajo académico, era necesaria una herramienta que pudiera simular, de la manera más automática posible, los datos requeridos por los modelos para la resolución del problema presentado, pues los tests de rendimiento deben ser ejecutados con numerosas instancias. Por lo tanto, los ficheros de simulación, que a continuación se describirán, han sido creados buscando atender tanto al requisito de automatización como al de proximidad de los datos reales.

Se ha considerado el primer fichero empleado en los tests computacionales, relativo a la instancia más pequeña del problema (5 auditores y 70 proyectos), para explicar, paso a paso, los principales conceptos y métodos utilizados para diseñar la herramienta de simulación de los datos[1]. Los ficheros de las demás instancias siguen la misma lógica, solo cambiando las cantidades de los datos requeridos.

4.1.1. Calendario de disponibilidad para los auditores y estimación de esfuerzo para los proyectos

Se ha partido inicialmente de un calendario de disponibilidad de un equipo de 5 auditores para el periodo de un año, habiendo sido utilizado el calendario laboral brasileño. Son tomados en consideración apenas los días útiles para el contaje de unidades de tiempo, lo que se puede observar en la hoja "Calendario" del respectivo fichero Excel. De este modo, se tiene 247 días laborales para el año considerado, siendo estos identificados entre 0 y 246. Para cada columna de disponibilidad del auditor, se define el valor 1 para la representación de disponibilidad y 0 en el caso contrario, es decir, el auditor se encuentra indisponible (por razón de vacaciones o alguna licencia). Como ya se ha mencionado anteriormente, se

[1] Este fichero, denominado *"05Aud_70P_1.xlsm"*, ha sido anexado al presente trabajo.

considera que cada auditor puede tener un máximo de 3 periodos de indisponibilidad previamente programados.

Los datos acerca de los periodos de indisponibilidad de los auditores han sido representados de forma matricial con el objetivo de facilitar su posterior importación y tratamiento por otros *softwares*. Se han creado dos matrices, una para los tiempos de inicio de los periodos de indisponibilidad y otra para los tiempos de fin. Cada línea de la matriz se refiere a un auditor *i* y cada columna se refiere a un intervalo de indisponibilidad *q*, conforme será detallado más adelante. Celdas rellenadas con el valor -1 significan que no hay indisponibilidad en aquel intervalo. En las tablas siguientes podemos visualizar esta representación:

Auditor ↓	**B1**	**B2**	**B3**
1	1	230	-1
2	1	159	225
3	15	164	-1
4	119	230	-1
5	1	-1	-1

Tabla 2 – Inicio de los periodos de indisponibilidad para 5 auditores.

Auditor ↓	**F1**	**F2**	**F3**
1	21	247	-1
2	5	168	234
3	29	172	-1
4	123	247	-1
5	41	-1	-1

Tabla 3 – Fin de los periodos de indisponibilidad para 5 auditores.

En la hoja "Proyectos" se han insertado 70 proyectos, siendo 10 obligatorios y 60 optativos. Se ha creado una macro VBA denominada "*RellenarEsfuerzos"* para rellenar los datos de entrada relativos a la cantidad estimada de horas para la ejecución de cada uno de los proyectos (los cuales serán indicados por E_j). Los datos son automáticamente insertados de modo aleatorio, pero variando dentro de un rango de valores previamente determinado, de modo que reflejen los datos usualmente encontrados en la realidad. Estos rangos de valores cambian según los tipos de proyectos a que se refieren (lo que se explica en la próxima subsección) y su obligatoriedad o no.

La macro "*RellenarEsfuerzos"* también rellena los valores de los beneficios obtenidos con la realización de cada proyecto optativo, datos estos que son necesarios para la función objetivo del problema. Los proyectos optativos son clasificados en grados de importancia de su beneficio, de acuerdo con la siguiente escala, basada en la propuesta por Saaty (1977) para el método AHP[(2)] (Saaty, 1980):

Importancia del Proyecto	Valor del Beneficio
Muy Inferior	1
Inferior	3
Media	5
Superior	7
Muy Superior	9

Tabla 4: Evaluación del beneficio del proyecto según su importancia.

La importancia de los proyectos es determinada por los gestores de las unidades de auditoría y se refiere tanto al grado de correlación de los proyectos con los temas listados en el plan táctico de la Secretaría Federal de Control para el ejercicio como a los resultados esperados con la realización de auditoría, por ejemplo: beneficio financiero estimado o impacto positivo en la mejora de la gestión de las unidades auditadas o del respectivo programa de gobierno. Al final, lo que importa, es que se tenga una representación numérica de esta importancia para que se pueda plantear el problema por medio de un modelo matemático.

4.1.2. Las eficiencias de los auditores

Los proyectos de auditoría realizados en la CGU pueden, de modo bastante simplificado, ser clasificados en tres grandes grupos: i) evaluación de la gestión y auditorías de cuentas; ii) acciones investigativas y operaciones especiales; y iii) otros proyectos de naturaleza distinta. A pesar de ser una simplificación de la realidad, esta sencilla clasificación ya es de bastante utilidad para la diferenciación de los proyectos.

Los auditores en el equipo no presentan la misma eficiencia en la ejecución de los distintos tipos de proyectos de auditoría. Hay auditores, por ejemplo, que son más eficientes en la realización de proyectos relacionados con la evaluación de la gestión y menos eficientes

[(2)] Del inglés, *Analytic Hierarchy Process*.

en la realización de acciones de naturaleza investigativa, conforme su perfil formativo, experiencia y habilidades. Por tal razón, se ha definido lo siguiente:

i) Los proyectos de auditoría, tanto obligatorios como optativos, han sido clasificados en uno de los tres grandes grupos ya descritos, los cuales son representados por sus abreviaciones: GES (primer grupo), INV (segundo grupo) y OTR (tercer grupo, exclusión de los dos primeros).

ii) Las eficiencias de los auditores en cada tipo de proyecto han sido estimadas en una escala idéntica a la especificada por la tabla 4, es decir: (1) muy inferior; (3) inferior; (5) media; (7) superior; (9) muy superior. Sin embargo, esta escala ahora tiene un significado un poco más concreto respecto a la variación en los tiempos de ejecución estimados para cada proyecto. Esto es lo que se muestra en la tabla 5 (ver subsección 4.1.4).

iii) Se ha rellenado una tabla de eficiencias de 3 filas (una para cada tipo de proyecto) y 5 columnas (una por cada auditor) por medio de una macro VBA denominada *"RellenarEficiencias"*.

4.1.3. Las aptitudes de los auditores

No todos los auditores son aptos para realizar todos los proyectos de auditoría del conjunto seleccionable. La inaptitud puede ocurrir por diversos motivos: impedimento por conflicto de intereses, cumplimiento de cuarentena, inhabilidad, etc. Por esta razón, se ha creado una macro VBA en el Excel, denominada *"RellenarAptitudes"*, por medio de la cual se especifica qué auditores pueden realizar o no los diferentes proyectos en el periodo de planificación. Como resultado, se obtiene una matriz de tamaño AxN de ceros y unos, siendo A la cantidad de auditores y N el número de proyectos. Se escribirá uno si el auditor es apto y cero en caso contrario.

4.1.4. La tabla de tiempos de ejecución (duraciones estimadas)

Una vez hechas las debidas explicaciones iniciales, se puede, por fin, llegar a la tabla de datos de entrada más importante para el problema, relativa a los tiempos de ejecución de cada proyecto de auditoría según se asigne a un determinado auditor.

Se ha construido una matriz *N*x*A* por medio de la macro *"RellenarTiempos"*. Inicialmente cabe destacar que en esta tabla los valores de los tiempos de ejecución de los proyectos, de acuerdo con el auditor asignado, deben estar en días y no en horas. Por lo tanto, se debe realizar una división por 8 (número de horas de trabajo por día útil) del valor relativo al esfuerzo estimado. Además, se ha utilizado una función de redondeo para que se pudiera trabajar con valores enteros en vez de reales para la duración del proyecto en número de días.

En la construcción de la tabla de tiempos se han utilizado los datos de eficiencias y aptitudes presentados en las secciones anteriores, de la manera que se describe a continuación:

1. Si el auditor no es apto para determinado proyecto, se debe poner el tiempo de ejecución con un valor muy alto, de modo que el algoritmo de optimización no realice tal asignación.
2. Se debe generar un tiempo aleatorio de ejecución del proyecto para cada auditor de acuerdo con los datos de eficiencia presentados en la tabla 5.

Escala Eficiencia	**Descripción Eficiencia**	**Intervalo para generación aleatoria del valor estimado para el tiempo de ejecución del proyecto**
1	Entre 50 y 100% menos eficiente	$[1.5 \cdot E_j, 2 \cdot E_j]$
3	Entre 16 y 49% menos eficiente	$[1.16 \cdot E_j, 1.49 \cdot E_j]$
5	Entre 15% menos eficiente y 15% más eficiente	$[0.85 \cdot E_j, 1.15 \cdot E_j]$
7	Entre 16 y 29% más eficiente	$[0.71 \cdot E_j, 0.84 \cdot E_j]$
9	Entre 30 y 50% más eficiente	$[0.5 \cdot E_j, 0.7 \cdot E_j]$

Tabla 5 – Intervalo de variación de la duración estimada de un proyecto según la escala de eficiencia adoptada (utilizado en la macro "*RellenarTiempos*").

En la tabla 5, E_j se refiere a la duración estimada del proyecto sin considerar su asignación a un auditor en específico, es decir, indica el esfuerzo de ejecución estimado para cada proyecto, conforme se ha descrito en la subsección 4.1.1. Los datos son generados de modo aleatorio, pues no se dispone de una base de datos históricos acerca de los tiempos de ejecución utilizados por los auditores en cada uno de los tipos de proyectos de auditoría.

4.1.5. *Release times* y *deadlines* de los proyectos

No todos los proyectos de auditoría están disponibles desde el primer instante del periodo de planificación. Algunos proyectos, tanto obligatorios como optativos, solo están disponibles en fechas específicas. La disponibilidad del proyecto para inicio de la ejecución es determinada por distintos factores. En los trabajos de auditoría de cuentas (de naturaleza obligatoria), por ejemplo, se sigue un calendario predeterminado anualmente por el Tribunal de Cuentas de la Unión. En otros proyectos, la razón de un *release time* específico puede ser de naturaleza más puntual. En las auditorías de obras públicas (construcción civil), por ejemplo, se debe observar el cronograma previamente establecido para las etapas de ejecución física y financiera del emprendimiento. Por tales razones, se debe considerar el *release time* de cada proyecto en la modelización del problema.

Además, algunos proyectos tienen también fechas de entrega específicas, no admitiéndose, a *priori*, su incumplimiento. De este modo, también se debe considerar este parámetro, el cual se denomina *deadline*. Se trata de un *deadline* y no de un *due date* debido a la restricción impuesta de que no se admiten atrasos en su programación. Por esta misma razón, no se van a considerar pesos en la tardanza (*tardiness*) de los trabajos. La restricción de tardanza total (de los proyectos) igual a cero se debe al hecho de que se debe entender la programación como una estimación del tiempo de realización, ya que se habla de personas en este problema, no de máquinas. Las personas no van a ejecutar los proyectos exactamente en el tiempo que se ha programado. En la realidad, es siempre más probable que el tiempo de ejecución real sea más grande que lo programado. Sin embargo, al principio, se había planteado el problema con el objetivo de minimizar las tardanzas de los trabajos, lo que ha sido posteriormente modificado al comprenderse que este es un enfoque que tiene más que ver con máquinas, de naturaleza determinista, que con recursos humanos. En el caso de que no se impusiera la restricción de tardanza total igual a cero, se deberían definir pesos para determinar objetivamente el grado de importancia de la fecha de entrega de cada proyecto, es decir, la severidad de la penalización – en la función objetivo del modelo matemático propuesto – que es debida al atraso en su finalización.

Se ha creado una macro para generar – de modo aleatorio, pero considerando las características del problema – los *release times* y *deadlines* para cada uno de los proyectos de la relación total disponible inicialmente. Si determinado proyecto está disponible para

realizarse desde el inicio de la ventana de tiempo, su *release time* vale cero. En caso contrario, recibirá el valor correspondiente al día útil en que se estima que estará disponible. Además, si el proyecto no posee restricción respecto a su fecha de entrega, su *deadline* recibirá el valor relativo al último instante de tiempo (último día útil del año) del periodo de planificación. Por otro lado, si el proyecto sí que tiene una fecha específica de entrega, se deberá asignar a la variable *deadline* el valor que le corresponde.

4.2. Modelización matemática exacta

Puesto que se ha completado la descripción textual del problema en que se centra este trabajo y ya han sido definidos los métodos de obtención de los datos que sirven como parámetros de entrada, se pasa a la determinación del modelo matemático que lo representa.

La literatura sobre problemas de secuenciación comprende una amplia gama de soluciones heurísticas para su resolución, una vez que la gran mayoría de los problemas reales de programación de la producción son del tipo *NP-hard*. Sin embargo, según Unlu y Mason (2010), es corriente que el primer paso de un investigador, antes de desarrollar un método heurístico, sea la formulación de un modelo de programación matemática para el cálculo de la solución óptima exacta para instancias pequeñas – es decir, computacionalmente viables – del problema abordado. Esto sirve tanto para intentar entender la estructura subyacente del problema como para evaluar la eficacia de los heurísticos que se desarrollen.

Hay varias maneras de modelizarse matemáticamente un mismo problema, las cuales dependen de las variables de decisión que son empleadas en la construcción del modelo.

Los modelos para la resolución exacta de los problemas de programación de la producción son del tipo MIP, es decir, son modelos de programación matemática entera mixta, los cuales difieren de los modelos de programación lineal por la existencia de variables de decisión que solo admiten valores enteros.

Unlu y Mason (2010) han clasificado las formulaciones MIP de los problemas de secuenciación, de acuerdo con las variables de decisión empleadas, en cinco tipos principales: i) variables de tiempo de finalización de los trabajos (*job completion time*); ii) variables de asignación y de fechas posicionales (*assignment and positional date*); iii)

variables de ordenación linear (*linear ordering*); iv) variables de indexación en el tiempo (*time indexed*); y v) variables de red (*network*).

En este trabajo se ha elegido la formulación de variables indexadas en el tiempo para modelizar el problema de selección y secuenciación de los proyectos de auditoría para el periodo de un año. Esta formulación ha sido introducida por primera vez en la literatura académica por Sousa y Wolsey (1992), los cuales estudiaron el problema de programación de trabajos para una única máquina.

De acuerdo con Van den Akker, Hurkens y Savelsbergh (2000), una ventaja importante de la formulación de variables indexadas en el tiempo es su versatilidad, además, su relajación lineal puede proveer una cota fuerte, la cual domina sobre las cotas proporcionadas por las demás formulaciones. Por otro lado, su principal desventaja se refiere a la gran cantidad de variables de decisión, por lo tanto, para instancias con muchos trabajos o con trabajos de largos tiempos de procesamiento, los requerimientos de memoria y los tiempos de computación serán elevados.

Con respecto a la representación de los periodos de indisponibilidad de los auditores, se ha utilizado la notación descrita por el artículo de Ma, Chu, y Zuo (2010). En este trabajo, los autores llevan a cabo un *survey* de los modelos de programación de máquinas con restricciones de indisponibilidades deterministas.

4.2.1. El modelo MIP con variables indexadas en el tiempo

A partir de los artículos de Unlu y Mason (2010) y de Ma, Chu, y Zuo (2010), se ha definido a continuación un modelo matemático para la resolución del problema de programación de proyectos de auditoria presentado en este trabajo. No obstante, para poder expresar el modelo, debemos previamente definir los conjuntos de datos, las variables de decisión y los parámetros tal y como se muestran en las tablas siguientes:

P	Conjunto **total** de proyectos de auditoría, indexados $j = 1, \ldots, n$
P_{Obl}	Conjunto de proyectos de auditoría **obligatorios**, indexados $j = 1, \ldots, Obl$
P_{Opt}	Conjunto de proyectos de auditoría **optativos**, indexados $j = Obl + 1, \ldots, n$
A	Conjunto de auditores, indexados $i = 1, ..., m$
τ	Conjunto de períodos de tiempo, indexados $t = 1, ..., u$

Tabla 6 – Conjuntos de datos del modelo.

X_{ij}^{t}	Variable binaria que vale 1 cuando el proyecto *j* es iniciado por el auditor *i* en el tiempo *t* y 0 en caso contrario.
Y_{ij}^{q}	Variable binaria que vale 1 cuando el proyecto *j* ejecutado por el auditor *i* inicia antes del periodo de indisponibilidad *q* y no se puede completar antes de este periodo. Será 0 en caso contrario.
C_j	Tiempo de finalización (*Completion Time*) del proyecto *j*; no puede ser negativo.
T_j	Tardanza (*Tardiness*) en la ejecución de los proyectos *j*; tomará un valor no negativo.

Tabla 7 – Variables de decisión del modelo.

$\boldsymbol{Ben}_j$	Beneficio asociado a la realización del proyecto optativo *j*, cuyo valor es previamente estimado dentro de una escala de importancia variando de 2 en 2 unidades entre 1 y 9 (ver tabla 4).
d_j	Fecha límite de entrega (*deadline*) del proyecto *j*.
r_j	Fecha a partir de la cual se puede iniciar el proyecto *j* (*release time*).
p_{ij}	Tiempo de procesamiento (ejecución) del proyecto *j* por el auditor *i*; no negativo.
u	Límite superior (*upper bound*) para el tiempo total requerido para la ejecución de todos los trabajos.
n	Número total de proyectos de auditoría.
Obl	Número de proyectos de auditoría obligatorios.
m	Número de auditores.
S_i	Cantidad predeterminada de periodos de indisponibilidad del auditor *i* en el periodo por planificar (por razón de vacaciones, licencias médicas, acciones de capacitación, etc.). En el problema de este trabajo se ha definido que su valor mínimo es igual a 1 y su valor máximo es igual a 3.
$[B_i^q, F_i^q]$	Intervalo *q* de indisponibilidad del auditor *i*. El parámetro *q* varía entre 1 y S_i. *B* se refiere al tiempo de inicio y *F* al tiempo de fin del intervalo de indisponibilidad.

Tabla 8 – Parámetros de entrada del modelo.

Teniendo en cuenta la notación presentada, el modelo para el problema tratado en este trabajo se puede expresar como sigue:

$$\max \sum_{i \in A} \sum_{t=0}^{u-p_{ij}} \sum_{j=Obl+1}^{N} \boldsymbol{Ben}_j \cdot X_{ij}^t \tag{3.1}$$

$$T_j = \max\left(C_j - d_j, 0\right) \quad j \in P \tag{3.2}$$

$$\sum_{i \in A} \sum_{t=0}^{u-p_{ij}} X_{ij}^t = 1 \qquad j \in P_{Obl} \tag{3.3}$$

$$\sum_{i \in A} \sum_{t=0}^{u-p_{ij}} X_{ij}^t \leq 1 \qquad j \in P_{Opt} \tag{3.4}$$

$$\sum_{j \in P} X_{ij}^t \leq 1 \qquad i \in A, t = 1, \dots, u \tag{3.5}$$

$$\sum_{j \in P} \sum_{h=\max\left(0,\ L_{ij}^q\right)}^{t-1} X_{ij}^h \leq 1$$

$$donde\ L_{ij}^q = \begin{cases} t - p_{ij} - \left(F_i^q - B_i^q + 1\right), si\ t \in [B_i^q, F_i^q] \\ t - p_{ij}\ , en\ otro\ caso \end{cases} \tag{3.6}$$

$$\text{i} \in A, q \in S_i, t = 1, \dots, u$$

$$C_j = \sum_{i \in A} \sum_{t=0}^{u-p_{ij}} \left(t + p_{ij}\right) \cdot X_{ij}^t + \sum_{i \in A} \sum_{q \in \{1..S_i\}} \left(F_i^q - B_i^q + 1\right) \cdot Y_{ij}^q \quad j \in P \tag{3.7}$$

$$\sum_{i \in A} \sum_{t=0}^{r_j - 1} X_{ij}^t = 0 \qquad j \in P \tag{3.8}$$

$$\sum_{j \in P} \sum_{t=B_i^q}^{F_i^q} X_{ij}^t = 0 \qquad \text{i} \in A, q \in S_i \tag{3.9}$$

$$\sum_{t=0, t \neq B_i^q - p_{ij}}^{B_i^q - 1} \left(t + p_{ij}\right) \cdot X_{ij}^t \geq B_i^q \cdot Y_{ij}^q \tag{3.10}$$

$\text{i} \in A, j \in P, q \in S_i$

$$M \cdot Y_{ij}^q \geq \sum_{t=0, t \neq B_i^q - p_{ij}}^{B_i^q - 1} \left(t + p_{ij}\right) \cdot X_{ij}^t - B_i^q \tag{3.11}$$

$\text{i} \in A, j \in P, q \in S_i$

$$Y_{ij}^q \leq (1 - X_{ij}^t) \tag{3.12}$$

$\text{i} \in A, j \in P, q \in S_i, t = 1..u, t = B_i^q - p_{ij}$

$$\sum_{j \in P} T_j \leq 0 \tag{3.13}$$

$$X_{ij}^t, Y_j^q \in \{0,1\} \tag{3.14}$$

$$p_{ij}, r_j, d_j, u, C_j, T_j, B_i^q, F_i^q \in \mathbb{N} \tag{3.15}$$

$$\boldsymbol{Ben}_j \in \{1, 3, 5, 7, 9\}, \qquad S_i \in \{1, 2, 3\} \tag{3.16}$$

En (3.1) tenemos la función objetivo de nuestro modelo, que es la maximización de los beneficios adquiridos con la realización de los proyectos optativos en el periodo anual de programación. La ecuación (3.2) define la tardanza en la ejecución de un proyecto *j* como la diferencia entre la fecha de finalización del proyecto y la fecha límite de entrega preestablecida (*deadline*). Como la tardanza no puede ser negativa, en el caso de que la fecha de finalización sea inferior (anterior) al *deadline*, la tardanza toma el valor cero.

Las restricciones (3.3) y (3.4) aseguran que cada proyecto solo puede ser iniciado por un auditor y en un único instante de tiempo. Todos los proyectos obligatorios deben ser realizados, a diferencia de los optativos que pueden ser realizados o no.

La restricción (3.5) asegura que un auditor solo puede iniciar un proyecto en un instante de tiempo. La restricción (3.6) junto con la (3.5) evita que un auditor pueda iniciar un proyecto nuevo antes de terminar el que tiene en curso.

La ecuación (3.7) calcula el tiempo de finalización de cualquier proyecto *j*. Los tiempos de duración de las indisponibilidades de los auditores solo son considerados en el cálculo cuando se activan las variables auxiliares Y_{ij}^{q}. La restricción (3.8) es la relativa a los *release times*, es decir, mientras *t* sea menor que r_j, el proyecto *j* no puede ser iniciado por ningún auditor.

En (3.9) se garantiza que un auditor no puede iniciar un proyecto mientras esté en un periodo de indisponibilidad. Las restricciones (3.10), (3.11) y (3.12) activan las variables binarias Y_{ij}^{q} cuando corresponde. Es decir, las variables Y_{ij}^{q} solo pueden valer 1 cuando el proyecto se ha iniciado en algún instante anterior al actual y no ha podido finalizar por tener un período de indisponibilidad. El auditor reanuda el trabajo al finalizar el período de indisponibilidad, por lo que en la restricción (3.7) se considera la duración de la indisponibilidad en el tiempo de finalización del proyecto.

La restricción (3.13) determina que se debe asegurar, en la selección de los proyectos optativos, que las tardanzas de todos los proyectos sean nulas (no son admitidas tardanzas en la programación). Las demás restricciones – de (3.14) a (3.16) – solamente explicitan a que conjuntos numéricos pertenecen las variables y parámetros utilizados en el modelo.

Cabe observar que el modelo de programación lineal entera definido en esta sección se refiere al problema de programación de proyectos de auditoría tratado en este trabajo con la consideración de todas las restricciones comentadas en el capítulo 2, siendo las más importantes la de *release times* y la de indisponibilidades. La primera es de fácil representación en un modelo de variables indexadas en el tiempo, como se ha visto en la ecuación (3.8). Sin embargo, la restricción de indisponibilidades es más difícil de incorporar en el modelo, siendo responsable de un cambio más significativo en su estructura.

En el capítulo 6, relativo a los experimentos computacionales, el modelo MIP aquí presentado será probado en la resolución no solo de instancias del problema con todas sus restricciones, sino también en instancias donde no se consideran una de las dos (o ambas) restricciones mencionadas, es decir, la de *release times* y la de indisponibilidades. En el caso de no considerarse la primera, es suficiente, para la corrección del modelo, eliminar la ecuación (3.8) o asignar el valor cero a todos los r_j. Para no considerar las indisponibilidades, se eliminarían las ecuaciones (3.10), (3.11) y (3.12) y se cambiarían las ecuaciones (3.6) y (3.7), que quedarían de la siguiente manera:

$$\sum_{j \in P} \sum_{h=\max(0,t-p_{ij})}^{t-1} X_{ij}^{h} \leq 1 \qquad \text{i} \in A, t = 1, \ldots, u \tag{3.6'}$$

$$C_j = \sum_{i \in A} \sum_{t=0}^{u-p_{ij}} \left(t + p_{ij}\right) \cdot X_{ij}^{t} \qquad j \in P \tag{3.7'}$$

5. Algoritmos heurísticos

El modelo MIP propuesto no ha sido capaz de encontrar buenas soluciones factibles para instancias más allá de 10 auditores y 110 proyectos en un tiempo de computación razonable. Por ello, se ha considerado necesario diseñar un algoritmo heurístico capaz de obtener buenas soluciones del problema en un tiempo computacional muy inferior. En este punto, cabe destacar que Fernández et al (1996) define un método heurístico como un *"procedimiento para resolver un problema matemático bien definido mediante una aproximación intuitiva, en la que la estructura del problema se utiliza de forma inteligente para obtener una buena solución"*.

Los algoritmos heurísticos no garantizan la optimalidad de la solución encontrada, pero deben ser capaces de encontrar soluciones muy buenas (cercanas al valor óptimo, cuando éste sea conocido) con un coste computacional menor al incurrido con la resolución exacta. Además de ser eficiente y eficaz, un buen heurístico también debe ser robusto, es decir, la buena calidad de la solución se debe mantener independientemente de las características de la instancia a resolver.

Se denominan constructivos los heurísticos que empiezan con una solución vacía y que la extienden repetidamente hasta que una solución completa es obtenida. En este trabajo se han propuesto dos heurísticos únicamente de este tipo, pero cabe señalar que hay otros tipos como los de búsqueda local, metaheurísticos, hiperheurísticos y otros, cuya explicación se puede obtener en Martí, Pardalos y Resende (2018). En las subsecciones siguientes se describen los dos heurísticos constructivos desarrollados en este trabajo, habiendo sido utilizada la misma notación del modelo MIP para los datos del problema.

5.1. Primer heurístico constructivo

Este primer heurístico está formado por dos fases, denominadas fase 1 y fase 2, respectivamente. La fase 1 consiste en asignar y secuenciar los proyectos obligatorios, cuya lista se denota como P_{Obl}. Los proyectos en P_{Obl} son previamente ordenados, en orden no decreciente, según sus *deadlines* (parámetro principal) y *release times* (parámetro secundario). Esto implica que, tras ordenar los proyectos obligatorios según sus *deadlines*, en caso de que haya algún empate, se utiliza la segunda ordenación por los *release times*. Por lo tanto, se da prioridad a la ejecución de los proyectos obligatorios que tienen fechas máximas de entrega más tempranas, así como los que están disponibles lo antes posible. El primer proyecto de la lista P_{Obl} ordenada es secuenciado en el calendario del auditor que presente el menor tiempo de procesamiento para este proyecto (el más eficiente posible) y que tenga disponibilidad para ejecutarlo antes de su *deadline* (debido a los periodos de indisponibilidad del auditor). La secuenciación sigue hasta que se agoten todos los proyectos obligatorios. El pseudocódigo "Algoritmo 1 (fase 1)" muestra la primera fase del primer heurístico constructivo.

for *j* in 1:[número de proyectos obligatorios] **do**
- ➔ Seleccionar el j^{th} proyecto de la lista P_{Obl} **ordenada**

assignDone = false
A = conjunto de todos los *m* auditores
while assignDone = false
- ➔ Seleccionar el auditor *i* del conjunto *A* con el menor tiempo de procesamiento para el proyecto *j* seleccionado

if el auditor *i* seleccionado tiene disponibilidad de calendario para ejecutar el proyecto *j* antes de su *deadline* **then**
- ➔ Asignar el proyecto *j* al auditor *i,* secuenciándolo en el menor instante de tiempo disponible.
- ➔ Actualizar el calendario del auditor *i*.
- ➔ assignDone = true

else
- ➔ Eliminar el auditor *i* del conjunto *A*.

```
        end
    end
end
```

Algoritmo 1 (fase 1) – Primer Heurístico Constructivo: secuenciación de los proyectos obligatorios.

La fase 2 implica la selección y secuenciación de los proyectos optativos. Se ha hecho la ordenación de los proyectos de la lista P_{Opt} según sus beneficios (no creciente), en primer lugar, y sus tiempos medios de procesamiento (no decreciente), como criterio de desempate. Las fechas de entrega (*deadlines*) no han sido consideradas en la ordenación porque, en el caso de los proyectos optativos, serán siempre las mismas (el último día del año). Con esto se tiene que los primeros proyectos optativos a elegir son los que agregan más beneficios con su ejecución y requieren menos tiempo de procesamiento. Respecto a los tiempos medios de procesamiento, es importante destacar que se refieren a un promedio simple de los tiempos de ejecución de los proyectos por cada uno de los auditores aptos (se excluyen del cálculo los declarados no aptos en los datos de entrada).

El algoritmo de secuenciación de los proyectos optativos es bastante parecido al de secuenciación de los proyectos obligatorios. La principal diferencia es que no es obligatoria la asignación a alguno de los auditores. Por esta razón se introduce un control adicional en la condición del *while*, para averiguar si ya se ha verificado la disponibilidad de todos los auditores. Una vez comprobado para todos los auditores, si ninguno ha presentado disponibilidad para la ejecución del proyecto seleccionado, ese proyecto no se asigna y se pasa al siguiente. El pseudocódigo "Algoritmo 1 (fase 2)" muestra el funcionamiento de la segunda fase del primer heurístico constructivo.

```
TotalSelectedProjs = 0
SumaBeneficios = 0

for j in 1:[número de proyectos optativos] do
    ➔ Seleccionar el j^th proyecto de la lista P_Opt ordenada
    assignDone = false
    i = 1
```

while assignDone = false **AND** i <= [número de auditores]

➔ Seleccionar el auditor *i* con el menor tiempo de procesamiento para el proyecto *j* seleccionado

if el auditor *i* seleccionado tiene disponibilidad de calendario para ejecutar el proyecto *j* antes de su *deadline* **then**

➔ Asignar el proyecto *j* al auditor *i,* secuenciándolo en el menor instante de tiempo disponible.

➔ Actualizar el calendario del auditor *i*.

➔ assignDone = true

➔ SumaBeneficios = SumaBeneficios + [Beneficio Proyecto Asignado]

➔ TotalSelectedProjs = TotalSelectedProjs + 1

else

➔ Mover al próximo auditor (i = i + 1)

end

end

end

Algoritmo 1 (fase 2) – Primer Heurístico Constructivo:
secuenciación de los proyectos optativos.

Es importante destacar que los pseudocódigos de las fases del algoritmo han sido presentados de modo simplificado, sin entrar en los detalles de implementación.

Para que se pueda comprender mejor el funcionamiento del algoritmo constructivo descrito, se presenta, a continuación, un ejemplo de una instancia muy pequeña del problema con 2 auditores, 4 proyectos obligatorios y 20 proyectos optativos. Las tablas 9 y 10 siguientes muestran los datos de entrada para este ejemplo.

Proyectos (*j*)	p_{1j}	p_{2j}	*Release*	*Deadline*	Tiempo Medio	Ben_j
1	24	41	60	144	33	--
2	25	47	0	186	36	--
3	26	48	60	144	37	--
4	31	44	102	186	38	--
5	31	50	187	246	41	5
6	23	50	39	246	37	5
7	23	36	123	246	30	9
8	28	39	0	246	34	5
9	31	45	81	246	38	3
10	26	40	39	246	33	3
11	26	37	39	246	32	3
12	20	44	123	246	32	7
13	17	50000	0	246	17	7
14	50000	41	0	246	41	9
15	26	52	39	246	39	7
16	17	30	0	246	24	7
17	37	15	39	246	26	9
18	42	19	123	246	31	1
19	47	25	123	246	36	1
20	49	24	123	246	37	7
21	16	12	123	246	14	1
22	27	24	0	246	26	5
23	27	24	39	246	26	1
24	16	12	0	246	14	5

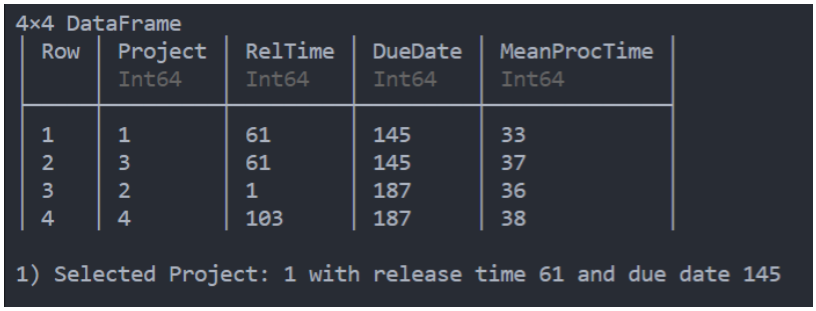
Tabla 9 – Datos de entrada para una instancia del problema con 2 auditores y 24 proyectos (4 obligatorios y 20 optativos), relativos a los tiempos de ejecución, *release times*, *deadlines* y beneficios.

Auditores (*i*)	$B_i^1 - F_i^1$	$B_i^2 - F_i^2$	$B_i^3 - F_i^3$
1	0 – 20	229 – 246	NA
2	0 – 4	158 – 167	224 – 233

Tabla 10 – Datos de entrada para una instancia del problema con 2 auditores y 24 proyectos (4 obligatorios y 20 optativos), relativos a las indisponibilidades.

En la columna "Tiempo Medio" se muestra el tiempo de procesamiento medio para cada proyecto, lo cual se calcula como la media del tiempo de proceso entre los auditores 1 y 2. La columna "Ben_j" se refiere al beneficio asociado a la realización de los proyectos optativos, sombreados en amarillo. A los proyectos obligatorios, sombreados en rojo, no se aplican los valores de beneficios.

Lo primero que hace el algoritmo es, como ya se ha enunciado, ordenar la lista de proyectos obligatorios. En este ejemplo, se tiene que la lista P_{Obl} ordenada es la que se visualiza en la figura 1.



4×4 DataFrame

Row	Project	RelTime	DueDate	MeanProcTime
	Int64	Int64	Int64	Int64
1	1	61	145	33
2	3	61	145	37
3	2	1	187	36
4	4	103	187	38

1) Selected Project: 1 with release time 61 and due date 145

Figura 1 – Lista de datos P_{Obl} ordenada según los *deadlines* (primer criterio) y los *release times* (criterio de desempate).

Se aprecia en la figura 1 que hubo un incremento de una unidad en todos los tiempos. Se ha hecho esto para evitar problemas con el valor 0 (cero) en la indexación de los *arrays* del lenguaje de programación Julia, utilizado en la codificación de los heurísticos.

El primer proyecto en la lista P_{Obl} ordenada es el primero que se selecciona para la secuenciación y así sucesivamente hasta el último. Por lo tanto, como se ve en la figura, el primer proyecto obligatorio que se selecciona es el de número 1 (como hubo un empate para los dos criterios entre los proyectos 1 y 3, se ha elegido el de menor número de identificación). Para el proyecto 1, el tiempo de ejecución para el auditor 1 es de 24 unidades de tiempo (días), mientras para el auditor 2 es de 41, conforme se ve en la tabla 9. Luego, se elige el auditor 1 para la asignación del proyecto 1, ya que no existe todavía problema de indisponibilidad en su calendario, es decir, la ventana de tiempo para secuenciación aún no ha sido ocupada por cualquier proyecto. Como el *release time* para el proyecto 1 es igual a 61, su tiempo de inicio de la ejecución por el auditor 1 será precisamente este valor, pues el algoritmo se lo asigna en el menor instante de tiempo posible. El tiempo de finalización del proyecto 1 por el auditor 1 será en el instante 84[(3)], ya que no existe, en este intervalo, algún periodo de indisponibilidad previamente determinado para este auditor.

[(3)] El tiempo de finalización es en el instante 84 y no en el 85 porque se considera que tanto el instante inicial (61) como el final (84) componen el periodo de procesamiento del proyecto, luego, el tiempo de finalización es igual al tiempo de inicio más el tiempo de procesamiento menos una unidad. El próximo instante de tiempo disponible para el inicio de un nuevo proyecto que será igual al instante 85. Esta lógica ha sido seguida en todos los tres heurísticos constructivos que se presentan en este capítulo.

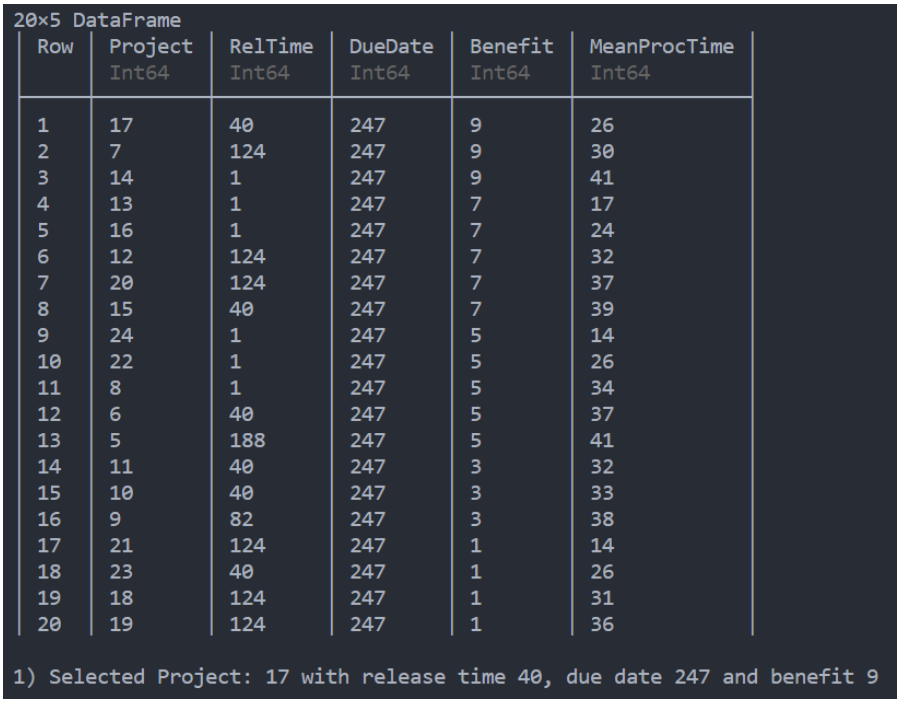
Pasando al próximo proyecto de la lista P_{Obl} ordenada, el algoritmo selecciona ahora el proyecto 3. Para este proyecto, el tiempo de ejecución para el auditor 1 es de 26 unidades de tiempo, mientras para el auditor 2 es de 48. Luego, como el algoritmo es del tipo voraz, es decir, siempre elije la mejor eficiencia disponible en cada paso de la construcción de la solución, el auditor 1 será elegido de nuevo. El *release time* del proyecto 3 es igual a 61, pero ahora, debido al resultado de la secuenciación anterior del proyecto 1, el menor instante de tiempo disponible es el instante 85. Luego, el tiempo de inicio de la ejecución del proyecto 3 por el auditor 1 asumirá este valor (85). El tiempo de finalización de este proyecto por el auditor 1 será en el instante 110, ya que tampoco existe, en este intervalo, algún periodo de indisponibilidad. Esta lógica continúa siendo seguida por el algoritmo hasta la secuenciación de todos los proyectos obligatorios de la lista P_{Obl} ordenada. Al final de esta etapa se tiene el resultado parcial que se presenta en la figura 2. En esta figura se verifica, en la columna "AA" (del inglés, *"Assigned Auditor"*) que todos los proyectos obligatorios son programados para el auditor 1. Cabe señalar que la columna "ST" (del inglés, *"Start Time"*) se refiere a los tiempos de inicio de los proyectos.

```
julia> showall(SchedData)
24×3 DataFrame
 Row │ Project  AA     ST
     │ Int64    Int64  Int64
─────┼──────────────────────
   1 │       1      1     61
   2 │       2      1     22
   3 │       3      1     85
   4 │       4      1    111
```

Figura 2 – Resultado de la secuenciación de los proyectos obligatorios (*Julia DataFrame*).

Se percibe que el calendario del primer auditor se rellena primero porque, de modo general, es un auditor más eficiente que el segundo. Una vez que ya se han secuenciado todos los proyectos obligatorios, el algoritmo pasa a la segunda fase de la construcción de la solución que es la secuenciación de los proyectos optativos.

De semejante modo a la primera fase, el primer paso del algoritmo consiste en ordenar la lista de proyectos optativos. En este ejemplo, se tiene que la lista P_{Opt} ordenada es la que se visualiza en la figura 3.



20×5 DataFrame

Row	Project	RelTime	DueDate	Benefit	MeanProcTime
	Int64	Int64	Int64	Int64	Int64
1	17	40	247	9	26
2	7	124	247	9	30
3	14	1	247	9	41
4	13	1	247	7	17
5	16	1	247	7	24
6	12	124	247	7	32
7	20	124	247	7	37
8	15	40	247	7	39
9	24	1	247	5	14
10	22	1	247	5	26
11	8	1	247	5	34
12	6	40	247	5	37
13	5	188	247	5	41
14	11	40	247	3	32
15	10	40	247	3	33
16	9	82	247	3	38
17	21	124	247	1	14
18	23	40	247	1	26
19	18	124	247	1	31
20	19	124	247	1	36

1) Selected Project: 17 with release time 40, due date 247 and benefit 9

Figura 3 – Lista de datos P_{Opt} ordenada según los beneficios (primer criterio) y los tiempos medios de procesamiento (criterio de desempate).

El primer proyecto en la lista P_{Opt} ordenada es el primero que se selecciona para la secuenciación y así sucesivamente hasta el último. Por lo tanto, como se ve en la figura, el primer proyecto optativo que se selecciona es el de número 17. Para este proyecto, el tiempo de ejecución para el auditor 1 es de 37 días, mientras para el auditor 2 es de 15, conforme se ve en la tabla 9. Por tanto, se elige el auditor 2 para la asignación del proyecto 17, lo cual no tiene todavía problema de indisponibilidad en su calendario. Como el *release time* para el proyecto 17 es igual a 40, su tiempo de inicio de la ejecución por el auditor 2 será precisamente este valor, pues el algoritmo se lo asigna en el menor instante de tiempo posible. El tiempo de finalización del proyecto 17 por el auditor 2 será en el instante 54, ya que no existe, en este intervalo, algún periodo de indisponibilidad previamente determinado para este auditor.

Pasando al próximo proyecto de la lista P_{Opt} ordenada, el algoritmo selecciona ahora el proyecto 7. Para este proyecto, el tiempo de ejecución para el auditor 1 es de 23 días, mientras para el auditor 2 es de 36. Por lo tanto, se le asigna el proyecto al primer auditor. El *release time* del proyecto 7 es igual a 124, pero, debido al resultado de las secuenciaciones realizadas anteriormente, el menor instante de tiempo disponible superior al *release time* para el auditor 1 es 142. Consecuentemente, el tiempo de inicio de la ejecución del proyecto

7 por el auditor 1 asumirá este valor (142). El tiempo de finalización de este proyecto por el auditor 1 será en el instante 164, ya que también no existe, en este intervalo, algún periodo de indisponibilidad.

El próximo proyecto de la lista ordenada es el proyecto 14. El auditor 1 no es apto para este proyecto (ya que tiene un tiempo de proceso enorme que indica esta circunstancia tal y como se aprecia en la tabla 9), por lo tanto, solo se puede asignar al auditor 2. Como no es posible secuenciarlo antes del proyecto 17, el tiempo de inicio de su ejecución será inmediatamente posterior al término de este proyecto, es decir, asumirá el valor 55. La finalización ocurrirá en el instante 95, ya que el tiempo de procesamiento por el auditor 2 es de 41 días y no existen indisponibilidades programadas en este intervalo. Se sigue esta lógica hasta la secuenciación de todos los proyectos obligatorios de la lista P_{Opt} ordenada, obteniéndose al final la solución presentada en la figura 4. Esta figura muestra, a través de un gráfico de Gantt, los proyectos programados para los auditores 1 y 2, respectivamente. Cabe señalar que los rectángulos rojos se refieren a los intervalos de indisponibilidad para cada auditor.

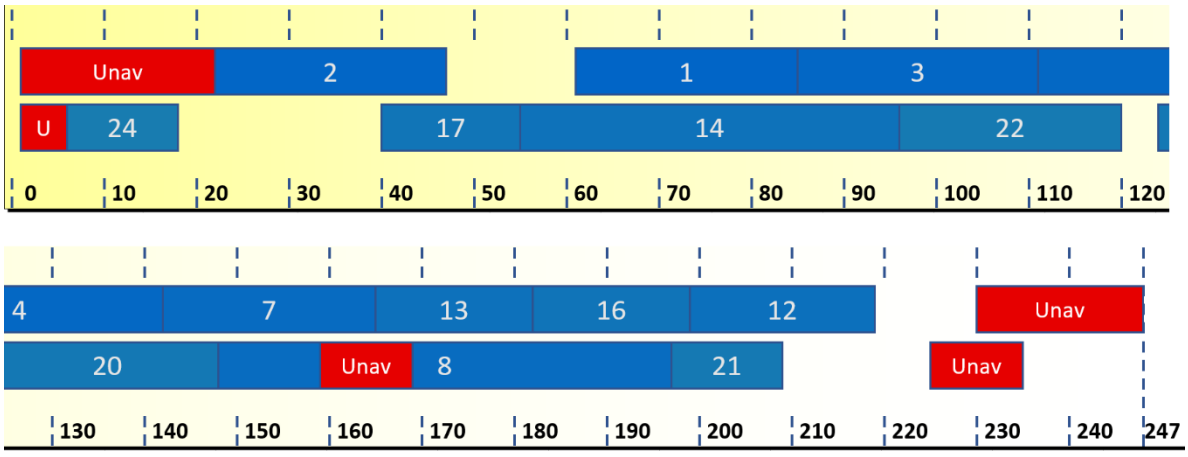


Figura 4 – Gráfico de Gantt de la solución del primer heurístico constructivo para una instancia del problema con 2 auditores y 24 proyectos (4 obligatorios y 20 optativos).

En esta solución son seleccionados 11 de los 20 proyectos optativos disponibles, habiendo sido obtenida una suma de beneficios igual a 71. El modelo MIP resuelto por CPLEX ha encontrado una solución óptima con una suma de beneficios igual a 82, con la selección de 12 proyectos optativos. Por lo tanto, la solución encontrada por el heurístico

para esta pequeña instancia tiene un RPD[4] (*Relative Percentage* Deviation), respecto al valor óptimo conocido, igual a 13,41%.

5.2. Segundo heurístico constructivo

En el segundo heurístico constructivo se ha adoptado un enfoque distinto para la construcción paso a paso de la solución. En lugar de asignar el proyecto al auditor disponible con menor tiempo de procesamiento (enfoque del primer heurístico), ahora se decide, para cada auditor y en cada instante de tiempo, cual es el mejor proyecto en base a unos criterios de ordenación que son presentados a continuación.

Para cada auditor se han calculado los siguientes parámetros de rendimiento respecto al procesamiento de los proyectos:

1. *Rank*: Rango de clasificación del auditor respecto al rendimiento en el procesamiento del proyecto a que se refiere. Por ejemplo, para la instancia del problema con 5 auditores, el auditor con el tiempo más pequeño de procesamiento para determinado proyecto tiene *rank* igual a 1 y el auditor con el tiempo de procesamiento más grande para determinado proyecto tiene *rank* igual a 5.
2. *ProcRatio*: Es el cociente (*ratio*) del tiempo de procesamiento del auditor para un determinado proyecto por el tiempo medio de procesamiento de este proyecto para todos los auditores aptos. De su definición se nota que cuanto más pequeño sea su valor, mejor será el rendimiento del auditor en el procesamiento del proyecto.
3. *BenefTime*: Parámetro exclusivo de los proyectos optativos que se refiere al cociente entre el beneficio obtenido con la realización del proyecto y el tiempo utilizado por el auditor en su procesamiento. De su definición se deduce que cuanto más grande sea su valor, mejor será el aprovechamiento del calendario del auditor en la programación de los proyectos que proporcionan más beneficios con su realización.

[4] $RELATIVE\ PERCENTAGE\ DEVIATION\ (RPD) = \left(\frac{Best_{sol} - Method_{sol}}{Best_{sol}}\right) \times 100$

Para que se pudieran almacenar los valores de estos parámetros para cada proyecto y cada auditor, se han creado dos vectores de listas, uno para los proyectos obligatorios y otro para los proyectos optativos (P^*_{Obl} *y* P^*_{Opt}, respectivamente). En el caso de que se quiera referir a la lista de proyectos relativa a un auditor en específico, se utilizará la notación $P_{Obl}[i]$ (o $P_{Opt}[i]$), donde *i* es el índice relativo al número del auditor considerado.

El vector de listas de proyectos obligatorios, P^*_{Obl}, ha sido ordenado, en orden no decreciente, por los valores de *deadline*, *Rank* y *ProcRatio* (en el mismo orden que se presentan estos parámetros). Es decir, se prioriza la programación de los proyectos con fecha de entrega más temprana y que son procesados más eficientemente por el auditor considerado (en comparación con los demás auditores). El segundo parámetro de ordenación es considerado cuando hay empate para el parámetro anterior y así sucesivamente.

El vector de listas de proyectos optativos, P^*_{Opt}, ha sido ordenado, en el orden que se presenta a continuación, por los valores de *BenefTime* (orden no creciente), *Rank* y *ProcRatio* (orden no decreciente para estos dos). Como ya registrado en la sección anterior relativa al primer heurístico constructivo, en el caso de los proyectos optativos no es necesario considerar las fechas máximas de entrega (*deadline*) porque tienen el mismo valor en las instancias de dados generadas.

Para visualizarse mejor lo explicado, se presenta en la figura 5 una captura de pantalla relativa a la lista ordenada de proyectos optativos para el auditor número 3 en una instancia del problema con 10 auditores y 110 proyectos totales.

```
julia> showall(opdSorted[3])
90×7 DataFrame
```

Row	Project	RelTime	DueDate	Rank	RatioProc	Benefit	BenefTime
	Int64	Int64	Int64	Int64	Float64	Int64	Float64
1	105	124	247	1	0.583333	9	0.642857
2	73	1	247	1	0.583333	7	0.5
3	68	40	247	3	0.76	9	0.473684
4	55	1	247	5	0.875	9	0.428571
5	65	124	247	7	0.875	9	0.428571
6	90	124	247	3	0.758621	9	0.409091
7	24	1	247	4	0.741935	9	0.391304
8	41	1	247	1	0.694444	9	0.36
9	86	1	247	6	0.807692	7	0.333333
10	53	124	247	6	0.84	7	0.333333
11	100	1	247	2	0.733333	7	0.318182
12	30	1	247	3	0.733333	7	0.318182
13	89	1	247	4	0.878788	9	0.310345
14	98	124	247	3	0.731707	9	0.3
15	80	40	247	9	1.33333	7	0.291667
16	103	40	247	1	0.607843	9	0.290323
17	29	1	247	4	0.806452	7	0.28
18	69	124	247	4	0.892857	7	0.28
19	34	188	247	3	0.782609	5	0.277778
20	62	1	247	4	0.829268	9	0.264706
21	81	1	247	6	0.829268	9	0.264706
22	56	124	247	3	0.826087	5	0.263158
23	28	40	247	4	0.791667	5	0.263158
24	52	1	247	4	0.8	7	0.25
25	94	124	247	6	0.833333	5	0.25

Figura 5 – Ejemplo de lista ordenada de proyectos optativos para el auditor identificado con el número 3 (solo es posible visualizarse los 25 primeros).

El pseudocódigo "Algoritmo 2" muestra el funcionamiento del segundo heurístico.

Benef* = vector de tamaño *m* (número de auditores) inicializado con ceros
SumaBeneficios = 0
SelProjs* = vector de tamaño *m* (número de auditores) inicializado con ceros
TotalSelProjs = 0
AudOrder* = vector del orden de secuenciación de los auditores, de tamaño *m*

for *i* in AudOrder* **do**

#Realizar en primer lugar la programación de los proyectos obligatorios
startTime = resultado de la función que busca el menor instante de tiempo disponible para el auditor *i* (no inferior al menor *release time* de los proyectos obligatorios)
while startTime < 247 (último instante de tempo)
➔ Seleccionar el primer proyecto obligatorio *j* de la lista P_{Obl}**[i] ordenada** cuyo *release time* no sea superior al actual valor de startTime
if encontrado proyecto obligatorio *j*
if el proyecto *j* aún no ha sido secuenciado

if existe intervalo de tiempo suficiente para la programación

if el proyecto *j* puede ser finalizado antes de su *deadline*

➔ Programar el proyecto *j* para el auditor *i*, considerando la existencia o no de indisponibilidad dentro del intervalo de tiempo de la programación

➔ Eliminar el proyecto *j* de P_{Obl}**[i] ordenada**

➔ startTime = resultado de la función que busca el menor instante de tiempo disponible para el auditor

else

➔ Eliminar el proyecto *j* de P_{Obl}**[i] ordenada**

end

else

➔ startTime = resultado de la función que salta al próximo instante de tiempo disponible para el auditor

if no se encuentra un startTime para que quepa el proyecto seleccionado en el calendario del auditor

➔ startTime = resultado de la función que busca el menor instante de tiempo disponible para el auditor

➔ Eliminar el proyecto *j* de P_{Obl}**[i] ordenada**

end

end

else *#proyecto ya secuenciado previamente para otro auditor*

➔ Eliminar el proyecto *j* de P_{Obl}**[i] ordenada**

end

else *#no ha sido posible encontrar un proyecto obligatorio para empezar en startTime*

if P_{Obl}**[i] ordenada** no es vacía

➔ startTime = próximo instante de tempo en que se pueda iniciar la ejecución de otro proyecto obligatorio

else

➔ startTime = 247 *#salir del bucle*

end

end

end *#fin while*

#Realizar en segundo lugar la programación de los proyectos optativos

startTime = resultado de la función que busca el menor instante de tiempo disponible para el auditor (no inferior al menor *release time* de los proyectos optativos)

while startTime < 247 (último instante de tempo)

➔ Seleccionar el primer proyecto optativo *j* de P_{Opt}**[i] ordenada** cuyo *release time* no sea superior al actual valor de startTime

if encontrado proyecto optativo *j*

if el proyecto *j* aún no ha sido secuenciado

if existe intervalo de tiempo suficiente para la programación

if el proyecto *j* puede ser finalizado antes de su *deadline*

➔ Programar el proyecto *j* para el auditor *i*, considerando la existencia o no de indisponibilidad dentro del intervalo de tiempo de la programación

➔ Benef[i] = Benef[i] + benefSP$^{(1)}$

➔ SelProjs[i] = SelProjs[i] + 1

➔ Eliminar el proyecto *j* de P_{Opt}**[i] ordenada**

➔ startTime = resultado de la función que busca el menor instante de tiempo disponible para el auditor

else

➔ Eliminar el proyecto *j* de P_{opt}**[i] ordenada**

end

else

➔ startTime = resultado de la función que salta al próximo instante de tiempo disponible para el auditor

if no se encuentra un startTime para que quepa el proyecto seleccionado en el calendario del auditor

➔ startTime = resultado de la función que busca el menor instante de tiempo disponible para el auditor

➔ Eliminar el proyecto *j* de P_{opt}**[i] ordenada**

end

end

else *#proyecto ya secuenciado previamente para otro auditor*

➔ Eliminar el proyecto *j* de P_{opt}**[i] ordenada**

end

else *#no ha sido posible encontrar un proyecto obligatorio para empezar en startTime*

if P_{opt}**[i] ordenada** no es vacía

➔ startTime = próximo instante de tempo en que se pueda iniciar la ejecución de otro proyecto optativo

else

➔ startTime = 247 *#salir del bucle*

```
            end
          end
      end #fin while
end #fin for

#Totalizar elementos de los vectores
SumaBeneficios = sum(Benef*)
TotalSelProjs = sum(SelProjs*)
```

(1)*benefSP* = beneficio asociado a la realización del proyecto *j* seleccionado.

Algoritmo 2 - Segundo Heurístico Constructivo:
secuenciación de los proyectos obligatorios y optativos.

Lo que hace resumidamente el algoritmo es, para cada auditor, en el orden de secuenciación definido por el vector *AudOrder**, llenar su calendario individual hasta que no quepan más proyectos – observados los criterios de priorización presentados – para, posteriormente, pasar a la secuenciación del calendario de próximo auditor, hasta que todos los auditores tengan sus calendarios programados. Es fácil percibir que un cambio en el orden de recorrido de los auditores en el bucle representa un cambio en los intentos de programación del algoritmo y, por lo tanto, en los resultados de la solución encontrada, es decir, en la cantidad total de proyectos optativos seleccionados y en la suma total de beneficios obtenida con la realización de ellos.

Para que se pueda comprender mejor el funcionamiento del algoritmo, se utilizará la misma instancia considerada en la explicación del primer heurístico para la visualización de los pasos dados en la construcción de la solución para el problema. Lo primero que hace el algoritmo es ordenar las listas de proyectos obligatorios para cada auditor. En este ejemplo, se tiene que el vector de listas P^*_{Obl} ordenado es lo presentado en la figura 6.

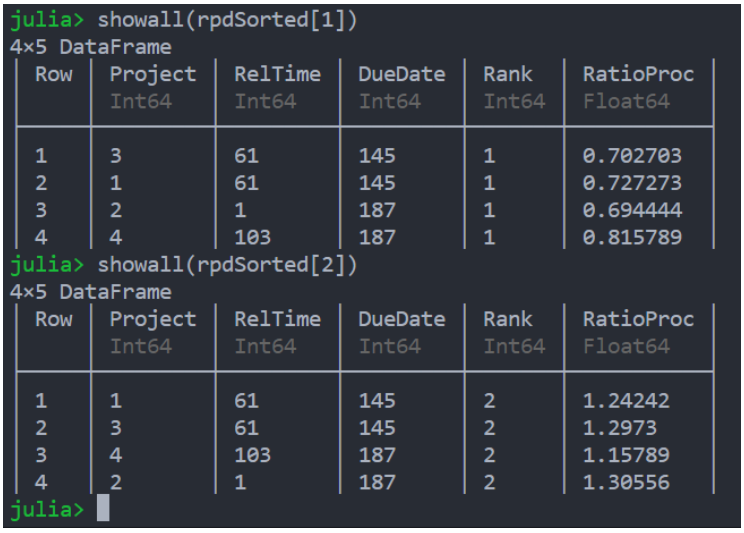


Figura 6 – Listas de datos $P_{Obl}[1]$ y $P_{Obl}[2]$ ordenadas según los *deadlines* y los parámetros de rendimiento *Rank* y *ProcRatio*.

Se ha elegido el orden natural (*i* iniciando por 1 y terminando en 2) para la programación del calendario de los auditores, lo que se hace en la declaración del bucle principal del algoritmo. Luego, se considerarán primero las listas de datos $P_{Obl}[1]$ y $P_{Opt}[1]$, relativas a las ordenaciones, para el auditor 1, de los proyectos obligatorios y optativos, respectivamente.

En el ejemplo de esta pequeña instancia de 2 auditores y 24 proyectos, iniciando por el auditor 1, se determina, al principio, el menor instante de tiempo en que se puede asignar un proyecto a este auditor. En este caso, este instante se refiere precisamente al valor 22, ya que la primera indisponibilidad del auditor 1 está comprendida entre los instantes 1 y 21. Como el proyecto 2 es el único proyecto obligatorio que está disponible en este instante (los demás serán liberados solo en los instantes 61 y 103), este es el primer proyecto que se le asigna al auditor 1. Una vez hecha la inserción de este proyecto en el calendario del auditor, se elimina de la lista $P_{Obl}[1]$ y se actualiza la variable *start time* para el valor del próximo instante de tiempo de disponibilidad para el auditor 1, que es el valor 47, ya que el tiempo de procesamiento del proyecto 2 por el auditor 1 es de 25 unidades. Sin embargo, el valor mínimo del *release time* para los proyectos que restaran es 61, por lo tanto, el código se encarga de asignar este valor al *start time*.

En el instante 61 existen 2 proyectos obligatorios disponibles: el 3 y el 1. Estos dos proyectos empatan en los dos primeros criterios de ordenación, que son el *deadline* y el *Rank*. Luego, la priorización será determinada por el tercer criterio, que es el parámetro

ProcRatio. Como el proyecto 3 presenta un valor ligeramente más pequeño para este parámetro, se elige para la asignación al auditor 1 en el instante 61. Tras la inserción del proyecto 3 en el calendario del auditor 1, se elimina de la lista $P_{Obl}[1]$ y se actualiza el *start time* para el valor del próximo instante de tiempo disponible para este auditor, que será el valor 87, ya que el tiempo de procesamiento del proyecto 3 por el auditor 1 es de 26 unidades.

En el instante 87, de los dos proyectos obligatorios que restan (el 1 y el 4), solamente el 1 ya se encuentra disponible para inicio de la ejecución, pues el proyecto 4 tiene un *release time* igual a 103. Por lo tanto, se le asigna el proyecto 1 al auditor 1 en el instante 87. El tiempo de finalización será en el instante 110, ya que el tiempo de procesamiento del proyecto 1 por el auditor 1 es de 24 unidades. Luego, el *start time* será actualizado para el valor inmediatamente posterior, que es el instante 111. En este instante solo queda el proyecto 4 por asignar, lo cual ha sido liberado para la ejecución en el instante 103. Se le asigna entonces el proyecto 4 al auditor 1 con inicio en el instante 111 y finalización en el instante 141.

Una vez que ya se han recurrido todos los proyectos obligatorios para el auditor 1, el algoritmo pasa a analizar los proyectos optativos, de acuerdo con su orden de priorización, para intentar asignárselos a este auditor. En este ejemplo, se tiene que la lista $P_{Opt}[1]$ ordenada es la presentada en la figura 7.

```
julia> showall(opdSorted[1])
20×7 DataFrame
```

Row	Project	RelTime	DueDate	Rank	RatioProc	Benefit	BenefTime
	Int64	Int64	Int64	Int64	Float64	Int64	Float64
1	16	1	247	1	0.708333	7	0.411765
2	13	1	247	1	1.0	7	0.411765
3	7	124	247	1	0.766667	9	0.391304
4	12	124	247	1	0.625	7	0.35
5	24	1	247	2	1.14286	5	0.3125
6	15	40	247	1	0.666667	7	0.269231
7	17	40	247	2	1.42308	9	0.243243
8	6	40	247	1	0.621622	5	0.217391
9	22	1	247	2	1.03846	5	0.185185
10	8	1	247	1	0.823529	5	0.178571
11	5	188	247	1	0.756098	5	0.16129
12	20	124	247	2	1.32432	7	0.142857
13	10	40	247	1	0.787879	3	0.115385
14	11	40	247	1	0.8125	3	0.115385
15	9	82	247	1	0.815789	3	0.0967742
16	21	124	247	2	1.14286	1	0.0625
17	23	40	247	2	1.03846	1	0.037037
18	18	124	247	2	1.35484	1	0.0238095
19	19	124	247	2	1.30556	1	0.0212766
20	14	1	247	2	1219.51	9	0.00018

```
julia>
```

Figura 7 – Lista de datos $P_{Opt}[1]$ ordenada según los parámetros *BenefTime*, *Rank* y *ProcRatio*.

El primer proyecto, de acuerdo con la lista $P_{Opt}[1]$ ordenada, es el proyecto 16. El algoritmo intenta asignar este proyecto en el instante 47, pues ha quedado un periodo de ociosidad entre los instantes 47 y 60 para el auditor 1. Sin embargo, como el tiempo de procesamiento de este proyecto por el auditor 1 es de 17 unidades, su inserción no cabe ahí. Se actualiza el *start time* para el próximo instante de tiempo disponible que es el instante 142. Luego, se le asigna el proyecto 16 al auditor 1 con tiempo de inicio de la ejecución en el instante 142 y tiempo de finalización en el instante 158. De la misma manera, pasando al próximo proyecto de la lista, el de número 13, se intenta secuenciar en el instante 47, pero sin éxito, pasando entonces al próximo instante disponible en el calendario, que es 159. Se le asigna el proyecto 13 al auditor 1 iniciando en el instante 159 y finalizando en el instante 175. Siguiendo esta misma lógica, se le asigna el proyecto 7, iniciando en 176 y finalizando en 198, así como el proyecto 12, con inicio en 199 y finalización en el instante 218.

Como existe un intervalo de indisponibilidad para el auditor 1 entre los instantes 230 y 247, solo ha restado al final del calendario una ventada de disponibilidad de 11 unidades de tiempo, entre los instantes 219 y 229. De este modo, como en este intervalo no caben los demás proyectos optativos restantes en la lista $P_{Opt}[1]$ ordenada, el algoritmo pasará al llenado del calendario del próximo auditor, que es el de número 2. Para que no quede exhaustiva la explicación y ya que la lógica es la misma, se salta directo al resultado final de la programación de los 16 proyectos totales para los dos auditores, presentada a continuación:

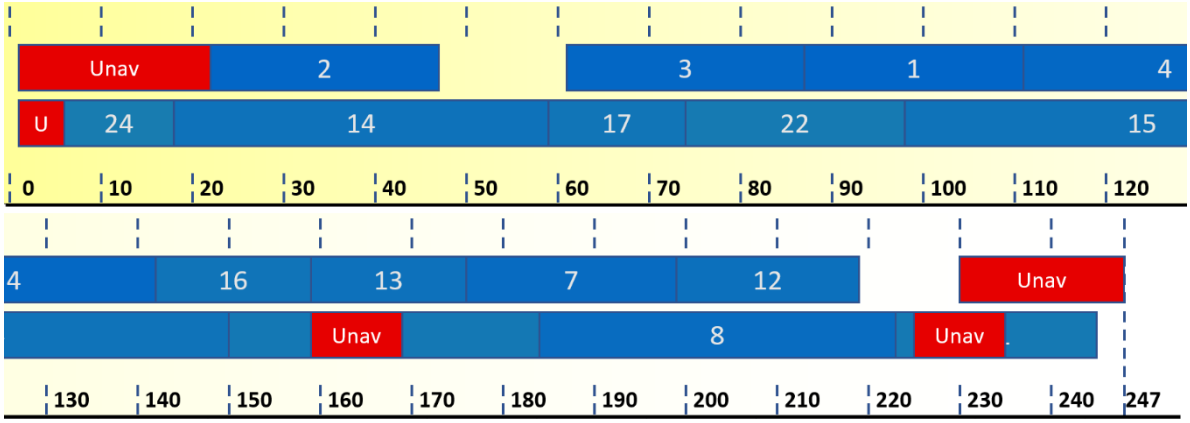


Figura 8 – Gráfico de Gantt de la solución del segundo heurístico constructivo para una instancia del problema con 2 auditores y 24 proyectos.

En esta solución son seleccionados 12 de los 20 proyectos optativos disponibles, habiendo sido obtenida una suma de beneficios igual a 78. Como ya ha sido presentado anteriormente, en la solución óptima conocida se tiene una suma de beneficios igual a 82. Luego, la solución encontrada por el segundo heurístico para esta pequeña instancia tiene un RPD igual a 4,87%, lo que representa una calidad superior frente a la solución encontrada por el primer heurístico.

Por otro lado, en el caso de que se hubiera elegido el orden reverso (primero el auditor 2 y después el 1), se obtendría una solución en que se selecciona 11 de los 20 proyectos optativos, con una suma de beneficios igual a 69 (RPD igual a 15,85%). Esta nueva solución es de calidad inferior a la obtenida por el primer heurístico. Por lo tanto, se ve que la alteración en el orden de llenado del calendario de los auditores para este pequeño ejemplo de apenas dos auditores ha representado un cambio significativo en el resultado final.

5.3. Segundo heurístico constructivo modificado

Tras analizar algunas soluciones después de realizar algunas pruebas, se ha notado que los resultados de las soluciones encontradas podrían eventualmente mejorar si los proyectos obligatorios fueran asignados a los auditores de mejor eficiencia, lo que no estaba implementado en el diseño original de este algoritmo.

Por lo tanto, se ha realizado una modificación del código para que, al inicio, se asignaran los proyectos obligatorios a los auditores más eficientes en su ejecución, es decir, aquellos con *Rank* igual a 1 para cada proyecto. Si no se consigue asignar todos los proyectos obligatorios con esta condición, se incrementa en una unidad el parámetro *Rank* y se ejecuta el código de secuenciación de los proyectos obligatorios otra vez, hasta que se asegure que todos han sido asignados. Esta modificación se ha incorporado al pseudocódigo "Algoritmo 2" añadiendo un bucle *while* para controlar esta condición. También se ha incorporado una función (denominada *isAllReqAssigned*) para verificar la asignación de todos los proyectos obligatorios, tal y como se muestra en el pseudocódigo "Algoritmo 3":

```
minRank = 1
while isAllReqAssigned = false
    for i in AudOrder* do
```

➔ ejecutar el código de secuenciación de los proyectos obligatorios del **algoritmo 2**, asignándolos solamente a los auditores con *Rank* menor o igual a **minRank**.

end

minRank = minRank + 1

end

Algoritmo 3 – Modificación del Algoritmo 2.

Volviendo a la instancia del ejemplo de 2 auditores y 24 proyectos, se ha resuelto con el segundo heurístico modificado. Si resolvemos siguiendo el orden natural de los auditores (primero el 1, después el 2), como cabía esperar, no se ha obtenido una solución con una suma de beneficios más grande, pues el auditor 1 es el más eficiente en la ejecución de todos los proyectos obligatorios y en la solución del segundo heurístico constructivo sin modificación ya se habían asignado a ello. Sin embargo, si siguiéramos el orden inverso de auditores (primero el auditor 2, después el 1), la suma de beneficios coincidiría con el valor óptimo de 82 unidades. En las figuras 9 y 10 se pueden visualizar, en primer lugar, la solución obtenida con esta última versión del algoritmo y, en segundo, la proporcionada por el modelo MIP.

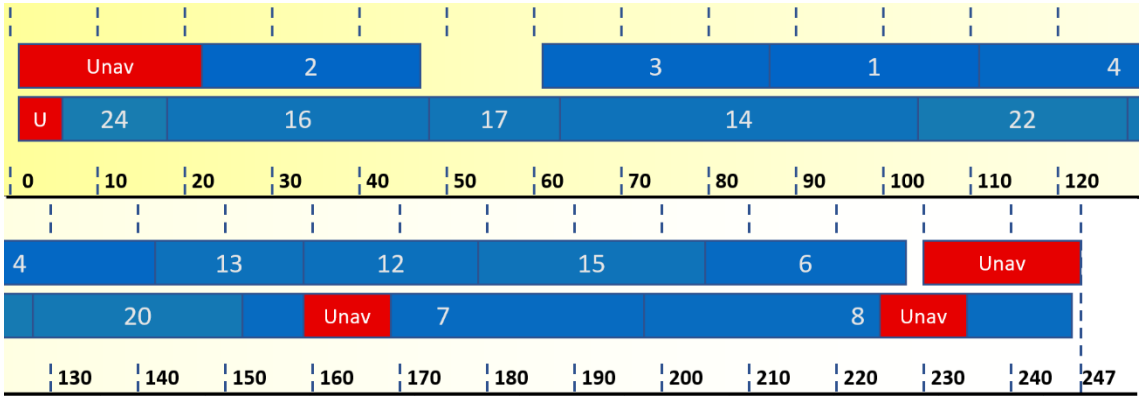


Figura 9 – Gráfico de Gantt de la solución del segundo heurístico constructivo (modificado) para una instancia del problema con 2 auditores y 24 proyectos.

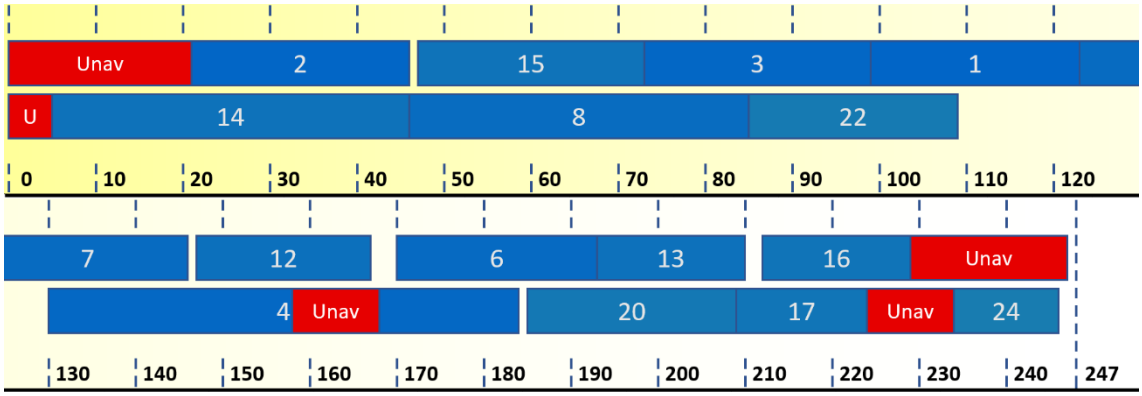


Figura 10 - Gráfico de Gantt de la solución del modelo MIP para una instancia del problema con 2 auditores y 24 proyectos.

En ambas soluciones son seleccionados los mismos 12 proyectos optativos (16 al total con los obligatorios), lo que implica que el valor de la función objetivo coincida. Sin embargo, se nota claramente que las asignaciones a los auditores y las secuenciaciones en el tiempo no son las mismas.

Es importante resaltar que esta instancia muy pequeña ha sido utilizada solamente para fines didácticos, es decir, para explicar el funcionamiento de los algoritmos. Los resultados encontrados no sirven para demostrar a superioridad o no del rendimiento de un heurístico respecto a otro. Para esto, es necesario probar los algoritmos con numerosas instancias de distintos tamaños y características y realizar un análisis estadístico de los resultados para detectar si existe evidencia suficiente para concluir que algún heurístico es superior a los demás.

6. Experimentos computacionales

Por medio del lenguaje de programación VBA para Excel, tal como se ha explicado con detalles en el capítulo 4, se ha generado aleatoriamente un conjunto de 72 instancias de prueba para los experimentos computacionales con las características que se muestran en la tabla 11.

Cantidad Auditores	Cantidad Proyectos (Obligatorios/Optativos)	Restricciones
5	70 (10/60)	1. Sin *release times* y sin indisponibilidades. 2. Con *release times* y sin indisponibilidades. 3. Sin *release times* y con indisponibilidades. 4. Con *release times* y con indisponibilidades.
	130 (10/120)	
10	110 (20/90)	
	200 (20/180)	
15	150 (30/120)	
	270 (30/240)	

Tabla 11 – Distribución de instancias de prueba para los experimentos computacionales.

Por lo tanto, tal y como se observa en la tabla 11, tenemos 6 cantidades distintas de auditores y proyectos, así como 4 configuraciones de restricciones posibles. Han sido generadas 3 instancias con distintos valores para cada una de estas 24 combinaciones, lo que resulta un total de 72 instancias de prueba. No se han probado instancias de tamaño más pequeño (o incluso más grande) porque el objetivo de este trabajo se centra en resolver instancias cuyos tamaños sean próximos a los observados en las situaciones reales. Es decir, no se ha enfocado en determinar el tamaño máximo de instancia que el modelo MIP es capaz de resolver óptimamente en el *timeout* configurado, sino en utilizarlo para determinar la calidad de las soluciones obtenidas por los heurísticos constructivos en la resolución de las instancias predefinidas. Respecto al problema real tratado en este trabajo, es más interesante la obtención de buenas soluciones de programación de los proyectos de auditoría en un tiempo muy corto que la obtención de una solución óptima de coste computacional muy elevado.

6.1. Resultados del modelo MIP

El modelo MIP presentado en la subsección 4.2.1 ha sido implementado con el IBM ILOG CPLEX *Optimization Studio*, versión 12.8.0.0, por medio del OPL (*Optimization Programming Language*), que es un lenguaje de modelización para optimización combinatoria que tiene por objetivo simplificar la resolución de problemas de este tipo. El conjunto de instancias ha sido resuelto con el modelo MIP en un ordenador móvil Dell XPS 13 (9370) con un procesador Intel i7-8550U, 16 GB de memoria RAM y sistema operativo Windows 10 de 64 bits.

Se ha configurado un *timeout* de 30 minutos en el CPLEX. Sin embargo, ni siquiera para las instancias de 5 auditores y 70 proyectos, las más pequeñas testadas, el CPLEX ha conseguido finalizar la ejecución de su algoritmo *branch and bound*. Lo que hace este algoritmo, resumidamente, es una búsqueda implícita en todo el espacio de soluciones factibles, cuyo proceso se puede representar por un árbol de decisiones. Según Broek (2009):

> Cada nodo del árbol corresponde a un subconjunto del conjunto total de soluciones factibles del problema. Inicialmente, el árbol consiste en un único nodo, denominado raíz, lo cual contiene todas las soluciones factibles. Una regla de *branching* (ramificación) especifica como el conjunto de soluciones factibles de un nodo es particionado en subconjuntos, cada uno correspondiendo a un nodo descendiente del nodo actual. Un ejemplo de regla de ramificación es el redondeo (*bound*), en la relajación lineal de un problema MIP, de las variables de decisiones no enteras correspondientes al nodo. La relajación lineal de un MIP es el programa lineal que resulta cuando se quitan las restricciones de integralidad del problema. (Broek, 2009, p. 5)

Los resultados encontrados por el modelo MIP se presentan en las tablas 12 y 13. La tabla 12 muestra las instancias para las cuales se ha conseguido obtener una solución entera factible además de una cota superior, mientras en la tabla 13 se encuentran las instancias para las cuales no se ha conseguido obtener una solución entera factible para un *timeout* de 30 minutos. Si una instancia tiene *release times*, se la presenta con la abreviación *"Rel"* (en caso contrario, se presenta como *"noRel"*). De semejante modo, si una instancia tiene indisponibilidades de los auditores, se la presenta con la abreviación *"Unav"* (en caso contrario, se presenta como *"noUnav"*). La columna "*Gap*" en las tablas se refiere a la desviación porcentual de la mejor solución entera encontrada por el modelo respecto al *Best Bound* (mejor cota).

Instancia	Suma Beneficios	Proyectos Totales	*Best Bound*	Gap
05Aud 70P noRel noUnav 1	224	46	226,4725	1,10%
05Aud 70P Rel noUnav 1	224	46	225,4887	0,66%
05Aud 70P noRel Unav 1	153	33	208,5257	36,29%
05Aud 70P Rel Unav 1	189	39	200,1006	5,87%
05Aud 70P noRel noUnav 2	216	38	219,1578	1,46%
05Aud 70P Rel noUnav 2	217	39	218,4158	0,65%
05Aud 70P noRel Unav 2	180	34	187,8937	4,39%
05Aud 70P Rel Unav 2	174	32	180,3724	3,66%

Instancia	Suma Beneficios	Proyectos Totales	*Best Bound*	Gap
05Aud 70P noRel noUnav 3	249	43	250,5412	0,62%
05Aud 70P Rel noUnav 3	248	44	249,6006	0,65%
05Aud 70P noRel Unav 3	213	39	218,5984	2,63%
05Aud 70P Rel Unav 3	192	34	207,4790	8,06%
05Aud 130P noRel noUnav 1	383	61	385,4447	0,64%
05Aud 130P Rel noUnav 1	381	61	385,2912	1,13%
05Aud 130P noRel Unav 1	300	50	347,9230	15,97%
05Aud 130P Rel Unav 1	293	49	333,2245	13,73%
05Aud 130P noRel noUnav 2	312	54	317,0300	1,61%
05Aud 130P Rel noUnav 2	313	55	316,2821	1,05%
05Aud 130P Rel Unav 2	248	46	278,4123	12,26%
05Aud 130P noRel noUnav 3	283	49	287,5618	1,61%
05Aud 130P Rel noUnav 3	283	49	287,2109	1,49%
05Aud 130P noRel Unav 3	239	43	257,6737	7,81%
05Aud 130P Rel Unav 3	235	41	248,2399	5,63%
10Aud 110P noRel noUnav 1	463	93	466,7807	0,82%
10Aud 110P Rel noUnav 1	460	92	466,4557	1,40%
10Aud 110P noRel Unav 1	9	23	430,2946	--
10Aud 110P noRel noUnav 2	428	88	433,3632	1,25%
10Aud 110P Rel noUnav 2	428	86	433,0941	1,19%
10Aud 110P noRel noUnav 3	457	87	462,2550	1,15%
10Aud 110P Rel noUnav 3	457	87	461,5651	1,00%
10Aud 200P noRel noUnav 1	173	43	636,2747	267,79%
10Aud 200P Rel noUnav 1	175	43	635,9018	263,37%
10Aud 200P noRel noUnav 2	354	68	589,7242	66,59%
10Aud 200P Rel noUnav 2	240	50	588,6010	145,25%
10Aud 200P noRel Unav 2	274	56	526,3271	92,09%
10Aud 200P Rel Unav 2	8	22	520,9844	--
10Aud 200P noRel noUnav 3	69	29	554,5601	703,71%
10Aud 200P Rel noUnav 3	268	58	554,0652	106,74%
10Aud 200P noRel Unav 3	243	51	489,6004	101,48%
15Aud 150P noRel noUnav 1	117	47	597,4918	410,68%
15Aud 150P Rel noUnav 1	205	61	597,1686	191,30%
15Aud 270P noRel noUnav 1	233	59	992,9913	326,18%
15Aud 270P Rel noUnav 1	403	81	992,5584	146,29%
15Aud 270P noRel noUnav 2	0	30	890,6023	--
15Aud 270P Rel noUnav 2	0	30	890,5459	--
15Aud 270P noRel noUnav 3	201	59	970,0028	--
15Aud 270P Rel noUnav 3	433	91	969,0030	--

Tabla 12 – Resultados encontrados por el modelo MIP para el conjunto de 72 instancias testadas (solo las instancias **con** soluciones enteras factibles).

Instancia	Suma Beneficios	Proyectos Totales	*Best Bound*	Gap
05Aud 130P noRel Unav 2	--	--	286,1120	--
10Aud 110P Rel Unav 1	--	--	424,7637	--
10Aud 110P noRel Unav 2	--	--	398,4021	--
10Aud 110P Rel Unav 2	--	--	387,7159	--
10Aud 110P noRel Unav 3	--	--	424,2064	--
10Aud 110P Rel Unav 3	--	--	415,5257	--
10Aud 200P noRel Unav 1	--	--	573,3460	--
10Aud 200P Rel Unav 1	--	--	569,2719	--
10Aud 200P Rel Unav 3	--	--	480,4239	--
15Aud 150P noRel Unav 1	--	--	562,3803	--
15Aud 150P Rel Unav 1	--	--	558,1730	--
15Aud 150P noRel noUnav 2	--	--	626,0000	--
15Aud 150P Rel noUnav 2	--	--	626,0000	--
15Aud 150P noRel Unav 2	--	--	622,2769	--
15Aud 150P Rel Unav 2	--	--	620,5323	--
15Aud 150P noRel noUnav 3	--	--	634,0000	--
15Aud 150P Rel noUnav 3	--	--	634,0000	--
15Aud 150P noRel Unav 3	--	--	624,9625	--
15Aud 150P Rel Unav 3	--	--	621,2896	--
15Aud 270P noRel Unav 1	--	--	896,5186	--
15Aud 270P Rel Unav 1	--	--	889,4009	--
15Aud 270P noRel Unav 2	--	--	801,4462	--
15Aud 270P Rel Unav 2	--	--	795,1098	--
15Aud 270P noRel Unav 3	--	--	884,1905	--
15Aud 270P Rel Unav 3	--	--	878,0345	--

Tabla 13 – Resultados encontrados por el modelo MIP para el conjunto de 72 instancias testadas (solo las instancias **sin** soluciones enteras factibles).

Para las instancias de 5 y 10 auditores se han encontrado soluciones enteras factibles para la mayoría de las instancias. Para las instancias de 5 auditores (tanto de 70 como de 130 proyectos) y para las instancias de 10 auditores y 110 proyectos se han encontrado soluciones enteras muy próximas de la cota superior de la relajación lineal (identificada como *Best Bound* en las tablas 12 y 13). Para las instancias de 10 auditores y 200 proyectos y de 15 auditores (150 o 270 proyectos), la calidad de las soluciones enteras encontradas por el CPLEX ha sido muy baja. Además, para las instancias más grandes, en la mayoría de los casos no se consigue encontrar una solución entera factible dentro del *timeout* configurado, principalmente para las que presentan restricciones de indisponibilidades de los auditores. Por lo tanto, no se ha adoptado como valor de referencia para $Best_{sol}$ – en la ecuación (4),

presentada a continuación – el valor de la solución entera factible encontrada, sino la parte entera del valor encontrado para el *Best Bound* por el CPLEX, tras 30 minutos de cómputo de la resolución del modelo MIP.

6.2. Resultados de los heurísticos

El lenguaje de programación elegido para la tarea de codificación de los heurísticos ha sido *Julia*, desarrollado por cuatro científicos, tres de ellos con origen en el *Massachussets Institute of Technology* (MIT). De acuerdo con Bezanson, Karpinski, Shah y Edelman (2012), Julia es un lenguaje dinámico flexible, apropiado para computación numérica y científica, con rendimiento comparable a lo de un lenguaje estáticamente tipado como C. Antes de la codificación propiamente dicha de los heurísticos, ha sido necesario definir las estructuras de almacenamiento de los datos del problema, los cuales han sido importados de los ficheros Excel por medio del paquete *ExcelReaders*, de Anthoff (2018).

En esta sección, para medir la calidad de las soluciones obtenidas por cada algoritmo propuesto, se calcula el porcentaje de desviación relativa (RPD, por su denominación en inglés, *Relative Percentage Deviation*) entre los resultados obtenidos por los heurísticos y por el modelo MIP, lo cual se calcula según la siguiente fórmula (Yepes Borrero, 2017):

$$RPD = \left(\frac{Best_{sol} - Method_{sol}}{Best_{sol}}\right) \times 100 \qquad (4)$$

La variable $Best_{sol}$ se refiere a la parte entera del valor de la cota superior encontrada para la función objetivo (la suma de beneficios), tras la relajación lineal del problema. $Method_{sol}$ es el valor de la función objetivo encontrado por cada uno de los heurísticos analizados.

Como se ha explicado en el capítulo 5, en el segundo heurístico (y también en su versión modificada) se debe elegir un orden de llenado de los calendarios de los auditores, lo cual puede producir modificaciones en las soluciones encontradas y, consecuentemente, en los valores de la función objetivo. Por ello, se ha decidido probar el segundo heurístico modificado con 3 variantes.

En la primera variante, la cual es referenciada adelante por la abreviación "2nd Mod", se ha tomado el orden natural de los auditores (*i* variando de 1 a *m*), del mismo modo que se ha hecho para el algoritmo original del segundo heurístico, identificado como "2nd Heur".

En la segunda variante se ha añadido al código una ordenación de los auditores según su rendimiento general respecto a los tiempos de ejecución de los proyectos. La ordenación ha sido hecha para que se iniciara el procedimiento de asignación de los proyectos por los auditores con peor rendimiento general. A esta versión se le denomina "2nd Mod W", donde la letra "W" se refiere a palabra en inglés "*worst*" (peor).

Por fin, en la tercera variante del segundo heurístico modificado, se ha probado la elección aleatoria del orden de llenado de los calendarios de los auditores, por medio de la utilización de la función *shuffle* de la librería *Random*, nativa del lenguaje Julia. A esta versión se le denomina "2nd Mod R", donde la letra "R" se refiere a palabra en inglés "*random*" (aleatorio).

Luego, en resumen, se ha probado la resolución del conjunto de 72 instancias por cinco algoritmos distintos: i) "1st Heur" (primer heurístico); ii) "2nd Heur" (segundo heurístico original, con el orden natural para el llenado de los calendarios de los auditores); iii) "2nd Mod" (segundo heurístico modificado, con el orden natural para el llenado de los calendarios de los auditores); iv) "2nd Mod W" (segundo heurístico modificado, con la priorización del llenado de los calendarios de los auditores de peor rendimiento general respecto a los tiempos de ejecución); y v) "2nd Mod R" (segundo heurístico modificado, con la elección de un orden aleatorio de los auditores para el llenado de sus calendarios).

Los resultados encontrados para el RPD de los cinco algoritmos son presentados en la tabla siguiente (los mejores resultados en negrito y los peores en letra cursiva):

Instancia	1st Heur	2nd Heur	2nd Mod	2nd Mod W	2nd Mod R
05Aud 70P noRel noUnav 1	7,52	7,96	*8,41*	7,96	**7,08**
05Aud 70P Rel noUnav 1	9,33	*10,22*	*10,22*	9,78	**8,44**
05Aud 70P noRel Unav 1	**6,73**	9,13	8,17	7,69	*10,10*
05Aud 70P Rel Unav 1	*9,00*	*9,00*	**8,00**	8,50	**8,00**
05Aud 70P noRel noUnav 2	14,16	*20,55*	9,59	9,59	**8,68**
05Aud 70P Rel noUnav 2	*18,81*	11,01	9,17	10,09	**8,72**
05Aud 70P noRel Unav 2	13,90	*24,06*	9,09	8,02	**6,42**
05Aud 70P Rel Unav 2	*17,22*	12,78	10,00	**8,89**	11,11
05Aud 70P noRel noUnav 3	*17,60*	10,80	13,20	**9,20**	10,80
05Aud 70P Rel noUnav 3	*18,47*	12,05	14,06	**10,84**	11,65
05Aud 70P noRel Unav 3	*21,56*	11,47	19,27	**11,01**	16,97
05Aud 70P Rel Unav 3	*22,22*	**8,21**	17,39	12,56	14,98
05Aud 130P noRel noUnav 1	*13,25*	12,73	6,75	**5,71**	7,79

Instancia	1st Heur	2nd Heur	2nd Mod	2nd Mod W	2nd Mod R
05Aud 130P Rel noUnav 1	*18,70*	10,39	10,65	**9,35**	9,87
05Aud 130P noRel Unav 1	*23,34*	12,39	8,65	7,78	**5,48**
05Aud 130P Rel Unav 1	*21,92*	**9,31**	10,21	**9,31**	9,91
05Aud 130P noRel noUnav 2	*12,30*	11,67	**4,73**	8,52	11,04
05Aud 130P Rel noUnav 2	*18,35*	10,13	**7,28**	8,23	7,59
05Aud 130P noRel Unav 2	*19,23*	10,14	**8,39**	11,89	10,84
05Aud 130P Rel Unav 2	*20,86*	15,47	**7,19**	9,71	9,35
05Aud 130P noRel noUnav 3	*19,51*	11,85	11,85	**9,76**	11,85
05Aud 130P Rel noUnav 3	*19,86*	12,89	**9,06**	**9,06**	10,10
05Aud 130P noRel Unav 3	*24,12*	**11,67**	**11,67**	12,45	12,45
05Aud 130P Rel Unav 3	*20,97*	10,89	**7,26**	7,66	10,89
10Aud 110P noRel noUnav 1	7,94	*9,66*	7,51	6,22	**5,58**
10Aud 110P Rel noUnav 1	*8,58*	*8,58*	**8,15**	**8,15**	*8,58*
10Aud 110P noRel Unav 1	8,60	*12,56*	7,67	**7,44**	9,07
10Aud 110P Rel Unav 1	10,38	*11,08*	10,38	**8,96**	9,67
10Aud 110P noRel noUnav 2	**7,16**	*10,16*	9,70	8,31	8,31
10Aud 110P Rel noUnav 2	*13,63*	11,32	11,32	**9,70**	10,85
10Aud 110P noRel Unav 2	*15,08*	11,06	13,07	**10,05**	11,06
10Aud 110P Rel Unav 2	*14,73*	13,70	13,44	**10,08**	12,66
10Aud 110P noRel noUnav 3	8,01	*11,04*	8,87	8,87	**7,79**
10Aud 110P Rel noUnav 3	11,28	*11,50*	**9,54**	10,20	10,41
10Aud 110P noRel Unav 3	10,61	*17,69*	**9,43**	12,26	11,32
10Aud 110P Rel Unav 3	13,49	*16,14*	**11,57**	12,53	12,53
10Aud 200P noRel noUnav 1	*13,99*	8,18	9,43	**7,08**	8,65
10Aud 200P Rel noUnav 1	*12,76*	10,08	10,24	9,61	**8,03**
10Aud 200P noRel Unav 1	*12,04*	**9,08**	*12,04*	**9,08**	9,77
10Aud 200P Rel Unav 1	*15,64*	11,78	10,37	**9,31**	9,84
10Aud 200P noRel noUnav 2	*12,39*	11,88	10,36	8,83	**7,64**
10Aud 200P Rel noUnav 2	*15,99*	11,05	**10,54**	10,88	11,05
10Aud 200P noRel Unav 2	**11,41**	11,60	*13,31*	11,60	11,60
10Aud 200P Rel Unav 2	*20,58*	13,08	**10,96**	11,92	11,73
10Aud 200P noRel noUnav 3	*11,37*	9,57	**6,86**	7,76	9,57
10Aud 200P Rel noUnav 3	*18,59*	**8,84**	9,21	10,83	11,55
10Aud 200P noRel Unav 3	*19,63*	11,66	10,22	11,66	**10,02**
10Aud 200P Rel Unav 3	*17,50*	10,83	**9,58**	12,71	12,08
15Aud 150P noRel noUnav 1	**4,52**	*8,04*	7,54	5,86	5,86
15Aud 150P Rel noUnav 1	7,54	*9,05*	8,04	7,54	**6,70**
15Aud 150P noRel Unav 1	**7,83**	*10,50*	9,61	8,01	9,25
15Aud 150P Rel Unav 1	11,11	*13,62*	9,86	9,50	**8,96**
15Aud 150P noRel noUnav 2	**0,64**	*3,67*	1,76	1,92	2,88
15Aud 150P Rel noUnav 2	*3,51*	*3,51*	3,35	**3,04**	3,19
15Aud 150P noRel Unav 2	6,59	7,72	**6,27**	6,91	*8,36*
15Aud 150P Rel Unav 2	8,23	*9,19*	8,23	8,06	**7,42**

Instancia	1st Heur	2nd Heur	2nd Mod	2nd Mod W	2nd Mod R
15Aud 150P noRel noUnav 3	1,10	*3,31*	**0,79**	2,05	*3,31*
15Aud 150P Rel noUnav 3	3,63	3,79	**2,52**	3,00	*4,73*
15Aud 150P noRel Unav 3	**5,45**	*10,42*	6,73	7,21	7,21
15Aud 150P Rel Unav 3	8,70	*9,66*	7,09	8,37	**6,60**
15Aud 270P noRel noUnav 1	12,10	*13,00*	9,27	**7,16**	8,87
15Aud 270P Rel noUnav 1	*13,61*	13,00	11,19	9,48	**8,67**
15Aud 270P noRel Unav 1	13,28	*13,50*	10,83	9,71	**9,15**
15Aud 270P Rel Unav 1	*15,64*	14,62	11,92	**10,46**	13,05
15Aud 270P noRel noUnav 2	*12,13*	11,01	8,54	**7,75**	8,88
15Aud 270P Rel noUnav 2	*15,62*	10,79	9,33	**7,87**	8,65
15Aud 270P noRel Unav 2	*12,73*	11,24	10,61	10,11	**9,99**
15Aud 270P Rel Unav 2	*14,47*	12,70	9,94	10,06	**9,81**
15Aud 270P noRel noUnav 3	*9,90*	*9,90*	**6,91**	7,73	7,73
15Aud 270P Rel noUnav 3	*13,73*	10,01	**7,74**	9,29	9,08
15Aud 270P noRel Unav 3	*14,03*	11,31	**9,05**	9,50	10,52
15Aud 270P Rel Unav 3	*18,68*	11,50	10,36	**10,25**	11,05
Cantidad Veces **MIN** RPD	7	*5*	22	**24**	18
Cantidad Veces **MAX** RPD	*45*	25	2	**0**	5
Promedio Todas Instancias	*13,27*	11,03	9,33	**8,87**	9,30
Promedio Instancias REL UNAV	*15,63*	11,87	10,21	**9,94**	10,54

Tabla 14 – Resultados encontrados por los heurísticos para el conjunto de 72 instancias testadas.

Se nota en la tabla 14 que el algoritmo de mejor rendimiento ha sido el segundo heurístico modificad con la priorización de los peores auditores en el orden de llenado de los calendarios ("2nd Mod W"). Este algoritmo ha sido el mejor en todos los criterios: i) es el que más veces ha presentado el valor mínimo del RPD; ii) no ha tenido el peor (máximo) RPD en la resolución de ninguna de las 72 instancias; y iii) es el de menor RPD promedio tanto cuando se consideran todos los cuatro tipos de instancias respecto a las restricciones de *release times* e indisponibilidades, como cuando se toman en cuenta solamente las instancias que presentan los dos tipos de restricciones. En el otro extremo, el primero heurístico ha presentado el peor rendimiento en 3 de los 4 criterios observados, habiendo sido superior solo al segundo heurístico original respecto a la cantidad de veces en que ha presentado el menor valor para el RPD. Cuando se consideran solamente las instancias con los dos tipos de restricciones, se tiene que el valor del RPD promedio para el primer heurístico es un 57,24% más grande (peor) que el RPD promedio obtenido por el segundo heurístico modificado de mejor rendimiento.

Con relación al segundo heurístico modificado con el orden natural del llenado de los calendarios ("2nd Mod") y con el orden aleatorio ("2nd Mod R"), se ve que sus rendimientos han sido muy parecidos, lo que nos lleva a creer que el orden del procedimiento de programación en este algoritmo no es significativo para el valor del RPD promedio alcanzado tras la resolución de numerosas instancias distintas. Sin embargo, para que se pueda validar esta inferencia, es necesaria la realización de un análisis de variancia con el objetivo de verificar si existe una diferencia estadística significativa entre los RPDs promedios obtenidos por cada algoritmo. El resultado del ANOVA se presenta en la tabla siguiente:

Fuente	*Suma de Cuadrados*	*Gl*	*Cuadrado Medio*	*Razón-F*	*Valor-P*
Entre grupos	958,33	4	239,584	20,00	0,0000
Intra grupos	4.252,64	355	11,979		
Total (Corr.)	5.210,97	359			

Tabla 15 – ANOVA para RPD por Algoritmo.

Como la razón-*F* calculada es demasiado grande para ser una *F* de Fisher con los grados de libertad correspondientes (4 y 355), lo que representa un *p*-valor muy inferior al riesgo de 1ª especie definido (0,05), se debe rechazar la hipótesis nula de que las medias de todos los algoritmos son iguales. Para que se pueda determinar cuáles medias son significativamente diferentes de otras, es necesario ejecutarse la prueba de múltiples rangos, presentada en las dos tablas a continuación:

Algoritmo	*Casos*	*Media*	*Grupos Homogéneos*
2nd Mod W	72	8,86722	X
2nd Mod R	72	9,29708	X
2nd Mod	72	9,32653	X
2nd Heur	72	11,0271	X
1st Heur	72	13,265	X

Tabla 16 – Prueba de múltiples rangos para RPD por Algoritmo con 95,0 porcentaje Tukey HSD (identificación de los grupos homogéneos).

Contraste	*Sig.*	*Diferencia*	*+/- Límites*
1st Heur - 2nd Heur	*	2,23792	1,58154
1st Heur - 2nd Mod	*	3,93847	1,58154
1st Heur - 2nd Mod R	*	3,96792	1,58154
1st Heur - 2nd Mod W	*	4,39778	1,58154
2nd Heur - 2nd Mod	*	1,70056	1,58154
2nd Heur - 2nd Mod R	*	1,73	1,58154
2nd Heur - 2nd Mod W	*	2,15986	1,58154
2nd Mod - 2nd Mod R		0,0294444	1,58154

2nd Mod - 2nd Mod W		0,459306	1,58154
2nd Mod R - 2nd Mod W		0,429861	1,58154

***indica una diferencia significativa.**

Tabla 17 – Prueba de múltiples rangos para RPD por Algoritmo con 95,0 porcentaje Tukey HSD (contrastes entre los pares de medias).

A partir de los resultados presentados, se puede inferir que el primer heurístico, el segundo heurístico original y el segundo heurístico modificado sí que poseen diferencias estadísticas significativas entre los valores medios obtenidos para el RPD. Sin embargo, lo mismo no se puede concluir acerca de las tres variantes del segundo heurístico modificado, pues los resultados de la prueba de múltiples rangos indican que las diferencias entre sus medias para el RPD no son estadísticamente significativas a un nivel de 95%. Se presenta a continuación el gráfico de medias del RPD por algoritmo para el test de Tukey – un método bastante conservador – con riesgo de primera especie del 5%:

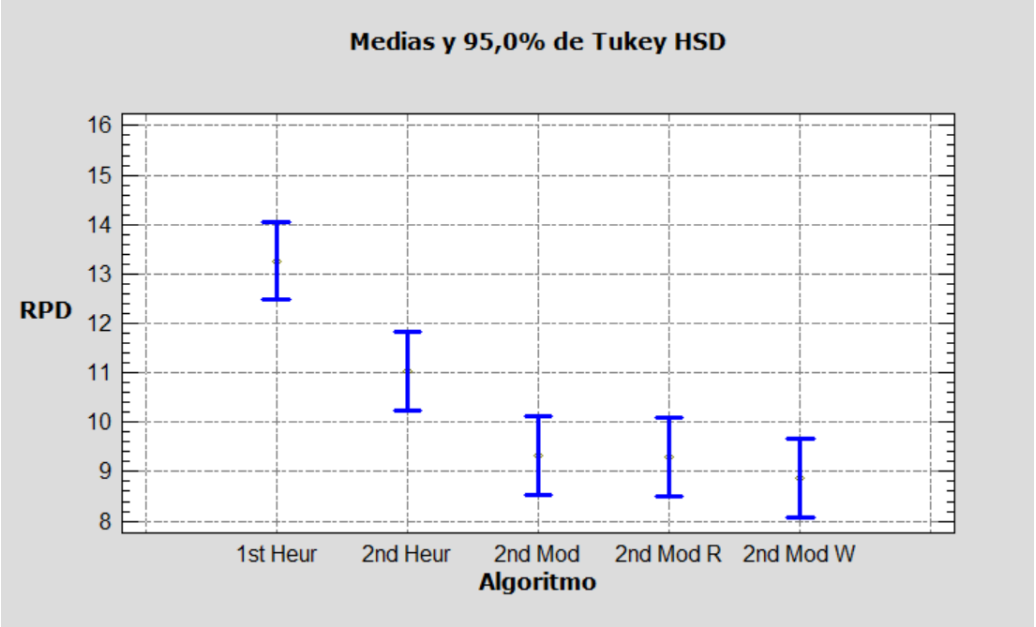


Figura 11 – Gráfico de Medias del RPD por Algoritmo para el test de Tukey a un nivel de 95% de confianza.

En este gráfico se puede visualizar claramente que los intervalos de confianza para los RPDs promedios de las tres versiones del segundo heurístico modificado se solapan, en que pese observarse el mejor rendimiento del algoritmo "2nd Mod W". Por otro lado, no existe solape de intervalos de medias entre el grupo de algoritmos del segundo heurístico modificado y el segundo heurístico original, ni entre estos dos tipos y el primer heurístico.

Confirmando lo anticipado anteriormente, también se observa que los intervalos de medias entre los algoritmos "2nd Mod" y "2nd Mod R" son prácticamente idénticos, lo que demuestra no haber diferencia entre fijar un orden específico y elegir cualquiera al azar en el procedimiento de llenado de los calendarios de los auditores en el segundo heurístico modificado.

Los tres supuestos en que se basa el ANOVA son: la normalidad de los residuos, la igualdad de varianzas entre los grupos del factor estudiado y la independencia de las muestras. Por lo tanto, no sería prudente confiar en los resultados del modelo sin antes verificar la validad de sus supuestos para los datos analizados (Montgomery, 2017). El tercer supuesto ya se ha garantizado por la aleatoriedad en la generación de las instancias de prueba. Respecto a la normalidad de los residuos del modelo, se ha utilizado el gráfico de probabilidad normal, presentado a continuación, para verificarla:

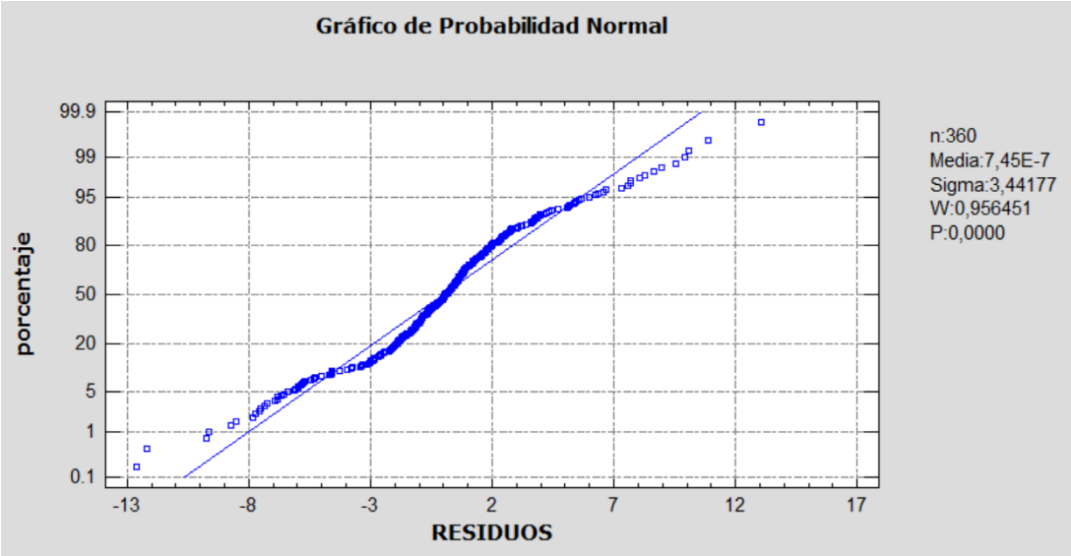


Figura 12 – Gráfico de Probabilidad Normal para los residuos del ANOVA.

Se nota que los residuos se distancian un poco de una distribución normal, por lo que no se ha podido comprobarla. Sin embargo, ya se ha demostrado en varios artículos, como en lo de Blanca, M., Alarcón, R., Arnau, J., Bono, R., y Bendayan, R. (2017), que la prueba de la función *F* de Fisher en que se basa el modelo ANOVA es suficientemente robusta frente al no cumplimiento del supuesto de normalidad de los datos.

Respecto a la homogeneidad de las varianzas entre los grupos, como la media aritmética de los cuadrados de los residuos tiende a ser igual a la varianza cuando la cantidad

de muestras es suficientemente grande, se ha hecho un ANOVA de los cuadrados de los residuos del modelo – como lo sugiere Romero Villafranca, R., y Zúnica Ramajo, L. R. (2013) – para examinarse el cumplimiento de este supuesto:

Fuente	*Suma de Cuadrados*	*Gl*	*Cuadrado Medio*	*Razón-F*	*Valor-P*
Entre grupos	28.757,7	4	7.189,43	14,78	0,0000
Intra grupos	172.665,5	355	486,38		
Total (Corr.)	201.422,2	359			

Tabla 18 – ANOVA para $(Residuos)^2$ por Algoritmo.

Del análisis de la tabla, puesto que el *p*-valor de la prueba-*F* es menor que el nivel de significancia estadística (0,05), se debe rechazar la hipótesis nula de que sean iguales todas las medias de los cuadrados de los residuos. Para que se pueda determinar cuáles medias son significativamente diferentes de otras, es necesario ejecutarse más una vez la prueba de múltiples rangos, la cual se presenta en las tablas siguientes:

Algoritmo	*Casos*	*Media*	*Grupos Homogéneos*
2nd Mod W	72	4,94318	X
2nd Mod R	72	6,1263	X
2nd Mod	72	8,52395	X
2nd Heur	72	10,1561	X
1st Heur	72	29,3149	X

Tabla 19 – Prueba de múltiples rangos para $(Residuos)^2$ por Algoritmo con 95,0 porcentaje Tukey HSD (identificación de los grupos homogéneos).

Contraste	*Sig.*	*Diferencia*	*+/- Límites*
1st Heur - 2nd Heur	*	19,1588	10,0775
1st Heur - 2nd Mod	*	20,7909	10,0775
1st Heur - 2nd Mod R	*	23,1886	10,0775
1st Heur - 2nd Mod W	*	24,3717	10,0775
2nd Heur - 2nd Mod		1,63217	10,0775
2nd Heur - 2nd Mod R		4,02982	10,0775
2nd Heur - 2nd Mod W		5,21294	10,0775
2nd Mod - 2nd Mod R		2,39765	10,0775
2nd Mod - 2nd Mod W		3,58077	10,0775
2nd Mod R - 2nd Mod W		1,18312	10,0775

Tabla 20 – Prueba de múltiples rangos para $(Residuos)^2$ por Algoritmo con 95,0 porcentaje Tukey HSD (contrastes entre los pares de medias).

Por los resultados del ANOVA de los cuadrados de los residuos, se puede inferir que el primer heurístico presenta una variancia significativamente distinta de los demás

algoritmos, lo que también se puede visualizar en el gráfico de medias presentado a continuación:

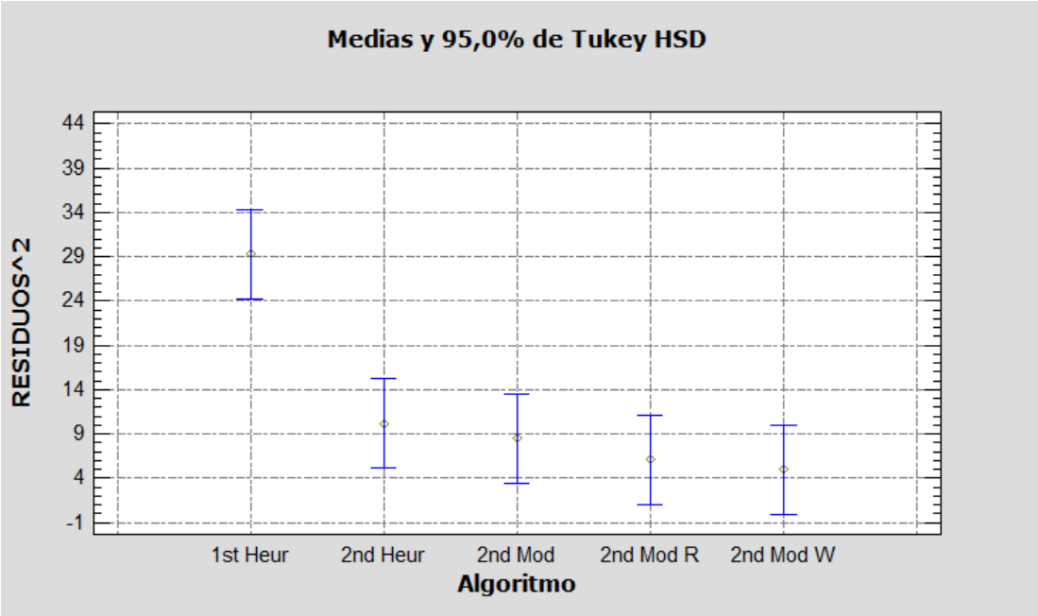


Figura 13 – Gráfico de Medias para (*Residuos*)^2 por Algoritmo para el test de Tukey a un nivel de 95% de confianza.

Por lo tanto, también no se ha podido comprobar el supuesto de homogeneidad de las varianzas entre los grupos. Sin embargo, según Montgomery (2017), en los modelos de efectos fijos con los mismos tamaños de muestra en todos los tratamientos, la prueba-*F* es apenas ligeramente afectada. Además, en Blanca, M. J., Alarcón, R., Arnau, J., Bono, R., y Bendayan, R. (2018) también se ha comprobado la robustez de la prueba-*F* en el caso de que los tamaños muestrales sean iguales para los distintos grupos, independiente del tamaño total de la muestra y del *ratio* de varianzas. Por todo lo expuesto, se concluye que no se deben despreciar los resultados encontrados por el ANOVA. De todo modo, se ha decidido compararlos con los resultados de la prueba de Kruskal-Wallis, presentados en las dos tablas siguientes:

Algoritmo	***Tamaño Muestra***	***Rango Promedio***
1st Heur	72	244,007
2nd Heur	72	216,174
2nd Mod	72	152,958
2nd Mod R	72	152,826
2nd Mod W	72	136,535

Estadístico = 58,2594 Valor-P = 6,73173E-12

Tabla 21 – Prueba de Kruskal-Wallis para RPD por Algoritmo (resultado general).

Contraste	*Sig.*	*Diferencia*	*+/- Límites*
1st Heur - 2nd Heur		27,8333	48,6868
1st Heur - 2nd Mod	*	91,0486	48,6868
1st Heur - 2nd Mod R	*	91,1806	48,6868
1st Heur - 2nd Mod W	*	107,472	48,6868
2nd Heur - 2nd Mod	*	63,2153	48,6868
2nd Heur - 2nd Mod R	*	63,3472	48,6868
2nd Heur - 2nd Mod W	*	79,6389	48,6868
2nd Mod - 2nd Mod R		0,131944	48,6868
2nd Mod - 2nd Mod W		16,4236	48,6868
2nd Mod R - 2nd Mod W		16,2917	48,6868

***indica una diferencia significativa.**

Tabla 22 – Prueba de Kruskal-Wallis para RPD por Algoritmo (comparaciones por pares entre los rangos promedios de los 5 grupos).

Se observa que los resultados de la prueba de Kruskal-Wallis respecto a la comparación de las medias entre los pares de algoritmos son muy semejantes a los encontrados por el ANOVA, excepto por la comparación entre el primer heurístico y el segundo heurístico original, pues, al contrario del ANOVA, la prueba de Kruskal-Wallis ha apuntado que no existe diferencia estadística significativa, al nivel de 95% de confianza, entre las medias de la variable RPD para estos dos algoritmos.

En Hecke (2012), se demuestra que la potencia[(5)] de la prueba no paramétrica de Kruskal-Wallis es más grande que la del ANOVA para distribuciones de datos asimétricas. Respecto a la variable RPD analizada en este estudio, se ha detectado un sesgo estandarizado igual a 6,32, lo que representa una asimetría positiva. Sin embargo, como el ANOVA ha aceptado la hipótesis alternativa de la diferencia de medias del RPD entre el primer heurístico y el segundo heurístico original, mientras la prueba de Kruskal-Wallis la ha rechazado, se tiene que el error con lo cual se debería preocupar aquí no es lo del tipo II, sino lo del tipo I[(6)]. Luego, con base en lo que ya se ha comentado anteriormente acerca de las conclusiones obtenidas por Blanca et al (2018), se ha decidido considerar válido el resultado señalado por el ANOVA acerca de la existencia de una diferencia estadísticamente significativa entre las medias del RPD de estos dos algoritmos.

Respecto a las medias del RPD para el segundo heurístico modificado, queda muy claro que son significativamente más pequeñas que las obtenidas tanto por el primer

[(5)] La potencia de una prueba estadística es la probabilidad de que la hipótesis nula sea rechazada cuando la hipótesis alternativa es verdadera, es decir, de no cometer un error del tipo II.
[(6)] El error del tipo I se refiere al riesgo de rechazar incorrectamente la hipótesis nula cuando esta es verdadera.

heurístico como por el segundo heurístico original. Por lo tanto, se pude concluir por la superioridad de rendimiento del último algoritmo creado en este trabajo. Finalmente, con relación a las variantes de la selección del orden de llenado de los calendarios de los auditores en el segundo heurístico modificado, se ha comprobado que no han influido de modo estadísticamente significativo en los resultados obtenidos para la variable RPD.

Cabe resaltar que se ha probado la transformación de los datos de la variable respuesta RPD por medio de la aplicación de operadores matemáticos como el logaritmo neperiano o la raíz cuadrada, pero sin éxito en la resolución de los problemas de no normalidad y heterocedasticidad. También se han realizado ANOVAs multifactoriales para verificar la influencia de las restricciones del problema (*release times* e indisponibilidades) en los resultados obtenidos por los algoritmos. En las tablas 23 y 24 se observa que ambas las restricciones tienen un efecto estadísticamente significativo sobre el valor del RPD, siendo despreciables los efectos de sus interacciones con el factor "algoritmo". Las pruebas de múltiples rangos en los ANOVAs multifactoriales han llegado en resultados semejantes a los presentados en la tabla 17, razón por la cual no han sido transcritos a continuación.

Fuente	***Suma de Cuadrados***	***Gl***	***Cuadrado Medio***	***Razón-F***	***Valor-P***
EFECTOS PRINCIPALES					
A:Algoritmo	958,335	4	239,584	21,67	0,0000
B:Indispon	369,441	1	369,441	33,41	0,0000
INTERACCIONES					
AB	12,9038	4	3,22596	0,29	0,8833
RESIDUOS	3.870,29	350	11,058		
TOTAL (CORREGIDO)	5.210,97	359			

Tabla 23 – ANOVA Multifactorial (RPD por algoritmo y presencia de indisponibilidades).

Fuente	***Suma de Cuadrados***	***Gl***	***Cuadrado Medio***	***Razón-F***	***Valor-P***
EFECTOS PRINCIPALES					
A:Algoritmo	958,335	4	239,584	20,43	0,0000
B:Release	70,3222	1	70,3222	6,00	0,0148
INTERACCIONES					
AB	77,1375	4	19,2844	1,64	0,1627
RESIDUOS	4.105,18	350	11,7291		
TOTAL (CORREGIDO)	5.210,97	359			

Tabla 24 – ANOVA Multifactorial (RPD por algoritmo y presencia de *release times*).

En cuanto a los tiempos computacionales utilizados por los algoritmos, se ha verificado, por supuesto, su dependencia del tamaño de la instancia resuelta, como se puede ver en la figura 14. Las tres variantes del segundo heurístico constructivo modificado han

sido consideradas conjuntamente, pues no hay diferencias significativas entre ellas respecto al tiempo de ejecución.

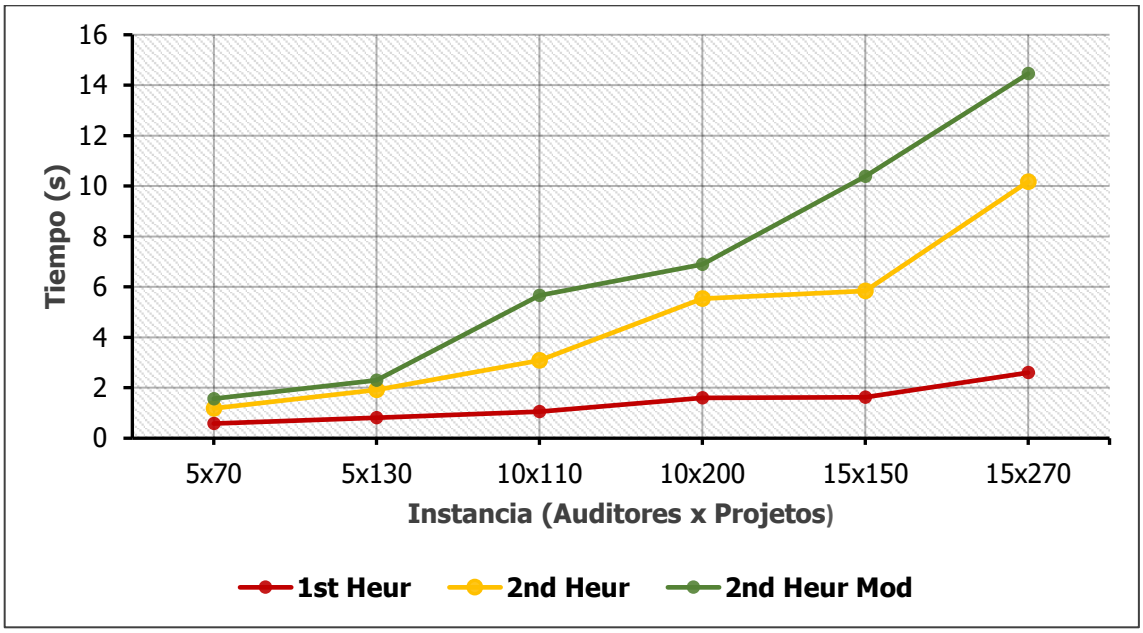


Figura 14 – Tiempos computacionales medios de ejecución por tamaño de instancia para los tres tipos de heurísticos.

Se nota claramente en la figura 14 que el tiempo de ejecución del segundo heurístico crece más rápidamente conforme se incrementa el tamaño de las instancias, pero, aun así, los valores de los tiempos son significativamente bajos (cuando se compara con el tiempo requerido por el modelo MIP), incluso para las instancias más grandes testadas. De hecho, se ha verificado que los valores promedios de los tiempos de computación utilizados por el primer heurístico, por el segundo heurístico original y por el segundo heurístico modificado en la resolución de las instancias más grandes del problema (15 auditores y 270 proyectos) han sido, respectivamente, de 2600, 10174 y 14452 milisegundos. Estos valores consideran solamente el tiempo de ejecución de las líneas de código relativas a los algoritmos heurísticos propiamente dichos, es decir, no incluyen los tiempos utilizados en la preparación del ambiente computacional, la cual comprende la importación de las librerías de código utilizadas y de los datos de los ficheros Excel, la creación de las estructuras de almacenamiento y la declaración de las funciones utilizadas por el heurístico. Respecto a estos últimos, se han observado valores muy similares para todos los heurísticos codificados, entre 15 y 20 segundos. Considerando los tiempos totales, se tiene que todas las instancias han sido resueltas por los heurísticos en menos de 40 segundos.

7. Conclusiones y futuras líneas de investigación

En este trabajo se han aplicado con éxito las técnicas de investigación operativa, más específicamente las relacionadas a la programación de la producción, para modelizar un problema real de asignación y secuenciación de proyectos de auditoría en el órgano de control interno del Poder Ejecutivo Federal de Brasil. Se ha demostrado que, al considerarse cada auditor como una máquina y cada proyecto de auditoría como un trabajo por asignar y secuenciar, el problema identificado podría ser tratado como un problema de máquinas en paralelo no relacionadas, con restricciones de *release times* e indisponibilidades y una función objetivo de maximizar la suma de beneficios asociados a la realización de los proyectos optativos por seleccionar en un conjunto previamente determinado.

Se ha identificado el estado del arte en la modelización matemática de este problema de secuenciación con estos tipos de restricciones para, tras la adaptación e incorporación de los modelos encontrados, construir un modelo de programación entera mixta que pudiera representar perfectamente el problema al que se enfrenta en el mundo real. Tras encontrar que el CPLEX no ha sido capaz de resolver óptimamente el modelo MIP en un tiempo de computación considerado razonable para ninguna de las instancias del conjunto de prueba, se han desarrollado dos algoritmos heurísticos constructivos por medio del lenguaje *Julia* con el objetivo de encontrar buenas soluciones en un tiempo computacional bastante inferior. El segundo heurístico desarrollado ha sido modificado con el objetivo de garantizarse que los proyectos obligatorios fueran asignados solamente a los mejores auditores y, además, se ha probado en 3 variantes, para poder detectar la existencia de resultados significativamente distintos para los valores del RPD tras el cambio del orden elegido para el llenado del calendario de los auditores.

Al final, por medio de un ANOVA de la variable RPD por algoritmo, se ha comprobado que el primer heurístico, el segundo heurístico original y el segundo heurístico modificado presentan diferencias estadísticamente significativas, con un error de primera especie del 5%, para las medias del RPD. Respecto a las variantes del orden del llenado de los calendarios de los auditores en el segundo heurístico modificado, se ha comprobado que no afectan de modo estadísticamente significativo el valor promedio encontrado para el RPD tras la resolución de varias instancias del problema.

Como la diferencia de los tiempos computacionales utilizados por las tres clases de algoritmos es relativamente pequeña en términos absolutos (el segundo heurístico modificado, en el peor de los casos, gasta cerca de 12 segundos más que el primer heurístico, que es el más rápido), se concluye que, en el caso de que tuviéramos que elegir un único algoritmo para la resolución del problema, nos quedaríamos con el segundo heurístico modificado.

Tras el éxito de la modelización y resolución de este problema, se espera que la aplicación práctica de los métodos aquí desarrollados pueda, de hecho, contribuir para que la programación de los proyectos de auditoría en las unidades de la CGU sea hecha de modo más eficiente, con mejor aprovechamiento de la fuerza de trabajo disponible y maximizando la entrega de beneficios no solo para la Administración Pública del Brasil, sino para toda la sociedad de este país.

Por fin, respecto a las futuras líneas de investigación, se tiene que el problema tratado en este trabajo podría continuar investigándose de diversas maneras, por ejemplo:

1. Desarrollando algoritmos de búsqueda local o empleando metaheurísticas (como búsqueda tabú, algoritmos genéticos u otros) para intentar mejorar las soluciones obtenidas por los heurísticos constructivos propuestos en este trabajo.
2. Eliminándose la restricción de que cada proyecto de auditoría debe ser realizado por solamente un único auditor. En este caso, se debería considerar las eficiencias de cada auditor al trabajar en conjunto con los demás, en un escenario de múltiples combinaciones, dependiendo de los tamaños permitidos para los equipos.
3. Considerándose el problema de secuenciación de las tareas que componen los proyectos, con sus posibles relaciones de dependencia. De este modo, la programación actuaría en un nivel inferior, incluso con la posibilidad de bajar aún más, llegando al nivel de subtareas (de la misma manera como ya se ejecuta el trabajo en la CGU). En este caso sería recomendable modelizar el problema como un RCPSP.
4. Eliminándose la restricción de que un proyecto de auditoría solo puede ser interrumpido por un único intervalo de disponibilidad, lo que exigiría la

reformulación del modelo matemático. También se podría quitar el límite máximo de tres indisponibilidades impuesto en este trabajo.

5. Considerándose las indisponibilidades de los auditores no solo como restricciones, sino como variables de decisión, es decir, el gestor debería decidir acerca de la programación de las vacaciones y de las demás licencias de los auditores de su unidad analizando el impacto que tendrían en la suma de beneficios obtenida tras la programación anual de los proyectos de auditoría.

Las opciones enumeradas, que no son exhaustivas, pueden ser de gran relevancia como temas para una futura tesis doctoral. Además, se debe considerar la posibilidad de redactar un artículo científico sobre el problema tratado en este trabajo.

Referencias

Anthoff, D. (2016). ExcelReaders: a package that provides functionality to read Excel files. Julia package version 0.9.0. Recuperado de https://github.com/queryverse/ExcelReaders.jl.

Avdeenko, T. V., & Mesentsev, Y. A. (2016). Efficient approaches to scheduling for unrelated parallel machines with release dates. IFAC-*PapersOnLine*, 49(12), 1743-1748.

Balachandran, B. V., & Zoltners, A. A. (1981). An interactive audit-staff scheduling decision support system. Accounting Review, 801-812.

Beaton, C., Diallo, C., & Gunn, E. (2016). Makespan minimization for parallel machine scheduling of semi-resumable and non-resumable jobs with multiple availability constraints. INFOR: Information Systems and Operational Research, 54(4), 305-316.

Bezanson, J., Karpinski, S., Shah, V. B., & Edelman, A. (2012). Julia: a fast dynamic language for technical computing. ArXiv preprint arXiv:1209.5145.

Blanca, M. J., Alarcón, R., Arnau, J., Bono, R., & Bendayan, R. (2018). Effect of variance ratio on ANOVA robustness: Might 1.5 be the limit?. Behavior research methods, 50(3), 937-962.

Blanca, M., Alarcón, R., Arnau, J., Bono, R., & Bendayan, R. (2017). Non-normal data: Is ANOVA still a valid option?. Psicothema, 29(4), 552-557.

Broek, van den, J. J. J. (2009). MIP-based approaches for complex planning problems. Eindhoven: Technische Universiteit Eindhoven DOI: 10.6100/IR653241.

Bülbül, K., Kedad-Sidhoum, S., & Şen, H. (2017). Single-machine common due date total earliness/tardiness scheduling with machine unavailability. Journal of Scheduling, 1-23.

Chan, K. H., & Dodin, B. (1986). A decision support system for audit-staff scheduling with precedence constraints and due dates. Accounting Review, 726-734.

Dodin, B., & Chan, K. H. (1991). Application of production scheduling methods to external and internal audit scheduling. European Journal of Operational Research, 52(3), 267-279.

Dodin, B., Elimam, A. A., & Rolland, E. (1998). Tabu search in audit scheduling. European Journal of Operational Research, 106(2-3), 373-392.

Fanjul Peyró, L. (2011). Nuevos algoritmos para el problema de secuenciación en máquinas paralelas no relacionadas y generalizaciones. Tesis Doctoral, Universitat Politècnica de València, España.

Fernández, A. D., Velarde, J. G., Laguna, M., Mascato, P., Tseng, F. T., Glover, F., & Ghaziri, H. M. (1996). Optimización heurística y redes neuronales. Ed. Paraninfo SA, Madrid, España.

Gedik, R., Rainwater, C., Nachtmann, H., & Pohl, E. A. (2016). Analysis of a parallel machine scheduling problem with sequence dependent setup times and job availability intervals. European Journal of Operational Research, 251(2), 640-650.

Harris, H., DuBois, C., White, J. M. & others (2015). Dataframes: tools for working with tabular data in Julia. Julia package version 0.6. Recuperado de https://github.com/JuliaData/DataFrames.jl.

Hashemian, N., Diallo, C., & Vizvári, B. (2014). Makespan minimization for parallel machines scheduling with multiple availability constraints. Annals of Operations Research, 213(1), 173-186.

Hecke, T. V. (2012). Power study of anova versus Kruskal-Wallis test. Journal of Statistics and Management Systems, 15(2-3), 241-247.

Kaabi, J., & Harrath, Y. (2014). A survey of parallel machine scheduling under availability constraints. International Journal of Computer and Information Technology, 3(2), 238-245.

Kurt, A. (2012). Unrelated parallel machines scheduling under machine availability and eligibility constraints. Thesis work, Master of Science in Industrial Engineering, Çankaya University, Turkey.

Lee, C. Y. (1996). Machine scheduling with an availability constraint. Journal of Global Optimization, 9, 363–382.

Lin, Y. K. (2013). Particle swarm optimization algorithm for unrelated parallel machine scheduling with release dates. Mathematical Problems in Engineering, volume 2013.

Low, C., Li, R. K., & Wu, G. H. (2016). Minimizing total earliness and tardiness for common due date single-machine scheduling with an unavailability interval. Mathematical Problems in Engineering, 2016.

Ma, Y., Chu, C., & Zuo, C. (2010). A survey of scheduling with deterministic machine availability constraints. Computers & Industrial Engineering, 58(2), 199-211.

Martí, R., Pardalos, P. M., & Resende, M. G. (Eds.). (2018). Handbook of Heuristics. Springer.

Mohamed, A. M. (2015). Operations Research Applications in Audit Planning and Scheduling. International Journal of Social, Behavioral, Educational, Economic, Business and Industrial Engineering, 9(6), 2026-2034.

Montgomery, D. C. (2017). Design and analysis of experiments. John Wiley & Sons.

Noronha, F. (2018). XLSX: Excel file reader/writer coded in pure Julia. Recuperado de https://github.com/felipenoris/XLSX.jl.

Pinedo, M. (2016). Scheduling: Theory, Algorithms, and Systems (5th ed.), Springer, 674p.

Potts, C. N., & Strusevich, V. A. (2009). Fifty years of scheduling: a survey of milestones. Journal of the Operational Research Society, 60(sup1), S41-S68.

Romero Villafranca, R., & Zúnica Ramajo, L. R. (2013). Métodos Estadísticos para Ingenieros. Colección Académica. Editorial UPV.

Saaty, T. L. (1977). A scaling method for priorities in hierarchical structures. Journal of Mathematical Psychology, 15, 59-62.

Saaty, T. L. (1980). The Analytic Hierarchy Process. McGraw-Hill International Book Company, New York.

Sevindik, K. (2006). Parallel machine scheduling subject to machine availability constraints. Thesis work, Master of Science, Institute of Engineering and Science, Bilkent University, Turkey.

Sousa, J. P., & Wolsey, L. A. (1992). A time indexed formulation of non-preemptive single machine scheduling problems. Mathematical programming, 54(1-3), 353-367.

Summers, E. L. (1972). The audit staff assignment problem: a linear programming analysis. The Accounting Review, 47(3), 443-453.

The Institute of Internal Auditors (2017). Normas internacionales para el ejercicio profesional de la auditoría interna. Recuperado de https://na.theiia.org/translations/PublicDocuments/IPPF-Standards-2017-Spanish.pdf.

Unlu, Y. & Mason, S. J. (2010). Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems. Computers & Industrial Engineering, 58(4), 785-800.

Vallada, E., & Ruiz, R. (2011). A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. European Journal of Operational Research, 211(3), 612-622.

Van den Akker, J. M., Hurkens, C. A., & Savelsbergh, M. W. (2000). Time-indexed formulations for machine scheduling problems: Column generation. INFORMS Journal on Computing, 12(2), 111-124.

Wang, G., Sun, H., & Chu, C. (2005). Preemptive scheduling with availability constraints to minimize total weighted completion times. Annals of Operations Research, 133(1-4), 183-192.

Anexos

1. ***"05Aud_70P_1.xlsm"*** – Instancia (primera) del problema con 5 auditores y 70 proyectos.
2. ***"05Aud 70P Rel Unav 1 (2019-06-08) -- 30min.xlsm"*** – Ejemplo de fichero de visualización, por medio de un gráfico de Gantt, de los resultados de una solución de secuenciación de proyectos de auditoría (obtenida por el modelo MIP con un *timeout* de 30 minutos para el CPLEX).
3. ***"To-Sel-Proj Rel Unav.mod"*** – Código OPL para la resolución exacta (modelo MIP) del problema con las restricciones de *release times* e indisponibilidades.
4. ***"To-Sel-Proj 05Aud 70P Rel Unav.dat"*** – Fichero para importación de los datos del modelo MIP para una instancia de 5 auditores y 70 proyectos, con las restricciones de *release times* e indisponibilidades.
5. ***"1st Heuristic 05Aud 70P Rel Unav.jl"*** – Código Julia para el primer heurístico ("1st Heur") aplicado a una instancia de 5 auditores y 70 proyectos, con las restricciones de *release times* e indisponibilidades.
6. ***"2nd Heuristic 05Aud 70P Rel Unav (2).jl"*** – Código Julia para el segundo heurístico ("2nd Heur") aplicado a una instancia de 5 auditores y 70 proyectos, con las restricciones de *release times* e indisponibilidades.
7. ***"2nd Heuristic 05Aud 70P Rel Unav (2.1).jl"*** – Código Julia para el segundo heurístico modificado, con el orden natural de los auditores ("2nd Mod"), aplicado a una instancia de 5 auditores y 70 proyectos, con las restricciones de *release times* e indisponibilidades.
8. ***"2nd Heuristic 05Aud 70P Rel Unav (2.2).jl"*** – Código Julia para el segundo heurístico modificado, con el orden de los peores auditores ("2nd Mod W"), aplicado a una instancia de 5 auditores y 70 proyectos, con las restricciones de *release times* e indisponibilidades.
9. ***"2nd Heuristic 05Aud 70P Rel Unav (2.3).jl"*** – Código Julia para el segundo heurístico modificado, con el orden aleatorio de los auditores ("2nd Mod R"), aplicado a una instancia de 5 auditores y 70 proyectos, con las restricciones de *release times* e indisponibilidades.
10. ***"Anova Heuristicos 4.SGP"*** – Fichero *Statgraphics* de los resultados del ANOVA para la comparación del rendimiento de los heurísticos.
11. ***"Datos RPD Heuristicos 4.sgd"*** – Fichero *Statgraphics* de los datos de rendimiento de los heurísticos.