Proyecto I: Calendario de Ofertas Personalizadas

Grado en Ciencia de Datos (Optimización)

Universitat Politécnica de Valencia

Álvaro Faubel Sanchis, Ignacio Cano Navarro, Daniel Goig Martínez, Angel Langdon Villamayor

***Abstract* - La rápida evolución de las tecnologías de la información durante los últimos años nos ha permitido obtener una enorme cantidad de información de los usuarios que navegan por Internet, la cual puede ser usada para perfilar a los clientes que consumen ciertos productos. Esto es especialmente útil para los sistemas de recomendación, que se basan en dicha información para enviar ofertas personalizadas a los mismos. En este trabajo académico se propone un sistema recomendador capaz de enviar cinco ofertas personalizadas por día a los usuarios de una empresa de e-commerce, maximizando la satisfacción del usuario y los beneficios de la empresa mediante un modelo de optimización mixta entera lineal (MILP). También se propone un modelo de KNN capaz de estimar la satisfacción de un usuario al recibir ofertas de ciertos productos basándose en la información previa de usuarios similares. Por último se ha comprobado la rapidez y calidad de los resultados utilizando diferentes *'solvers'* capaces de resolver el problema presente. Se ha concluido que 'SCIP' es el *solver* mejor funciona en rasgos generales en cuanto a estabilidad y rendimiento.**

I. Introducción

Los sistemas de recomendación son herramientas importantes que ayudan al usuario a conocer opciones o elementos de interés que permiten personalizar su experiencia. Tenemos contacto con estos poderosos sistemas de recomendación a diario. Cuando disfrutamos de un video en YouTube o dejamos que Spotify haga una mezcla de artistas para una playlist, estamos aportando elementos de personalización para que estos sistemas construyan sus recomendaciones en base a nuestros datos.

Nuestra empresa se encarga de intermediar entre proveedores de servicios (Shows, Restaurantes y Viajes) y los clientes, llevándose una tarifa por cada servicio vendido a través de su página web.

El objetivo encargado es construir un sistema recomendador, que a partir de datos de la empresa, sea capaz de ofertar cinco productos durante cada día del mes de Julio a un usuario en concreto. Esto permitirá a la empresa conseguir unos clientes que confien y generen 'engagement' en su página web, así como aumentar las posibilidades de que dichos clientes terminen comprando a través de esta.

Para cumplir dicho objetivo, se ha propuesto optimización mixta entera lineal en el que se tiene en cuenta en la función de maximización factores como la satisfacción del usuario con un producto, el beneficio de la empresa si dicho usuario comprara dicho producto y si el producto tiene descuento. También se tendrán en cuenta una serie de restricciones, detalladas más adelante, con el objetivo de modelizar el problema lo más cercano a la realidad posible.

Para la resolución del modelo, se utilizarán tres 'solvers' capaces de resolver problemas MILP y se compararán y analizarán sus resultados. Este trabajo académico se dividirá en los apartados, 'Trabajo relacionado', 'Modelo matemático', 'Preproceso de los datos', 'Implementación', 'Experimentos', y 'Conclusiones'.

II. Trabajo Relacionado

A. Sistema de recomendación por optimización de enjambre de partículas [1]

Este artículo describe un nuevo sistema de recomendación, que emplea un algoritmo de optimización de enjambre de partículas (PSO) para aprender las preferencias personales de los usuarios y proporcionar sugerencias a medida. Esto se hace construyendo perfiles de usuarios y utilizando un algoritmo para encontrar perfiles similares al usuario actual. Se llevan a cabo experimentos para observar el rendimiento del sistema y los resultados se comparan con los obtenidos por el algoritmo genético (CA) y un sistema estándar no adaptativo basado en el algoritmo de Pearson.

Se realizaron dos versiones del sistema. Una llamada tolerancia cero en la que la precisión del sistema se determina calculando el porcentaje del número de valoraciones que el sistema predijo correctamente sobre el número total de valoraciones disponibles por el usuario activo actual. La otra, llamada Tolerancia At-Most-One, es igual que la tolerancia cero pero si la diferencia entre las valoraciones predichas y las reales es menor, sí se considera que la clasificación prevista es correcta. El PSO obtuvo resultados significativos en un menor tiempo que los otros dos algoritmos. Además, se obtiene una mejora significativa si se utiliza At-Most-One.

Este trabajo es muy similar al nuestro ya que busca recomendar productos a usuarios en base a sus preferencias. Entre las diferencias que tiene respecto a nuestro proyecto

están el algoritmo utilizado para resolver el problema y las características utilizadas para llevarlo a cabo. Mientras que en este paper utilizan 22 variables como la edad, el género o la ocupación del usuario, nosotros empleamos el descuento, el predicted score o el beneficio que se lleva la empresa por producto/servicio vendido.

B. Optimización de objetivos múltiples en los sistemas de recomendación. [2]

En este trabajo académico, se busca mediante el uso de una función optimización multiobjetivo la maximización de los mejores matches (candidatos) para una oferta de trabajo publicada en Linkedin y el 'Job Seeking Intent', es decir, la probabilidad de que acepte una oferta. Así pues, la función de MOO tendrá que buscar los top K candidatos ideales para un puesto de trabajo maximizando estas dos características ya que un potencial candidato puede tener un match perfecto para un puesto de trabajo, pero su 'Job Seeking Intent' ser bajo y probablemente nunca aceptaría dicha oferta. Este modelo se puso en prueba mediante A/B 'testing'. Las similitudes con el objetivo de este trabajo son muy grandes, siendo la diferencia más reseñable el hecho de que estos autores están resolviendo un problema multiobjetivo, mientras que en este se ha optado por simplificar el problema mediante 'pesos' para siempre tratar de resolver un problema de optimización con un único objetivo.

C. Un sistema de recomendación basado en un algoritmo de optimización de malezas invasoras [3]

En este trabajo académico, se busca optimizar un sistema recomendador de artículos a usuarios utilizando un 'nuevo' algoritmo, el 'Invasive Weed Optimization Algorithm' (IWO). En este caso, se utiliza el IWO para optimizar los 'priority levels', es decir, la prioridad que se le da a cada una de las características de los individuos de la base de datos sobre las que se predecirá posteriormente el match. Esta táctica de dar prioridad a las características de los usuarios es conocida como 'prioritized user-profile'. Para demostrar la calidad de los resultados se comparan sobre el mismo problema los resultados obtenidos con los siguientes algoritmos: 'Pearson, Particle Swarm (PSO) and Genetic Algorithm'. La métrica utilizada para la evaluación es el RMSE y de los cuatro algoritmos, el IWO es el que consigue el menor valor. Cabe destacar que aunque los autores afirman que IWO ha conseguido los mejores resultados, no se realiza ningún test para demostrar que estadísticamente los resultados son mejores que los del resto de algoritmos mostrados.

D. *Sistema de recomendación basado en el filtrado colaborativo para los usuarios de Spotify* [4]

Este trabajo propone una metodología basada en el filtering colaborativo para recomendar música a los usuarios de Spotify a partir de una lista ordenada de las canciones más reproducidas durante un periodo de tiempo. Proporciona predicciones de canciones basadas en las opiniones de otros usuarios similares con el algoritmo de los vecinos más cercanos (KNN). La similitud entre usuarios se calcula mediante coeficientes de correlación como el coeficiente de similitud del coseno, el coeficiente de correlación de Pearson, el coeficiente de correlación de Spear-man, o el coeficiente de Phi.

Para comprobar los resultados se han entrenado dos modelos de KNN, uno con el coeficiente de Pearson y otro con el coeficiente de Phi. Los resultados de ambos se han comparado utilizando las métricas de mean average precisión (MAP), F1-score y normalized discounted cumulative gain (NDCG). Los resultados de ambos modelos son malos, obteniendo un MAP de 0.1, lo que significa que solo una de cada diez canciones recomendadas será correcta. El f1-score y el NDCG son igual de malos para los dos modelos. Esto indica que para un subconjunto ordenado de artículos con la máxima calificación, si la dimensión del subconjunto es mucho menor que la dimensión del conjunto, el orden apenas aporta información. Por eso ambos modelos proporcionan los mismos resultados.

El planteamiento de este trabajo nos permite estudiar otras aproximaciones a sistemas recomendadores que no utilizan problemas de optimización lineal.

III. Preproceso de los datos

Este apartado tiene como propósito introducir los datos facilitados por la empresa, así como el proceso realizado a los mismos para su futura consideración en el modelado.

Dado que el propósito del trabajo es realizar un recomendador de productos a nivel usuario, el preproceso se ha llevado a cabo bajo el mismo nivel. En rasgos generales el preproceso ha consistido en el filtrado de los datos más importantes relativos a un usuario bajo diversas consideraciones. Buscamos con esto quedarnos solo con la información útil de cada base de datos relativa a un cliente, que después será empleada en la creación del modelo.

Inicialmente disponemos de 5 bases de datos en *.csv*:

- Productos: Contiene la información de todos los productos que ofrece el e-commerce (3803 en total).

- Usuarios: Contiene la información pertinente a los usuarios del e-commerce (1000 usuarios en total).

- Preferencias: Contiene la valoración del 0-5 de diversos usuarios en relación con algunos de los productos ofrecidos. (573894 valoraciones en total).

- Compras: Contiene información relativa a las compras de productos realizadas por algunos de los usuarios. (5117 registros de compras en total).

- Descuentos: Contiene el dato de los descuentos de algunos productos en un día del mes concreto.

En cuanto a los productos que se le pueden mandar a un usuario, de entre todos los disponibles hemos decidido en primer lugar no considerar aquellos que estén de fuera de su ciudad, pues creemos que no le serán interesantes. A su vez, se han descartado todos aquellos productos relacionados con espectáculos (tipo SHW) que no estén disponibles a fecha de Julio de 2021, pues es obvio que no tiene sentido recomendar un espectáculo al cual ya no se puede asistir. Además, en caso de que el usuario haya comprado anteriormente productos vacacionales o de espectáculos (tipos VAC y SHW), estos tampoco serán incluidos. Por lo general a una persona le gusta recibir recomendaciones de productos diferentes a los que ya ha comprado, y en el caso de los viajes y los espectáculos es raro que se quiera repetir. No obstante, no hemos considerado eliminar los restaurantes (RES) ya visitados por ser más común que un usuario quiera volver a comer en un sitio de su agrado o del cual tiene una buena valoración.

Como hemos mencionado anteriormente, uno de los aspectos que necesitamos maximizar es la satisfacción del usuario, la cual se encuentra disponible en las columnas *predicted_score* y *real_score* del dataset de preferencias. No obstante, dado que no disponemos de las valoraciones de todos los usuarios para todos los productos, necesitamos de alguna forma estimar los *scores* faltantes asociados al subconjunto de productos del usuario a tratar.
Como solución a ello decidimos aplicar el método de los 5-Vecinos-Más-Cercanos (KNN con k = 5) sobre todos los productos filtrados del usuario que no posean de valoración con respecto a todos aquellos usuarios de su misma ciudad presentes en la base de datos de preferencias y que por tanto tienen asociados scores para diversos productos. Así pues, cada valoración de producto faltante será imputada con el *score* medio de sus 5 vecinos más cercanos.
Con la intención de obtener una mejor estimación de los vecinos más cercanos, hemos considerado la edad y el trabajo del usuario, así como el precio, el tipo y el subtipo del producto como columnas de entrada al modelo de KNN.

En cuanto a los descuentos, con la intención de tener una mejor estructura de datos de cara a la creación del modelo (en concreto de la función objetivo), para todos los productos filtrados del usuario se han rellenado a 0 todos aquellos días en los que no existe descuento.

Por último, se ha realizado un escalado *Min-Max* de las columnas con el *fee*, el *score* y el descuento de los productos filtrados del usuario para poder incluirlas con el mismo orden de magnitud dentro de la función de maximización del modelo.

IV. Modelo matemático de optimización propuesto

En este apartado se describe el modelo matemático de optimización utilizado para resolver el problema. Explicaremos las variables de decisión, las constantes, los parámetros, las restricciones y la función objetivo.

En cuanto a la definición de **variables**, hemos creado variables (booleanas) que indican si un producto *i* se manda en un día *j*. [1] Se definen así:

X_{ij} : Se envía el producto i el día j; $1 \leq i \leq 3803$; $1 \leq j \leq 31$.
Variable binaria; $X_{ij} = \{0,1\}$.

En cuanto a las **constantes**, hemos utilizado 3: el *product fee,* el *descuento* y el *predicted score*.

El *product fee* se define como:
F_i: Es la comisión que recibirá el e-commerce por cada persona que compre la comida/evento/viaje i; $1 \leq i \leq 3803$; $F_i \geq 0$; $F_i \in R$

El *descuento* se define como:
D_{ij}: Porcentaje de descuento aplicable sobre el precio base por persona para el producto i en el dia j; $1 \leq i \leq 3803$; $1 \leq j \leq 31$; $D_{ij} \geq 0$; $F_i \in R$

El *predicted score* se define como:
S_{iu}: Predicción de cuánto le gustará el producto i al usuario u, en una escala de 0 a 5 estrellas; $1 \leq i \leq 3803$; $1 \leq u \leq 1000$; $S_{iu} \geq 0$; $S_{iu} \in R$

La **función objetivo** pretende maximizar la satisfacción del cliente tras recibir los productos y/o servicios de la e-commerce a la vez que los beneficios de la empresa.

Hemos considerado que la satisfacción del cliente viene dada por el *predicted score* y el posible *descuento* sobre el producto en el día que se envía. Por otro lado, los beneficios de la empresa vienen dados por el *product fee*.

Para maximizar el valor de la función objetivo hemos tenido en cuenta, por tanto, una función lineal que relaciona la comisión que recibirá el e-commerce por cada persona que participe en una comida/evento/viaje (*product fee*), la predicción de cuánto le gustará los productos enviados al usuario en concreto (*predicted score*) y el *descuento* diario de los productos que se le envíen al usuario. Consideramos además que el *predicted score* es el parámetro más importante seguido del *product fee* y el *descuento.* Por eso, les hemos asignado unos pesos de 0.5, 0.3 y 0.2 respectivamente Surge así una función lineal de la forma:

$$SB = (0.5 \times S_{iu} + 0.3 \times F_i + 0.2 \times D_{ij}),$$

la cual se activará en la función objetivo cuando se decida enviar el producto *i* al usuario *u*.

[1] Siempre y cuando el producto *i* de tipo *SHW* no haya caducado ese día *j*. *Además,* se consideran solamente aquellos productos *i* que puedan ser enviados al usuario (no se han eliminado en el preproceso mediante filtrados).

Cabe tener en cuenta que los parámetros *predicted score, product fee* y *discount* han sido escalados en el preproceso para que tengan el mismo orden de magnitud. A su vez, el *discount* puede valer cero cuando para un determinado día el producto no presente descuento.

Así pues se define la función de maximización para el usuario *u* como:

$$max\ Z = \sum_{i=1}^{N} \sum_{j=1}^{31} (0.5\, S_{iu} + 0.3\, F_i + 0.2\, D_{ij}) * X_{ij};$$

N: Número de productos que se pueden enviar al usuario (los que no se han eliminado después del preprocesado).

En cuanto a las **restricciones,** consideramos que el modelo ha de estar sujeto a:

- *Se deben mandar exactamente 5 productos cada día.*

[productos por dia]: $\sum_{i=1}^{N} X_{ij} = 5;\ 1 \le j \le 31$; siendo N el número de productos que se pueden enviar al usuario.

- *Un mismo producto se puede enviar como mucho 5 veces en todo el mes.*

[apariciones max producto] $\sum_{j=1}^{31} X_{ij} \le 5\ ;\ 1 \le i \le N;$

siendo N el número de productos que se pueden enviar al usuario.

- *Para que un producto pueda volverse a mandar tienen que pasar al menos 3 días.*

$X_{ij} + X_{ij+1} + X_{ij+2} \le 1\ ;\ 1 \le i \le N\ ;\ 1 \le j \le 29;$

siendo N el número de productos que se pueden enviar al usuario.

- *La cantidad de productos enviados de un tipo no puede superar el 40% de la suma total de los productos enviados.*

Definimos en primer lugar las variables auxiliares:

$$RES = \sum_{X_{ij} \in R} X_{ij}\ ;\ R = \{X_{ij} \mid tipo(i) = 'RES'\};$$

Conjunto de productos de tipo RES que se envían al usuario.

$$VAC = \sum_{X_{ij} \in R} X_{ij}\ ;\ R = \{X_{ij} \mid tipo(i) = 'VAC'\};$$

Conjunto de productos de tipo VAC que se envían al usuario.

$$SHW = \sum_{X_{ij} \in R} X_{ij}\ ;\ R = \{X_{ij} \mid tipo(i) = 'SHW'\};$$

Conjunto de productos de tipo SHW que se envían al usuario.

[tipo RES]: $RES \le 0.4(RES + VAC + SHW)$
[tipo VAC]: $VAC \le 0.4(RES + VAC + SHW)$
[tipo SHW]: $SHW \le 0.4(RES + VAC + SHW)$

- *La cantidad de productos enviados de un proveedor no puede superar un cierto porcentaje de la suma total de los productos enviados por los proveedores del mismo tipo de producto.*

Definimos en primer lugar las variables auxiliares:

$$PROV_p = \sum_{X_{ij} \in R} X_{ij}; R = \{X_{ij} \mid prov(i) = "p"\};\ 1 \le p \le 5;$$

conjunto de productos del proveedor *p* que se envían al usuario.[2]

[Prov 1]: $PROV_1 \le 0.4\ (PROV_1 + PROV_2 + PROV_3)$
[Prov 2]: $PROV_2 \le 0.4\ (PROV_1 + PROV_2 + PROV_3)$
[Prov 3]: $PROV_3 \le 0.4\ (PROV_1 + PROV_2 + PROV_3)$
[Prov 4]: $PROV_4 \le 0.6\ (PROV_4 + PROV_5)$
[Prov 5]: $PROV_5 \le 0.6\ (PROV_4 + PROV_5)$

- *Como mucho se pueden enviar 2 productos de un mismo tipo en un dia*

[max tipo día]: $\sum_{j=1}^{31} X_{ij} \mid tipo(i) = 'c' <= 2\ ;$

$c = \{RES, SHW, VAC\};\ 1 \le j \le 31$

- *Como mucho se pueden enviar 1 producto de un mismo subtipo en un dia*

[max subtipo día]: $\sum_{j=1}^{31} X_{ij} \mid subtipo(i) = 's' <= 1\ ;$

$s = \{Local, Tapas, Italian, Burguer, Creative, Vegetarian, Asian, Music\ Event, Theatre\ Event, Exhibition\};$
$1 \le j \le 31$

- *Como mucho se pueden enviar 5 restaurantes a los cuales ya haya ido el usuario*

Definimos la variable auxiliar necesaria:

$$RES_{purch} =$$

$$\sum_{X_{ij} \in R} X_{ij}; R = \{X_{ij} \mid tipo(i) = 'RES' \wedge comp(i) = True\}$$

[max prods RES ido]: $\sum_{j=1}^{31} X_{ij} \le 5 \mid i \in RES_{purch}\ ;$

[2] No se crea la variable auxiliar PROV6 porque dicho proveedor sólo ofrece productos de tipo *VAC*.

V. Implementación

La implementación del modelo, dada la gran cantidad de variables y restricciones, se ha realizado haciendo uso de un entorno informatizado. Para ello, se ha hecho uso de *Python 3* y en especial, la librería *pywraplp (ortools)* a la hora de plantear y resolver el modelo.

Así pues se han empleado las librerías *requests* para descargar los datos, *numpy* y *pandas* para el filtrado, manipulación y limpieza de los datos, y *sklearn* para la imputación y el escalado. Concretamente se han empleado los métodos *sklearn.KNNImputer* para la imputación del *score* de los productos faltantes dado un usuario, y *sklearn.MinMaxScaler* para el escalado del *fee*, el *score* y el descuento de los productos del usuario.
Además, en la experimentación, se emplea *random* para la selección aleatoria de los usuarios y las tallas del problema, y *matplotlib* para graficar los resultados obtenidos, así como *scipy* y *statsmodels* para la comprobación estadísticas de los resultados.

Todo el código ha sido desarrollado en un archivo de *Jupyter Notebook* (.ipynb) [5], el cuál está estructurado en secciones y subapartados de la siguiente forma:

TABLA I. Estructura del código desarrollado

Sección	***Subapartado***	***Descripcción***
Preprocesado	-	Funciones para el preprocesado del usuario.
Modelado	Variables	Funciones para la creación de las variables del modelo.
	Función objetivo	Función para la creación de la función objetivo del modelo.
	Restricciones	Funciones para la creación de las restricciones del modelo.
	Solución	Función para la resolución del problema con el modelo creado.
	Ejemplo de uso	Uso de todas las funciones anteriores para el modelado y resolución del problema para un usuario ejemplo.
Experimentación	Extracción resultados	Creación del bucle de experimentos y almacenaje de los resultados.
	Visualización resultados	Gráficas de los resultados experimentales obtenidos.
	ANOVA	Prueba del ANOVA sobre los resultados. Tests de normalidad y homocedasticidad.

VI. Experimentación

El objetivo de este apartado es evaluar el rendimiento de diferentes solvers ante varias tallas del problema planteado y comprobar si existen diferencias significativas entre las soluciones halladas por los *solvers* y el tiempo de ejecución de los mismos con diferentes tallas. Así pues se ha definido la talla del problema como la cantidad de productos disponibles que se le pueden mandar al usuario.

Cabe destacar que la experimentación ha sido realizada con una CPU de 12 Cores a 4,3 GHz. Dado que no todos los *solvers* de OR Tools soportan la ejecución multihilo, el número de hilos fijado para todos los experimentos ha sido de 1. Además se ha elegido un límite en el tiempo de ejecución máximo de 1 minuto para todos los solvers.

Así pues se han probado los *solvers* 'SCIP' [6], 'CBC' [7] y 'SAT' [8] (todos ellos destinados a la resolución de problemas MILP) con distintas tallas de problema y para múltiples usuarios. En concreto se ha elegido una muestra de diez usuarios aleatorios de la base de datos y tallas de tamaños: 100, 200, 300, 400, 500, 800 y 1000 productos aleatorios para cada uno de ellos. [3]

Para cada experimento se guarda el tipo de solución obtenida (óptima o factible), el valor de la función objetivo y el tiempo que ha tardado el *solver*.

Los resultados temporales obtenidos se presentan en la siguiente figura:

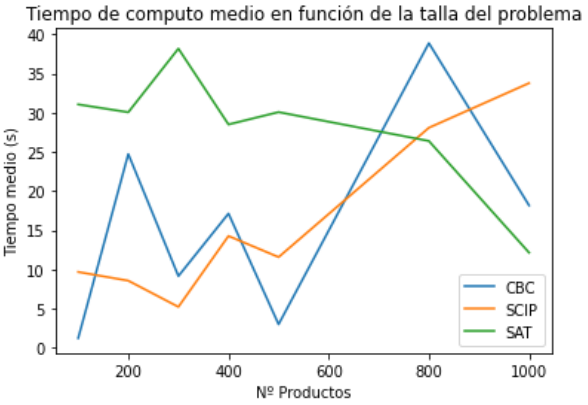

Fig. 1. Tiempo de cómputo medio solvers vs talla del problema

En la gráfica puede observarse el tiempo medio de resolución del problema de cada *solver* para los diez usuarios y para cada talla fijada. A simple vista se observan diferencias entre los tiempos de ejecución de los diferentes *solvers*, siendo el 'SAT' el que mejor funciona para tallas grandes (>800) y 'SCIP' para tallas pequeñas.
Observamos además que el único *solver* con un comportamiento esperado/estable (con tiempos más elevados a medida que aumenta la talla) es el 'SCIP'. No obstante cabe destacar que se están considerando todos los tiempos independientemente del tipo de solución obtenida. Esto provoca picos de tiempo medio pronunciados en algunas tallas para las cuales el *solver* no ha conseguido llegar a la solución óptima en el tiempo máximo fijado. Podemos constatar este hecho calculando los porcentajes de soluciones óptimas obtenidos para cada talla y *solver*

[3] Nótese que para algunos de los usuarios puede no haber tantos productos disponibles como para realizar los experimentos de todas las tallas. En este caso la experimentación de dicho usuario se termina en la talla máxima posible.

TABLA II. PORCENTAJE DE SOLUCIONES ÓPTIMAS POR SOLVER Y TALLA

Tipo de Solver	*% Sol óptimas*						
	100	*200*	*300*	*400*	*500*	*800*	*1000*
SAT	50	60	37.5	75	62.5	75	87.5
SCIP	90	90	100	87.5	100	87.5	87.5
CBC	100	60	87.5	75	100	37.5	75

TABLA III. PORCENTAJE DE SOLUCIONES ÓPTIMAS TOTALES POR SOLVER

Tipo de Solver	*% Sol óptimas*
SAT	63.33
SCIP	91.67
CBC	76.67

Como podemos observar en las tablas anteriores, el *solver* que mayor tasa de resultados óptimos proporciona es el 'SCIP'. De ahí a su mejor comportamiento en la *Fig.1* comentada.

Es de interés ahora ver las diferencias en cuanto al valor de la función objetivo para los distintos tipos de *solver* y tallas.

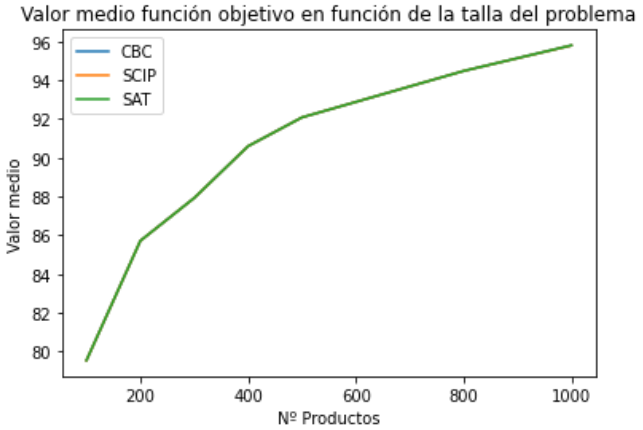

Fig. 2. Valor medio f. objetivo solvers vs talla del problema

Vemos en la *Fig. 2* como el valor de la función objetivo es prácticamente igual para todos los *solvers*. Esto implica a su vez la obtención de soluciones factibles muy cercanas a las óptimas. Además comprobamos que a medida que aumenta la talla el valor objetivo es mayor, lo que supone que cuantos más productos tenga el recomendador entre los que elegir para un usuario, mejor funcionará.

Con el objetivo de estudiar de forma más realista las posibles diferencias entre el *solver* y la talla del problema con respecto al tiempo de ejecución, hemos de analizar únicamente los resultados asociados a soluciones óptimas.

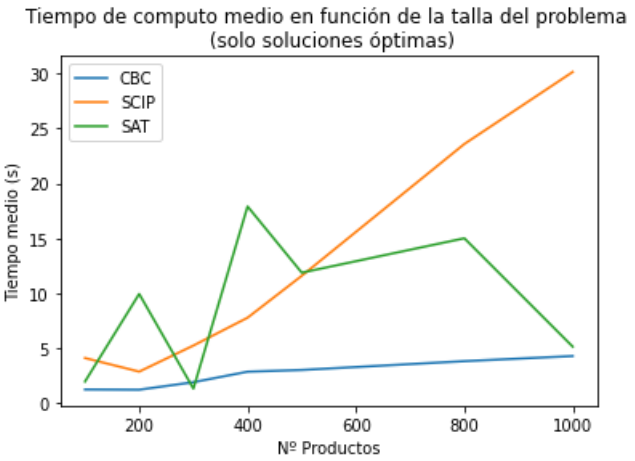

Fig. 3. Tiempo de cómputo medio solvers vs talla del problema (sol. óptimas)

Denotamos ahora un cambio significativo en los comportamientos de los diferentes *solvers*, siendo 'CBC' el que mejor funciona cuando se encuentran soluciones óptimas. No obstante cabe recordar que la tasa de soluciones factibles pero no óptimas para 'CBC' es de 23.33%, lo que afecta a sus tiempos medios globales. Es por ello que 'SCIP' es un mejor candidato en cuanto a estabilidad se refiere.

Observamos además que tanto 'SCIP' como 'CBC' aumentan sus tiempos de ejecución conforme a la talla, mientras que 'SAT' no parece seguir una clara tendencia.

Con el fin de garantizar que realmente existen diferencias significativas entre los tiempos de ejecución medios de las soluciones óptimas para los diferentes *solvers* y tallas, se ha utilizado un ANOVA [9] de dos factores.

Para poder realizar el análisis de la varianza se ha necesitado comprobar los principios de normalidad y homocedasticidad para las distintas poblaciones, en este caso los tiempos medios.

Dado que la cantidad de muestras es menor a 50, se ha utilizado el test de *Shapiro* para comprobar la normalidad de los datos.

TABLA IV. RESULTADOS TEST DE NORMALIDAD SAPHIRO

Tipo de Solver	*p value*	*Normalidad*
SAT	0.61 > 0.05	SI
SCIP	0.1 > 0.05	SI
CBC	0.48 > 0.05	SI

Como vemos en la TABLA III, se ha aceptado la hipótesis de normalidad para las tres poblaciones (p-value > 0.05).

En cuanto a la comprobación de la homocedasticidad, se ha llevado a cabo el test de *Levene*:

TABLA V. RESULTADOS TEST HOMOCEDASTICIDAD LEVENE

Prueba	*p value*	*Homocedasticidad*
Homocedasticidad	0.076 > 0.05	SI

Dado que el test de *Levene* resulta positivo (p-value > 0.05) confirmamos que no hay diferencias entre las varianzas de los diferentes tiempos y por tanto se cumple la homocedasticidad.

Se procede entonces al cálculo del ANOVA:

TABLA VI. RESULTADOS ANOVA PARA LOS TIEMPOS

Variable	*sum_sq*	*df*	*F*	*PR(>F)*
Solver	332	2	9.6	0.002
Size	334.6	1	19.89	0.0004
Solver-Size Interaction	325.3	2	9.4	0.0022

Como se puede ver en la *TABLA V,* ambos factores resultan significativos (PR(>F) < 0.05). Esto indica que tanto la talla como el *solver* afectan de forma significativa a los tiempos medios de ejecución cuando las soluciones son óptimas. Además, existe interacción entre ambos factores, lo que supone que el efecto de uno de ellos sobre la variable dependiente no es el mismo en todos los niveles del otro. Esto se podía evidenciar ya en la *Fig. 1I,* donde veíamos que el *solver* 'SAT' no seguía la misma tendencia que los otros para los distintos tipos de tallas.

VII. CONCLUSIONES

En este artículo académico se ha descrito el proceso de creación de un sistema de recomendación, el preproceso consistente en la imputación por KNN de valores faltantes y el problema de optimización mixta entera lineal a resolver. En este además se han descrito las restricciones que ajustan el modelo a la realidad, así como una función de maximización que considera tanto la satisfacción del usuario como los beneficios de la empresa.

En cuanto a los resultados de la experimentación asociada a varios *solvers* y tallas del problema, hemos visto que 'SCIP' es el solver que mejor funciona dada su estabilidad. A pesar de que 'CBC' ofrece resultados más rápidos cuando se encuentran la solución óptima, 'SCIP' consigue un mayor número total de soluciones óptimas, lo que provoca que los tiempos globales (considerando también las soluciones factibles) sean menores para dicho *solver.*

Respecto al modelo planteado, una propuesta de mejora para el futuro sería la creación de un modelo de *Reinforcement Learning* capaz de aprender los pesos asignados en el modelo de optimización lineal. Esto creemos que mejoraría el resultado general del modelo ya que recordamos que en este trabajo se han asignado manualmente basándonos en la experiencia.

Además, hubiera sido interesante extender los experimentos a toda la población de usuarios, cosa que por motivos de coste de computación y temporal no se ha podido llevar a cabo en este trabajo.

REFERENCIAS

[1] Ujjin, S., & Bentley, P. J. (2003, April). Particle swarm optimization recommender system. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No. 03EX706)* (pp. 124-131). IEEE.

[2] Rodriguez, M., Posse, C., & Zhang, E. (2012, September). Multiple objective optimization in recommender systems. In *Proceedings of the sixth ACM conference on Recommender systems* (pp. 11-18).

[3] Rad, H. S., & Lucas, C. (2007, September). A recommender system based on invasive weed optimization algorithm. In *2007 IEEE Congress on Evolutionary Computation* (pp. 4297-4304). IEEE.

[4] Pérez-Marcos, J., & Batista, V. L. (2017, June). Recommender system based on collaborative filtering for Spotify's users. In *International Conference on Practical Applications of Agents and Multi-Agent Systems* (pp. 214-220). Springer, Cham.

[5] Faubel, Á., Cano, I., Langdon, A., & Goig, D. (2021, 25 noviembre). Proyecto I: Calendario de Ofertas Personalizadas. Recursos y código.

[6] Achterberg, T. (2009). SCIP: solving constraint integer programs. *Mathematical Programming Computation*, *1*(1), 1-41.

[7] Forrest, J., & Lougee-Heimer, R. (2005). CBC user guide. In Emerging theory, methods, and applications (pp. 257-277). INFORMS.

[8] Eén, N., & Sörensson, N. (2003, May). An extensible SAT-solver. In International conference on theory and applications of satisfiability testing (pp. 502-518). Springer, Berlin, Heidelberg.

[9] Terrádez, M., & Juan, A. A. (2003). Análisis de la varianza (ANOVA). línea]. Disponible en: http://www. uoc. edu/in3/emath/docs/ANOVA. pdf.