Metaheurísticas para calibración de modelos basados en agentes en dinámicas de adopción premium

Ignacio Moya*, Manuel Chica*,William Rand†, Oscar Cordón*
*Instituto Andaluz Interuniversitario DaSCI (Data Science and Computational Intelligence), Universidad de Granada, España
† Poole College of Management, North Carolina State University, NC, United States
Emails: imoya@ugr.es, manuelchica@ugr.es, wmrand@ncsu.edu, ocordon@decsai.ugr.es

***Resumen*—Las aplicaciones *freemium* están creando nuevos escenarios de marketing, incentivando la adopción de servicios mediante la interacción entre usuarios. Para entender las dinámicas de estas aplicaciones son útiles los modelos basados en agentes, pero deben calibrarse con datos reales para poder ajustar su comportamiento a la realidad. Las metaheurísticas son métodos de optimización frecuentemente usados para calibración de modelos, dado que pueden ajustar los parámetros del modelo. En este artículo comparamos distintas metaheurísticas para calibrar un modelo basado en agentes que replica las dinámicas de adopción de contenido *premium* usando el modelo de Bass. Aplicamos estas metaheurísticas a cuatro datasets y llevamos a cabo un análisis de sensibilidad sobre los parámetros de las soluciones encontradas por los algoritmos. Nuestros experimentos muestran que CMA-ES encuentra mejores soluciones que los demás algoritmos en los distintos datasets de adopción *premium*. Nuestro análisis de sensibilidad muestra un amplio rango de valores para los coeficientes de imitación e innovación del modelo de Bass.**

***Index Terms*—calibración de modelos, metaheurísticas, modelado basado en agentes, modelo de negocio freemium**

I. Introducción

El modelo de negocio *freemium* combina la oferta de un producto o servicio sin coste con contenido *premium* que el usuario puede adquirir de manera opcional [1]. Este modelo se está extendiendo en aplicaciones *online* [2], donde están creando nuevos escenarios de marketing dado que este contexto propicia la adopción de productos a través de la interacción entre usuarios [3]. Con estos escenarios también surgen distintos problemas al planear campañas de marketing: qué individuos seleccionar para conseguir una campaña de marketing viral, cómo lidiar con *influencers*, o cómo incentivar a los usuarios a atraer a otros mandando invitaciones a través de sus redes [4]–[7]. Entender estos y otros efectos sociales detrás de las compras de contenido *premium* puede ser más fácil usando un modelo basado en agentes (ABM), puesto que permiten probar recompensas, incentivos, y políticas de *targeting* que incrementen el número de usuarios premium.

Un ejemplo es el ABM desarrollado para *Creature Party* [8]. Esta aplicación es un juego social multiplataforma para niños donde el usuario interactúa *online* con otros usuarios, que pueden ser *premium* o básicos. Un usuario básico tiene acceso total al juego pero un usuario *premium* recibe beneficios adicionales como una cantidad semanal de monedas del juego, la habilidad de adoptar mascotas virtuales, acceso a todos los avatares, y contenido reservado para usuarios *premium* como grupos y aventuras. La metodología ABM [9], [10] utiliza una población de entidades autónomas llamadas agentes que se comportan de acuerdo a reglas simples e interactúan unos con otros. Agregar estas reglas con la interacción entre los agentes nos permite representar dinámicas complejas y emergentes, así como definir escenarios *what-if* y predecir escenarios hipotéticos [11]. Este enfoque encaja con los patrones de crecimiento del mercado que resultan de la interacción de muchos usuarios, que son más complejos que cualquier adopción individual [12]. Gracias a sus aplicaciones exitosas, se ha incrementado el número de trabajos que emplean ABMs, particularmente para analizar la adopción de nuevos productos, políticas de adopción, y estrategias de *targeting* [13], [14].

Concretamente el ABM para *Creature Party* predice el número de usuarios básicos que adquieren contenido *premium* simulando periodos de tiempo específicos y monitorizando el comportamiento de los agentes. Este ABM utiliza información real de la red de usuarios para generar una red artificial [15] con una estructura similar en la que replicar el proceso social de adopción mediante el modelo de Bass [16]. También simula el comportamiento de usuarios siguiendo diferentes patrones estacionales, lo que es muy relevante para las aplicaciones que muestran diferente actividad entre semana que durante fines de semana.

En general, para usar un ABM es necesario validar su comportamiento calibrando sus parámetros. Llevamos acabo este procedimiento utilizando *data-driven automated calibration*, que consiste en modificar los parámetros del modelo de manera automática y así ajustar su salida a los datos reales. Este proceso es muy importante durante la validación del modelo [12], [17], [18]. Implementamos nuestro proceso de calibración empleando metaheurísticas [19] como algoritmo de búsqueda de la mejor configuración de parámetros del modelo. En este artículo presentamos una comparación de las siguientes metaheurísticas: *hill climbing* [19], algoritmos genéticos [20], *differential evolution* [21], *PSO* [22], y CMA-ES [23]. Elegimos estas metaheurísticas de modo que el grupo de métodos sea heterogéneo en cuanto a su complejidad, permitiendo apreciar la ganancia de rendimiento que consiguen métodos más avanzados. El rendimiento de las

metaheurísticas presentadas se basará en lo bien que el modelo calibrado replique los datos reales de *Creature Party*, que se presentan como la evolución de usuarios *premium* a lo largo del tiempo. Estos datos se presentan en distintos periodos de tiempo componiendo los distintos datasets que utilizaremos para validar las metaheurísticas para el problema. Una vez calibrado el modelo para *Creature Party* desarrollamos un análisis de sensibilidad sobre las soluciones generadas enfocándonos en los coeficientes de adopción del modelo de Bass y los parámetros de estacionalidad del modelo, utilizando técnicas de visualización de datos (*scatter plots*) para los valores obtenidos.

En la Sección II describiremos el ABM de adopción de contenido *premium* de *Creature Party*. La Sección III presenta las distintas metaheurísticas empleadas en el proceso de calibración. Mostramos la experimentación y el análisis de sensibilidad en la Sección IV. Por último, remarcamos nuestras conclusiones en la Sección V.

II. Descripción del modelo

La Sección II-A presenta la estructura general del modelo. En la Sección II-B se describen los mecanismos y el comportamiento de los agentes, incluyendo sus interacciones sociales en la aplicación. Finalmente presentamos el proceso de adopción seguido por el modelo en la Sección II-C.

A. Estructura general

El modelo simula la evolución del número de usuarios *premium* durante un periodo de tiempo determinado usando un *time step* diario. De este modo la salida del modelo consiste en los nuevos usuarios *premium* para cada paso de la simulación, que es el indicador principal de este tipo de mercados [24]. Observando la evolución de adopciones *premium* diarias podemos medir como de bien se ajusta el modelo a los datos históricos de las distintas instancias. El modelo ABM para *Creature Party* modela los usuarios existentes como agentes con un factor de escala de 2:1, por lo que 20000 agentes representan los 40000 usuarios activos de la aplicación. Esta escala se define para reducir el coste computacional manteniendo un buen nivel de representatividad.

Los agentes representan usuarios básicos o *premium* del total de usuarios de la aplicación, dependiendo del ratio inicial de usuarios *premium* α sobre el total de usuarios. Durante la inicialización del modelo, cada agente es inicializado como usuario básico o *premium* aleatoriamente. Debido a la aleatoriedad del modelo cada ejecución puede resultar en distintos patrones de difusión y resultados finales, por lo que usamos simulaciones de Monte-Carlo (MC) y ejecutamos múltiples veces la simulación. Cada agente toma decisiones asíncronamente, es decir, sin mecanismos de sincronización con el resto de agentes, dado que así el modelo se asemeja más a un modelo de simulación continuo, más cercano a la realidad [12].

B. Actualización del estado de los agentes y de su red

Los agentes tienen distintas variables de estado para representar la transición de usuario básico a *premium*. En el modelo un usuario pasa de básico a *premium* cuando compra en la tienda *in-game*. No consideraremos la transición inversa (esto es, dejar de ser *premium* para ser básico, *churn*), dado que no hay datos disponibles y no podríamos calibrar este comportamiento.

Cada agente tiene un conjunto de enlaces que representan las relaciones sociales entre usuarios de la aplicación. Estas relaciones son enlaces unidireccionales que habilitan el intercambio de información y canalizan influencias entre ellos. Los enlaces de la red social se generan usando un algoritmo basado en un grado de distribución dado [15], [25], con un grado medio de $48{,}19$ usuarios y una densidad de $0{,}0024$.

En cada paso del modelo, cada agente primero decide si jugar o no siguiendo una probabilidad que depende del día de la semana. Modelamos este comportamiento usando dos parámetros: (a) la probabilidad de jugar entre semana ($d \in [0,1]$) y (b) la probabilidad de jugar en fin de semana ($f \in [0,1]$). Si un agente juega a la aplicación, podrá decidir posteriormente si adoptar contenido *premium* en base a la influencia social de sus contactos (imitación) o por su propia iniciativa (innovación). Este proceso de adopción sigue el modelo de Bass [12], [16], descrito en la Sección II-C. La probabilidad de jugar de cada agente sigue la Ecuación 1. El agente A decide jugar el día t si la función $jugar_t^A(r)$ devuelve 1, donde $r \in [0,1)$ es un número aleatorio generado usando una distribución uniforme.

$$jugar_t^A(r) = \begin{cases} 1, & \text{si } r < d \wedge t \in entre{-}semana, \\ 1, & \text{si } r < f \wedge t \in fin{-}semana, \\ 0, & \text{e.o.c.} \end{cases} \quad (1)$$

C. Modelo de adopción

El modelo de adopción elegido es la versión ABM del modelo de Bass [8], [16]. En el modelo de adopción de Bass, un agente básico puede hacerse *premium* con una probabilidad que incluye tanto el efecto de la publicidad (externalidad) como el de la influencia social o boca a boca. La probabilidad de que un agente se haga *premium* debida a la interacción con otros agentes se regula por la fracción de sus vecinos que se han hecho *premium* en los pasos anteriores. Por tanto, en cada paso un agente básico i puede hacerse *premium* debido a dos circunstancias:

1. Innovación: con probabilidad $\hat{p}$ (coeficiente de innovación), $\hat{p} \in (0,1)$, un agente básico puede hacerse *premium* debido a información externa a la red.
2. Imitación: con probabilidad $\hat{q}$ (coeficiente de imitación), $\hat{q} \in (0,1)$, un agente básico puede hacerse *premium* observando el estado de sus vecinos, donde f es la fracción de vecinos *premium*.

En estas circunstancias podemos modelar el proceso de adopción *premium* siguiendo la Ecuación 2 [12]. Un agente A puede hacerse *premium* en un step t si $adoptar$ devuelve 1, donde $r, s \in [0,1)$ son números aleatorios generados usando una distribución uniforme y f_A es la fracción de vecinos *premium* de A.

$$adoptar_t^A(r,s) = \begin{cases} 1, & \text{si } r < \hat{p} \vee s < f_A \hat{q}, \\ 0, & \text{e.o.c.} \end{cases} \quad (2)$$

III. Metaheurísticas para calibración

En esta sección presentamos nuestra propuesta para calibrar ABM usando métodos automáticos basados en metaheurísticas. La calibración automática es un proceso intensivo que usa una medida de error para comparar la salida del modelo con datos reales, modificando sus parámetros para encontrar la configuración que más se ajuste a los datos reales [17], [18]. Medimos la calidad del modelo ejecutando una simulación y comparando su salida con los datos reales. En este trabajo hemos considerado metaheurísticas para calibración dado que los parámetros del ABM no muestran interacciones lineales y la mejor opción es usar un algoritmo de optimización no-lineal que pueda gestionar un espacio de búsqueda muy amplio [17], [26], [27]. Aun así, elegir la metaheurística más adecuada para calibrar ABM no es trivial dado que hay que tener en cuenta dos criterios opuestos: la exploración del espacio de búsqueda (diversificación) y la explotación de las soluciones más prometedoras (intensificación).

En general, las metaheurísticas elegidas modificarán el conjunto de parámetros del modelo para ajustarse a los datos reales de las distintas instancias. Calibramos cuatro parámetros reales del modelo anteriormente descrito: los dos parámetros que regulan la estacionalidad $d, f \in [0,1]$, el coeficiente de innovación $\hat{p} \in [0,1]$, y el coeficiente de imitación $\hat{q} \in [0,1]$. Las metaheurísticas seleccionadas usan la misma función de *fitness*, que determina la calidad del modelo con relación a los datos reales de usuarios *premium*. Para evaluar el ajuste de un modelo utilizamos cada *dataset* R y lo dividimos en R_{train} y R_{test} para seguir un enfoque de validación *hold-out* [27]. Ambos conjuntos tienen sus variables de entorno correspondientes E_{train} y E_{test}, que definen las condiciones como el estado de la red social y el número de usuarios *premium*. Utilizaremos 15 ejecuciones de MC, siendo la estimación de error $\epsilon(R_{train}, M(P^*, E_{train}))$ la media del ajuste en estas 15 ejecuciones de la simulación. En nuestro caso elegimos L^2 o distancia euclídea para calcular el error ϵ:

$$L^2 = \sqrt{\sum_{i=0}^{n-1} |V_M(i) - V_R(i)|^2},$$

donde n es el número de puntos del histórico y V_M y V_R son los vectores de salida (nuevos usuarios *premium*) del modelo y de la instancia respectivamente. Por último, hemos implementado las metaheurísticas en Java utilizando el *framework* ECJ [28]. Las secciones siguientes presentan los detalles de las metaheurísticas elegidas: *hill climbing* [19], algoritmos genéticos [20], *differential evolution* [21], *PSO* [22], y CMA-ES [23].

A. HC

Hemos definido una búsqueda local para comparar su rendimiento con las metaheurísticas poblacionales. Nuestro procedimiento de búsqueda local implementa una estrategia tipo *hill climbing* [19]. Partiendo de una solución aleatoria, refina la calidad de la solución incrementando o decrementando el valor de uno de los parámetros de manera iterativa. Nuestra búsqueda local considera la variable $incremento$, que regula la variación aplicada a los parámetros de la solución. En cada iteración, la búsqueda local se moverá al primer vecino que mejore su *fitness* por un $umbral$ dado.

B. GA

Diseñamos nuestro algoritmo genético (GA) siguiendo una estrategia generacional donde cada nueva generación reemplaza a la anterior [20]. Fijamos el tamaño de población del algoritmo en 100 invididuos y el el número de generaciones en 200. Nuestra implementación usa selección por torneo [19], con k siendo el tamaño del torneo. También incluimos elitismo débil, por lo que el mejor padre siempre pasa a la siguiente generación. Como estrategia de cruce elegimos un cruce multipunto *PMX* con probabilidad p_c para cada individuo. Respecto a la estrategia de mutación, cada individuo tendrá una probabilidad p_m de reiniciar uno de sus parámetros (genes), sustituyendo su valor por un número aleatorio usando una distribución uniforme en el intervalo $[0,1]$.

C. PSO

Para nuestra implementación de *Particle Swarm Optimization* (PSO) [22] consideramos también una población de 100 individuos y 200 iteraciones. Cada individuo considera 5 vecinos que se asignan aleatoriamente durante la inicialización del algoritmo. En cada iteración, cada individuo se mueve (modifica sus valores) considerando su mejor configuración conocida, el mejor de sus vecinos, y el mejor global. Este movimiento se calcula utilizando cuatro parámetros que actúan como pesos: velocidad (v_c), mejor personal (p_c), mejor vecino (i_c), y mejor global (g_c).

D. DE

El diseño que hemos elegido para *differential evolution* (DE) [21] considera una población de 100 invididuos y se ejecuta durante 200 generaciones. La variante utilizada es *DE/rand/2*, que genera el vector donante a partir de la siguiente ecuación: $x_i(G+1) = x_{r1}(G) + F(x_{r2}(G) - x_{r3}(G))$, donde x_i es el vector generado y $r1$, $r2$, y $r3$ son soluciones diferentes. F es un parámetro del algoritmo que actúa de amplificador de la mutación. Esta variante genera nuevos individuos usando el vector donante con una probabilidad definida por el parámetro CR.

E. CMA-ES

Covariance matrix adaptation evolution strategy (CMA-ES) [23] es otro algoritmo bio-inspirado que desarrolla la mutación de su población (o recombinación) de acuerdo a una matriz de covarianza. CMA-ES suele funcionar muy bien con problemas similares al nuestro, ya que los parámetros son valores reales y el tamaño del conjunto a calibrar es pequeño. Hemos elegido la variante $(\mu/\mu_I, \lambda)$-CMA-ES, donde en cada iteración se combinan los mejores μ candidatos con λ nuevas soluciones.

IV. Experimentación

A. Configuración

La Tabla I muestra los parámetros que hemos usado para nuestros experimentos. Nuestro proceso de calibración considera datos reales de *Creature Party* divididos en cuatro datasets diferentes. Estos datasets contienen el número de usuarios premium por día para periodos de tiempo distintos que a su vez dividimos en *training* y *test*. El primer dataset contiene 91 días, con 60 para *training* y 31 para *test*. El segundo y el tercer dataset consideran 46 días, divididos en 31 días para *training* y 15 para *test*. El cuarto y último dataset contiene 45 días, con 30 para *training* y 15 para *test*.

Tabla I: Valores de los parámetros de las metaheurísticas

General		**GA**		**PSO**	
Nombre	Valor	Nombre	Valor	Nombre	Valor
Evaluaciones	20,000	p_c	1.0	v_c	0.3
Individuos	100	p_m	0.2	p_c	0.04
Generaciones	200	k	3	i_c	0.3
Monte-Carlo	15			g_c	0.3
HC		**DE**		**CMA-ES**	
Nombre	Valor	Nombre	Valor	Nombre	Valor
$incremento$	0.01	F	0.6	λ	8
$umbral$	0.01	CR	0.3	μ	4

B. Resultados de calibración

Mostramos los resultados de calibración obtenidos por las metaheurísticas en la Tabla II. Estos resultados muestran el ajuste de la mejor solución después de 15 ejecuciones con distintas semillas. Este ajuste se presenta como el *fitness* medio y la desviación típica de las 15 simulaciones de MC para *training* y *test*. Como se puede observar, el *fitness* de la mayoría de metaheurísticas es mejor para los datasets más pequeños (2, 3, y 4), lo cual parece coherente dado que su comportamiento debería ser más fácil de ajustar. Por otro lado, los valores de desviación típica son similares. Con respecto al ajuste de las soluciones, estros resultados muestran que CMA-ES supera a los otros algoritmos por un pequeño margen. Además de valores de fitness bajos, CMA-ES consigue unos valores de desviación típica reducidos, por lo que se muestra estable con distintas semillas.

Mostramos el rendimiento de los algoritmos poblacionales durante el proceso de calibración usando el dataset 3 en la Figura 1. Esta gráfica muestra la evolución del *fitness* de la mejor solución en cada generación, por lo que el progreso de los algoritmos se muestra cada 100 evaluaciones. Estos resultados muestran que los algoritmos alcanzan valores de error reducidos rápidamente y sugieren que aumentar el número de evaluaciones no mejoraría el resultado final de manera significativa.

En la Figura 2 mostramos una comparación entre los datos históricos y la salida del modelo calibrado para la mejor solución encontrada por CMA-ES para el segmento *training* del dataset 2. Estos resultados muestran que el proceso de calibración obtiene buenos resultados y que el modelo calibrado

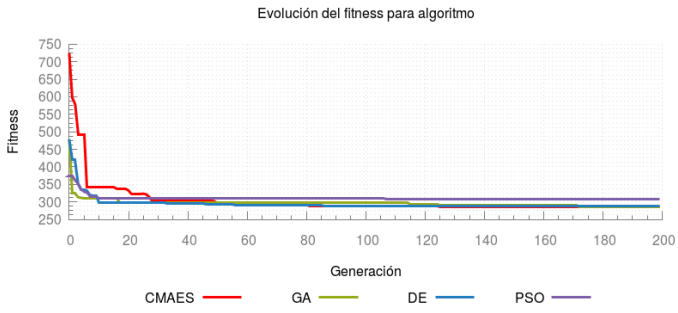


Figura 1: Evolución del *fitness* de los algoritmos poblacionales para el dataset 3.

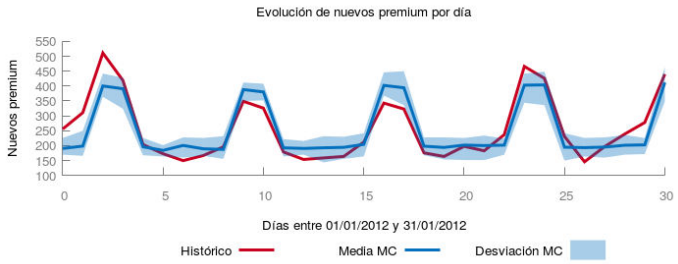


Figura 2: Comparación entre la salida del modelo calibrado y los datos históricos de *training* del dataset 2 (31 días).

se ajusta a los datos históricos. Por ejemplo en el step 4, el número de usuarios *premium* fue 204 y la simulación predice 196. Aunque hay excepciones, como el día 2 donde el número de usuarios *premium* fue 511 pero el modelo predice 404.

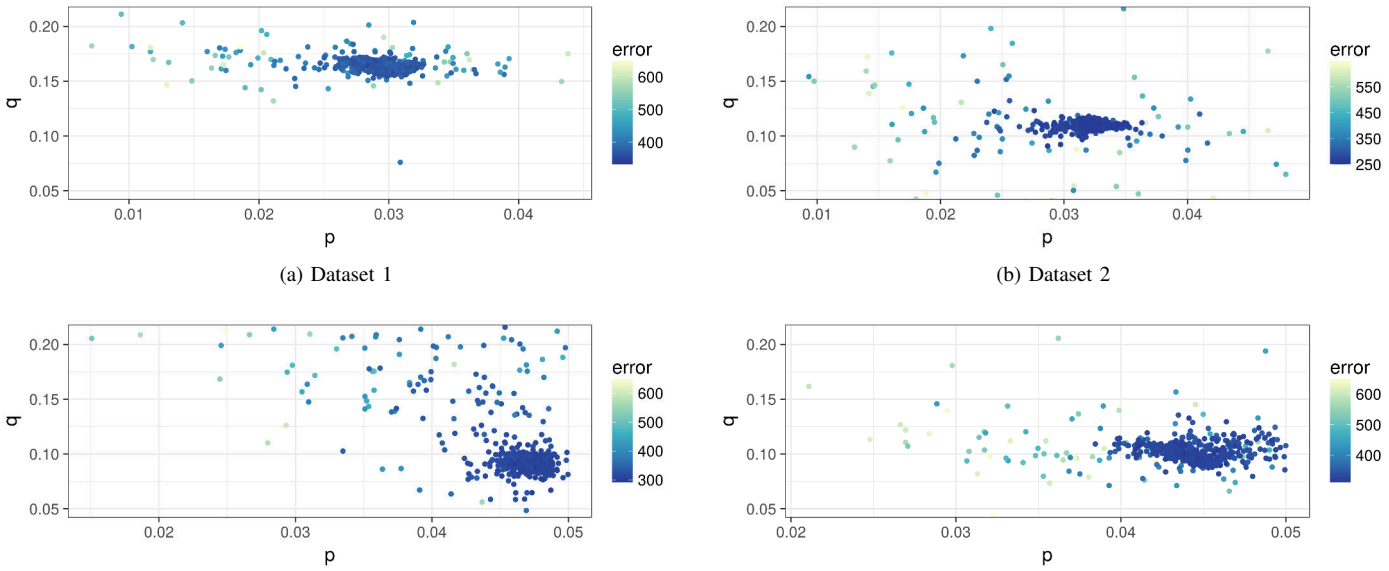
C. Análisis de sensibilidad

Nuestros experimentos señalan CMA-ES como el mejor método, consiguiendo los mejores resultados en tres conjuntos de *training* y uno de *test*. En esta sección hacemos un análisis de sensibilidad a las soluciones generadas por CMA-ES para cada dataset. Los *scatter plots* de las Figuras 3 y 4 muestran los valores de los coeficientes de Bass y los parámetros de estacionalidad para los soluciones generadas por CMA-ES durante el proceso de calibración. Estas soluciones provienen de una única ejecución de la metaheurística, recogidas desde el principio de la calibración hasta que el algoritmo termina, filtrando las soluciones que superen 500 de *fitness*. Este filtro permite concentrar el análisis en las mejores soluciones.

Podemos observar que los valores de los coeficientes de Bass de las soluciones para los distintos dataset son similares. Por ejemplo, para el primer dataset (Figura 3a) las mejores soluciones rondan $\hat{p} = 0{,}03$ y $\hat{q} = 0{,}16$. En cambio el coeficiente de imitación es ligeramente más bajo en el resto de datasets ($\hat{q} = 0{,}1$). En el caso del coeficiente de innovación, los dos primeros dataset (Figura 3a y 3b) muestran sus mejores valores alrededor de $\hat{p} = 0{,}03$ y los dos siguientes (Figura 3c y 3d)) alrededor de $\hat{p} = 0{,}045$. En estos últimos observamos también mayor dispersión en sus soluciones con

Tabla II: Fitness medio y desviación típica de cada metaheurística para cada dataset

	Dataset 1		Dataset 2		Dataset 3		Dataset 4	
	TRA	TEST	TRA	TEST	TRA	TEST	TRA	TEST
HC	365.6 (18.7)	349.2 (21.9)	256.0 (17.2)	211.0 (20.3)	301.1 (19.8)	351.7 (15.7)	336.1 (30.3)	**222.1 (20.1)**
GA	351.9 (19.6)	333.7 (22.4)	247.6 (24.8)	182.1 (19.1)	286.0 (22.4)	331.4 (20.0)	304.5 (26.0)	283.7 (31.5)
DE	351.8 (18.9)	335.3 (20.4)	**245.3 (22.0)**	194.8 (20.5)	286.8 (21.6)	348.0 (17,0)	**301.6 (17.6)**	283.5 (34.2)
PSO	347.2 (15.0)	332.5 (20.8)	246.1 (13.8)	191.7 (19.5)	308.0 (19.7)	**261.5 (22.7)**	316.5 (31.5)	365.2 (20.4)
CMA-ES	**326.9 (18.2)**	**296.4 (15.6)**	248.1 (13.9)	**164.8 (17.0)**	**286.0 (16.9)**	318.1 (22.3)	303.3 (24.4)	267.2 (26.4)



(a) Dataset 1

(b) Dataset 2

(c) Dataset 3

(d) Dataset 4

Figura 3: *Scatter plots* para el análisis de sensibilidad sobre los coeficientes de Bass (p/q).

mejor *fitness*, lo que podría indicar que estos dataset tienen mayor tolerancia a distintos valores de estos parámetros. Globalmente, se puede observar que las soluciones muestran mayor dispersión para el coeficiente de innovación ($\hat{p}$), lo que indica que es un parámetro más sensible. En comparación con otras configuraciones del modelo de Bass para otros mercados publicados anteriormente en la literatura de marketing [12], los modelos calibrados presentan un valor alto para el coeficiente de innovación ($\hat{p}$) y un valor bajo para el coeficiente de imitación ($\hat{q}$).

Por último, los *scatter plots* de la Figura 4 muestran los valores de estacionalidad para los soluciones encontradas para los distintos datasets. Estas soluciones muestran valores constantes en los diferentes datasets, con valores que rondan d=0,1 para la probabilidad de jugar entre semana y f=0,2 para la probabilidad de jugar en fin de semana. Estos parámetros también muestran mayor concentración en el parámetro (d), lo que sugiere que el ABM de *Creature Party* es más sensible a los cambios en la probabilidad de jugar entre semana que en para la probabilidad de jugar en el fin de semana.

V. Conclusiones

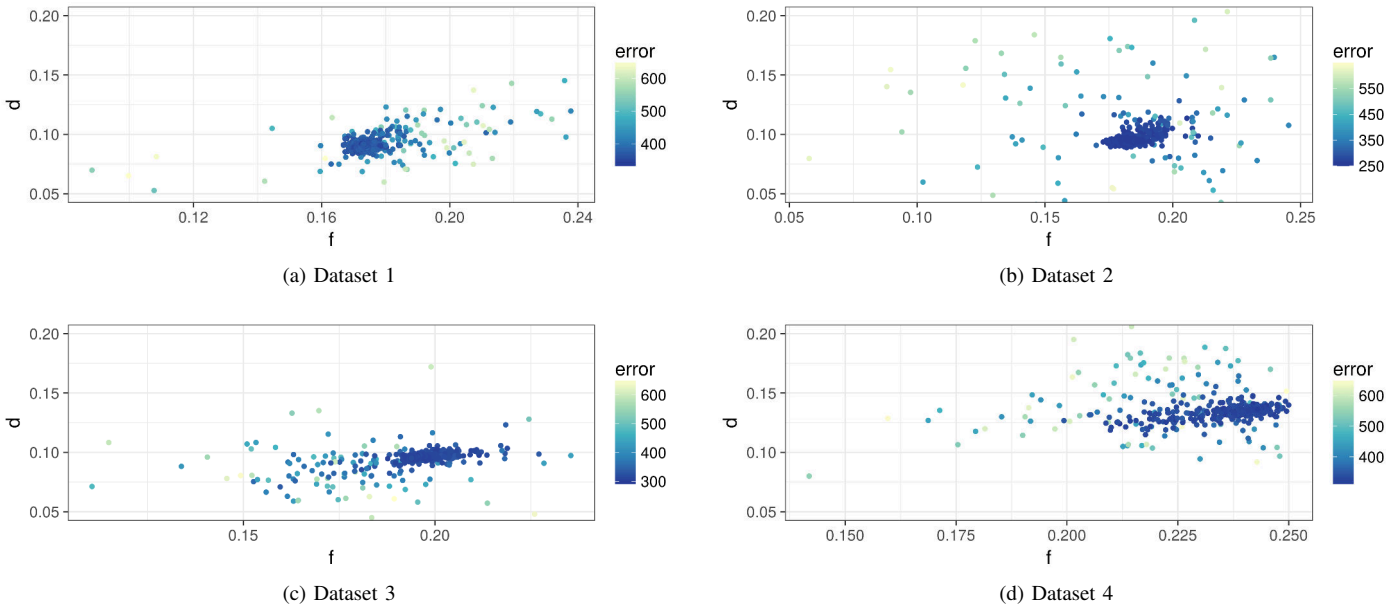
En este artículo hemos aplicado distintas metaheurísticas a la calibración de un modelo ABM usando datos reales. Después de comparar el rendimiento de los algoritmos, hemos elegido las soluciones calibradas por CMA-ES, la mejor metaheurística según nuestra experimentación, para desarrollar un análisis de sensibilidad sobre sus parámetros. Usando estas soluciones, hemos visualizado los valores de los coeficientes del modelo de Bass y los parámetros de estacionalidad.

Los resultados de calibración de las distintas metaheurísticas muestran valores de *fitness* cercanos. Sin embargo, CMA-ES consigue mejores soluciones por lo que lo distinguimos como el mejor método, lo cual es coherente con publicaciones anteriores sobre optimización continua utilizando metaheurísticas.

El análisis de sensibilidad aplicado sobre las soluciones calibradas por CMA-ES muestra que sus parámetros para los distintos dataset son similares. Además estos valores son diferentes de otros modelos similares [12], lo que podría indicar que las aplicaciones *freemium* como *Creature Party* tienen un componente de innovación mayor. Por último, dado que nuestro análisis no tiene en cuenta las características específicas del diseño del ABM subyacente, pensamos que nuestros resultados podrían ser generalizables a otros modelos.

Agradecimientos

Este trabajo está financiado por el Ministerio de Economía y Competitividad bajo el proyecto NEWSOCO (ref. TIN2015-67661-P), incluyendo Fondos Europeos de Desarrollo Regional (ERDF).



(a) Dataset 1

(b) Dataset 2

(c) Dataset 3

(d) Dataset 4

Figura 4: *Scatter plots* con los valores de los parámetros de estacionalidad: entre semana (d) y fin de semana (f).

Referencias

[1] V. Kumar, "Making freemium work: Many start-ups fail to recognize the challenges of this popular business model." *Harvard Business Review*, vol. 92, no. 5, pp. 27–29, May 2014. [Online]. Available: https://hbr.org/2014/05/making-freemium-work

[2] M. Trusov, R. E. Bucklin, and K. Pauwels, "Effects of word-of-mouth versus traditional marketing: Findings from an internet social networking site," *Journal of Marketing*, vol. 73, no. 5, pp. 90–102, 2009.

[3] B. Libai, R. Bolton, M. S. Bügel, K. De Ruyter, O. Götz, H. Risselada, and A. T. Stephen, "Customer-to-customer interactions: broadening the scope of word of mouth research," *Journal of Service Research*, vol. 13, no. 3, pp. 267–282, 2010.

[4] O. Hinz, B. Skiera, C. Barrot, and J. U. Becker, "Seeding strategies for viral marketing: An empirical comparison," *Journal of Marketing*, vol. 75, no. 6, pp. 55–71, 2011.

[5] R. Van der Lans, G. Van Bruggen, J. Eliashberg, and B. Wierenga, "A viral branching model for predicting the spread of electronic word of mouth," *Marketing Science*, vol. 29, no. 2, pp. 348–365, 2010.

[6] D. J. Watts and P. S. Dodds, "Influentials, networks, and public opinion formation," *Journal of Consumer Research*, vol. 34, no. 4, pp. 441–458, 2007.

[7] P. Schmitt, B. Skiera, and C. Van den Bulte, "Referral programs and customer value," *Journal of Marketing*, vol. 75, no. 1, pp. 46–59, 2011.

[8] M. Chica and W. Rand, "Building agent-based decision support systems for word-of-mouth programs: A freemium application," *Journal of Marketing Research*, vol. 54, no. 5, pp. 752–767, 2017. [Online]. Available: https://doi.org/10.1509/jmr.15.0443

[9] C. M. Macal and M. J. North, "Tutorial on agent-based modeling and simulation," in *Proceedings of the 37th conference on Winter simulation*. ACM, 2005, pp. 2–15.

[10] J. M. Epstein, *Generative social science: Studies in agent-based computational modeling*. Princeton University Press, 2006.

[11] M. A. Janssen and E. Ostrom, "Empirically based, agent-based models," *Ecology and Society*, vol. 11, no. 2, p. 37, 2006.

[12] W. Rand and R. T. Rust, "Agent-based modeling in marketing: Guidelines for rigor," *International Journal of Research in Marketing*, vol. 28, no. 3, pp. 181–193, 2011.

[13] M. Trusov, W. Rand, and Y. V. Joshi, "Improving prelaunch diffusion forecasts: Using synthetic networks as simulated priors," *Journal of Marketing Research*, vol. 50, no. 6, pp. 675–690, 2013.

[14] M. Haenlein and B. Libai, "Targeting revenue leaders for a new product," *Journal of Marketing*, vol. 77, no. 3, pp. 65–80, 2013.

[15] F. Viger and M. Latapy, "Efficient and simple generation of random simple connected graphs with prescribed degree sequence," in *Lecture Notes in Computer Science. Computing and Combinatorics*. Springer, 2005, vol. 3595, pp. 440–449.

[16] F. M. Bass, "A new product growth model for consumer durables," *Management Science*, vol. 36, no. 9, pp. 1057–1079, 1969.

[17] R. Oliva, "Model calibration as a testing strategy for system dynamics models," *European Journal of Operational Research*, vol. 151, no. 3, pp. 552–568, 2003.

[18] R. G. Sargent, "Verification and validation of simulation models," in *Proceedings of the 37th conference on Winter simulation*, 2005, pp. 130–143.

[19] E.-G. Talbi, *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009.

[20] T. Back, D. B. Fogel, and Z. Michalewicz, *Handbook of evolutionary computation*. Bristol (UK): IOP Publishing Ltd., 1997.

[21] K. Price, R. M. Storn, and J. A. Lampinen, *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media, 2006.

[22] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, vol. 4. IEEE, 1995, pp. 1942–1948.

[23] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary computation*, vol. 9, no. 2, pp. 159–195, 2001.

[24] P. W. Farris, N. T. Bendle, P. E. Pfeifer, and D. J. Reibstein, *Marketing metrics: The definitive guide to measuring marketing performance*, 2nd ed. Wharton School Publishing, 2010.

[25] R. Milo, N. Kashtan, S. Itzkovitz, M. Newman, and U. Alon, "On the uniform generation of random graphs with prescribed degree sequences," *arXiv preprint http://arxiv.org/abs/cond-mat/*, 2004.

[26] J. H. Miller, "Active nonlinear tests (ANTs) of complex simulation models," *Management Science*, vol. 44, no. 6, pp. 820–830, 1998.

[27] F. Stonedahl and W. Rand, "When does simulated data match real data? Comparing model calibration functions using genetic algorithms," in *Advances in Computational Social Science*, ser. Agent-Based Social Systems. Springer, Japan, 2014, vol. 11, pp. 297–313.

[28] S. Luke, L. Panait, G. Balan, S. Paus, Z. Skolicki, J. Bassett, R. Hubley, and A. Chircop, "Ecj: A java-based evolutionary computation research system," *Downloadable versions and documentation can be found at the following url: http://cs. gmu. edu/eclab/projects/ecj*, 2006.