WEB APPENDIX A: APPROPRIATENESS OF USING AGENT-BASED MODELING FOR MANAGING WORD-OF-MOUTH PROGRAMS

Rand and Rust (2011) proposed a set of indicators for when to use agent-based modeling (ABM). We describe in this section why agent-based models are appropriate when building a decision support system (DSS) for word-of-mouth (WOM) programs. The main reason is that analyzing WOM programs requires understanding groups of heterogeneous consumers and the emergent results of those consumers' interactions. In this list, we go through each of the indicators individually:

- **Medium number of users or customers**: In most WOM programs, the size of the set users or customers to be analyzed is in the medium range. If a key set of influentials exert their influence the system can dramatically change. ABM is useful when attempting to understand key interactions in complex systems.

- **Local and potentially complex interactions**: There are local and potentially complex interactions between customers in WOM programs. For instance, customers can be influenced by their friends, and some of those friends may have a different amount of influence on them than other friends. This is easily captured in an ABM since each user can be modeled with their own social network connection and the weights on those connections.

- **Heterogeneity**: WOM customers are very heterogeneous. They have both their own social networks that are unique to them, and potentially their own adoption process as well. With ABM, each agent can be modeled as differently from other agents as necessary (Rand and Rust 2011). In our case, users or customers could have different seasonality, different number of friends in her/his local social network, and different subscription status.

- **Rich environments**: The environment for a WOM DSS is a network-based space derived from the real social network when possible. ABM gives us the ability to capture this complex environment and study the user's adoption decision as a result of the effect of the local network of friends on each user.

- **Temporal aspects**: A typical goal of DSSs for WOM programs is to forecast a KPI for different time horizons. Therefore, the problem has a temporal nature and this is a necessary condition for the ABM approach. We need to identify when (daily, weekly or monthly) customers are likely to adopt a product or a service. We do this by simulating the daily activities over time.

- **Adaptive agents**: Users in WOM situations make daily/weekly/monthly decisions that have consequences when adopting or not. These decision are partially based on previous recommendations of their social network of friends. Then, they dynamically change their mental state based on their new trust network. Creating a model that has adaptive elements in it is difficult to do in any framework other than ABM.

WEB APPENDIX B: MODEL VERIFICATION AND VALIDATION

Model Verification

Verification is the process of making sure the implemented model matches the conceptual model. We can verify how well the implemented models for the Animal Jam DSS correspond to the conceptual models. Additionally to this Web appendix, we have uploaded the Java source code and Javadoc documents to a software repository, publicly available at *https://bitbucket.org/mchserrano/socialdynamicsfreemiumapps*. The three main verification steps, identified by Rand and Rust (2011), are: documentation, programmatic testing, and test cases.

The model was programmed in Java and we perform a set of walkthroughs, unit tests, and debugging walkthroughs using the Eclipse platform (*https://eclipse.org*) to ensure that the software project behaves correctly (programmatic testing). A lot of test cases were defined for the main components of the model to ensure its total verification. Mainly, the information diffusion models (agent-based version of the Bass model (Bass-ABM) and the two variants of the complex contagion model) were intensively analyzed by setting test cases. For instance, we verified corner cases and compared the software output with the expected behavior such as $\hat{p}, \hat{q} = 0, 0$ or $\hat{p}, \hat{q} = 1, 0$ for the Bass-ABM; and $\phi = 0$ or $\phi = m$ for the complex contagion, where m is the maximum possible number of friends. We do this in conjunction with different values for the initial number of premium adopters in the population (α rate parameter of the model) and other critical parameters of the model (e.g. usage seasonality). Additionally, we run random cases, i.e., cases where we choose random parameter values from the range of potential parameter values, and specific scenarios to check the cases when all the users adopt, when there is an equilibrium state, or when there is no adoption activity.

Documentation and ODD Description

First, the programming code is documented following the Javadoc specifications (*http://www.oracle.com/technetwork/articles/java/index-137868.html*). HTML and PDF documents were automatically generated from the code documentation to complement the software with a

developers' guide. You can also download documentation from *https://bitbucket.org/mchserrano/socialdynamicsfreemiumapps*.

The conceptual model is documented by using the overview, design concepts and details (ODD) protocol (Grimm et al. 2010) to describe all the components of the model:

Objective. The purpose of this model is to better understand the dynamics of premium and paid content adoption in apps such as online game subscriptions. The goal is to be used by marketers and decision makers to understand the social effects in these online purchases, and to test rewarding, incentives, and targeting policies to expand the number of premium users. This model will help us to better understand the dynamics of the social network of the app users.

Entities, state variables and scales. The basic entity in the model is a user of the app. These entities exist within the scope of the social network of the app. Then, another basic entity of the model is the relationship between two users (agents or individuals of the ABM). This relationship is unidirectional and potentially has a weighted social influence between them. These links enable the information exchange between the users and also provide a channel to exert influence between them, which directly affects their decisions.

The agents of the model have different state variables. First, they are either a basic or premium user. Being a premium user means the agent has already adopted or consumed the in-app/in-game purchase. Within our model, when an agent adopts the premium state (s)he never comes back to the basic state. This was motivated by the interests of the managers who were interested in conversion and not churn, and it simplified the model. All the agents possess a set of links which are the social relationships of that user to other users of the app. As to the scale of the model, there is a one-to-two mapping between the real app user and the agent of the model. For the experiments, the number of agents in the model is 20,000 which maps to 40,000 active users of the app. We set this one-to-two mapping to reduce the computational time of the simulations while getting good fitting results in a preliminary experimentation.

The time step of this model is one day. When validating, all the simulations are run for the needed time to match the historical data (61 days for the training data and 31 days for the test

dataset). The model outputs new premium adoptions every day and the cumulative number of them during the current simulation, which is the KPI we are trying to match.

Process overview and scheduling. The underlying process model is simple. The basic idea is that all the agents are initialized to basic or premium states depending on the rate of real premium users with respect to all the app users. The links of the social network are generated by using a prescribed degree distribution algorithm (Viger and Latapy 2005, Milo et al. 2004) to be close to the empirical distribution degree of the app. See in Figure 1 the degree distribution of the social network which has an average degree of 48.19 users and a density of 0.0024. The social weighted influence of the links between two users i and j, τ_{ij}, are set to a value $\in [1, n]$ where n is the number of users (upper bound). When τ_{ij} is set to 1, the effect of having different weighted influences is suppressed.

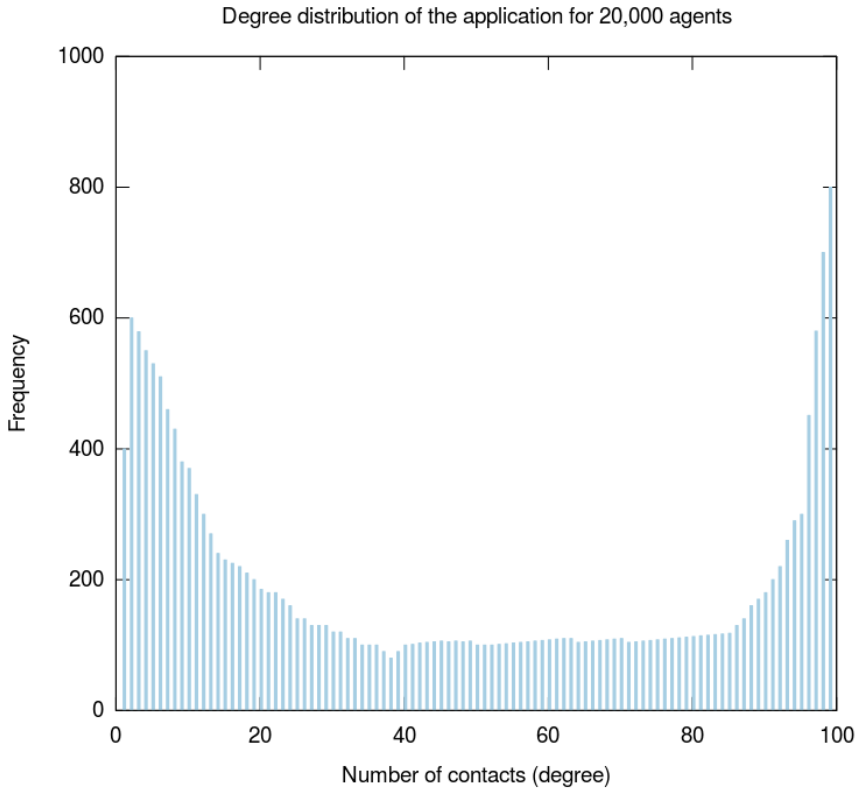

Figure 1: Degree distribution of the synthetic social network of 20,000 agents, generated for the Animal Jam agent-based DSS.

At each time step (day), the agents have a probability to play the app, which is dependent on

the day of the week, since we want to model the seasonality of play. When adapting the general model for the hedonic app we only consider two seasonality parameters which showed to be sufficient for the problem: (a) the probability of playing in a weekday ($\gamma_0 \in [0, 1]$) and (b) the probability of playing in a weekend day ($\gamma_1 \in [0, 1]$).

If an agent plays the app on a particular day then (s)he will be able to perform more actions afterwards during this day. If (s)he does not play, no action is required for this agent at this particular time step. While playing the app, the agent decides whether to adopt the premium content or not on the basis of the social influence of her/his contacts (internal influence) or on her/his own decision (external influence). This general adoption process changes depending on the three diffusion models considered in our study: the Bass-ABM, the traditional complex contagion model, and the complex contagion model with an additional external influence coefficient. The model update is asynchronous. This means that an agent decides whether to adopt or not based on her/his local network of premium friends including friend whose adoption occurred in that very same time step. Asynchronous updating creates a model which is closer to a continuous model (Rand and Rust 2011).

Design concepts. We explain the design concepts of the model in the next paragraphs. This describes how we went about designing the basic concepts of the model:

- *Basic principles:* Our basic goal is to explore and understand the role of social influence on the adoption of premium and in-app purchases. The goal is to use this model to help marketers to understand the value of WOM when making marketing decisions and to test rewarding and targeting policies to expand premiums.

- *Emergence:* The key aspect of this model that is emergent is the overall adoption rate of premium content and purchases. This is modeled by assuming that a small percentage of the agents are premium at the beginning of the simulation (given by ratio α) and they spread their information with other basic users at every time step.

- *Adaptation:* Agents are adaptive since they innovate by themselves and react to changing

social influence from their direct links in the social network. This adaptation clearly impacts the decision of purchasing in-app premium contents.

- *Objectives, learning and predictions:* Agents do not have detailed objectives in the current model. They make decisions at every step to play or not and to adopt premium content or not. Agents do not learn and do not predict future situations in the current version of the model.

- *Sensing:* The agents can sense the state of their immediate contacts (local social network). For instance, they know how many of their close friends have become premium users. Users can also sense advertising and information about the game and make a decision to adopt on their own.

- *Interaction:* Agents interact with each other by exchanging new information. Each agent determines whether or not to adopt the new information based on the diffusion models described below.

- *Stochasticity:* At each time step of the model, each agent first decides to play the app or not with a given probability depending on the week of the day. If an agent plays the app, then agents who have not adopted premium content yet draw random numbers on the uniform interval [0, 1] to determine if they should adopt premium or not. Because of this stochasticity, each run of the model can result in different diffusion patterns and output results. This fact makes it necessary to run a Monte-Carlo (MC) simulation for each model result.

- *Collectives:* There is one major collective in the model, which is the social network connecting all of the users of the app.

Observation. The observations we are concerned is the evolution of daily premium adoptions. We will use this measure in order to compare sub-models to better fit the historical data with the evolution of adoptions (macro-level forecast). We will also use the adoption model

to forecast the most likely agents to adopt premium content in the near future in order to understand targeting models (micro-level forecast).

Initialization. The following steps are the major parts of the initialization:

1. Create a number of agents to represent the users of the app at a rate of one-to-two for the real users.
2. Initialize all the agents with their states (basic or premium) depending on the rate α of premiums over the total number of users existing in the real app. These users are chosen uniformly randomly.
3. Set seasonality parameters (γ_0, γ_1) for joining or playing for the two seasonality periods, i.e., weekends versus weekdays.
4. Set the parameters for the information diffusion models depending on the model being examined: $\hat{p}$ and $\hat{q}$ for the Bass-ABM; and the complex threshold $\phi \in [0, m]$, where m is the maximum number of contacts of a user (100 in our case) for the complex contagion model. The parameter $\hat{p}$ is also set for the complex contagion model with external influence.
5. Create a social network from a given sequence degree to simulate the real app and connect the created agents. We use a fast social network generation algorithm, published in Viger and Latapy (2005), to create an artificial social network with the prescribed degree sequence.

Input data. The major source of input data is from the real app, Animal Jam. Its datasets contain the subscriptions to premium content of their users for several months in 2012 (about 1.4 million users). Also, the details of the real social network connections for a subset of users of the game is provided, which gives us the ability to analyze the network and obtain its degree distribution.

Diffusion sub-models. Our ABM model can use three different models which are involved in the decision to adopt premium content: (1) the Bass-ABM (Bass 1969, Rand and Rust 2011), (2) the complex contagion model (Centola and Macy 2007), and (3) the complex contagion model with an additional external influence coefficient.

- Agent-based Bass model (Bass-ABM): based on a hazard-rate model originally developed to understand the adoption of consumer durables (Bass 1969). The Bass-ABM is a discrete-time model in which each agent has one of two states at each time step t (Rand and Rust 2011): (1) basic (unaware) or (2) premium (aware). At each time step, a basic agent has an opportunity to become premium. Its state changes with a probability that reflects advertising and WOM effects. The probability that an agent becomes premium due to WOM increases as a function of the fraction of its neighbors who became premium in previous time steps. Once an agent becomes premium, it remains premium for the rest of the simulation. At each time step, a basic agent i can become premium due to one of two circumstances:

    1. External influence - With probability $\hat{p}$, a basic agent becomes premium due to outside effects, i.e., information from outside the network, where $\hat{p}$ is the external influence coefficient.
    2. Internal influence - With probability $\hat{q}$, a basic agent becomes premium due to observing the state of its neighbors, where f is the fraction of neighbors who have adopted and $\hat{q}$ is the internal influence coefficient. As our model can support bidirectional social influences by using a weighted social network, the fraction f is multiplied by all the τ_{ij} weights of the set of direct neighbors of the individual (i.e., a multiplicative combination of the social influences of the neighbors). When $\tau_{ij} = 1$ for all j neighbors of i, the internal influence value is calculated as in the traditional model with homogeneous influence.

- Complex contagion: the complex contagion model can be seen as a modified model of a threshold rule adoption mechanism. The distinction between simple and complex refers to the number of sources of exposure required for activation, not the number of exposures. A contagion is complex if its transmission requires an agent to have contact with two or more sources of activation. In the complex contagion model it may take multiple exposures to

pass on a contagion whose probability of transmission in a given contact is less than one. As in the Bass-ABM, there is a decision for each agent in a discrete-time model where an agent adopts (if she has not adopted yet) when the number of neighbors who became aware in previous time steps is greater or equal to the threshold parameter of the model ϕ. More formally, for an agent i, when the number of her/his friends that has adopted premium content, a_i is, at least, equals to threshold ϕ, the probability for i to adopt is 1. And when this number of adopted friends does not achieve ϕ, the probability jumps to zero:

$$Pr[adopt] = \begin{cases} 1 & \text{if } a_i \geq \phi, \\ 0 & \text{if } a_i < \phi. \end{cases} \quad (1)$$

- Complex contagion with external influence coefficient: this is an extension of the latter adoption model. In this case, the agent can also innovate with a probability $\hat{p}$ because of network externalities as in the Bass-ABM. Therefore, at each time step, a basic agent i becomes premium due to the external influence (given by probability $\hat{p}$) or complex contagion (given by the probability of Equation 1).

Model Validation

Validation is the process of determining how well a model replicates reality. We follow the four major steps suggested by Rand and Rust (2011) to examine whether our model is rigorously valid for our set of questions:

- **Micro-face validation**: We employ the diffusion of innovation or information paradigm which has been validated by numerous studies (Libai et al. 2013, Rand and Rust 2011, Rogers 2003) to model our micro-level behavior. It relates parameters of WOM and mass media to the adoption decision of a customer. Agents make this decision on the basis of their local network. Finally, the targeting and rewarding policies are consistent with the existing literature and practices: (a) users are rewarded after adopting, and (b) basic users

are stimulated with incentives by considering their potential revenue (Haenlein and Libai 2013).

- **Macro-face validation**: The aggregated patterns of the set of complex interactions of the agents of the model seem to suggest typical innovation adoption patterns, based on the classical diffusion of innovation literature (Rogers 2003, Bass 1969, Rand and Rust 2011) and the complex contagion model (Granovetter 1973, Centola and Macy 2007). The pattern of behavior of this model at the macro-level is similar to empirical patterns observed in this previous research.

- **Empirical input validation**: Initially we used the 2012 dataset of app users of the Animal Jam company to study the app usage to determine seasonality and the features of the real social to recreate the artificial social network with the same degree distribution characteristics. Also, we follow a hold-out approach for calibrating the model by using a genetic algorithm (GA) (Goldberg and Holland 1988), a useful tool for model validation (Miller 1998, Oliva 2003). We also employ this model in this study because of its capabilities when running sensitivity analysis (Stonedahl and Wilensky 2010). To carry this out, we performed a sensitivity analysis on the input parameters of the model to explore the response of the model to a particular set of inputs. This sensitivity analysis is shown in the experimentation of the paper by data boxplots. The details of the GA for calibrating the model are given in Web Appendix D.

- **Empirical output validation**: We use real data from the company of the hedonic app of our study to validate the output of the models. For the macro-level forecast we compare against the daily adoption rates of around 40,000 active users of the app from the first of January, 2012 until the thirty-first of March, 2012. For the micro-level forecast we use data of 10,798 real users from one month (June of 2012, to be independent from the period of time used to calibrate the macro-level model). Results of the validity of the agent-based DSS can be found in the main paper and Web Appendix E.

WEB APPENDIX C: GRAPHICAL USER INTERFACE FOR THE DSS

This Web appendix shows some screenshots of a graphical user interface of the DSS for the Animal Jam problem. Additionally to this Web appendix and the Java source code and Javadoc documents, publicly available at *https://bitbucket.org/mchserrano/socialdynamicsfreemiumapps*, we have uploaded a console-based running version of the model (i.e., Java JAR file) at *https://www.openabm.org/model/5191*.

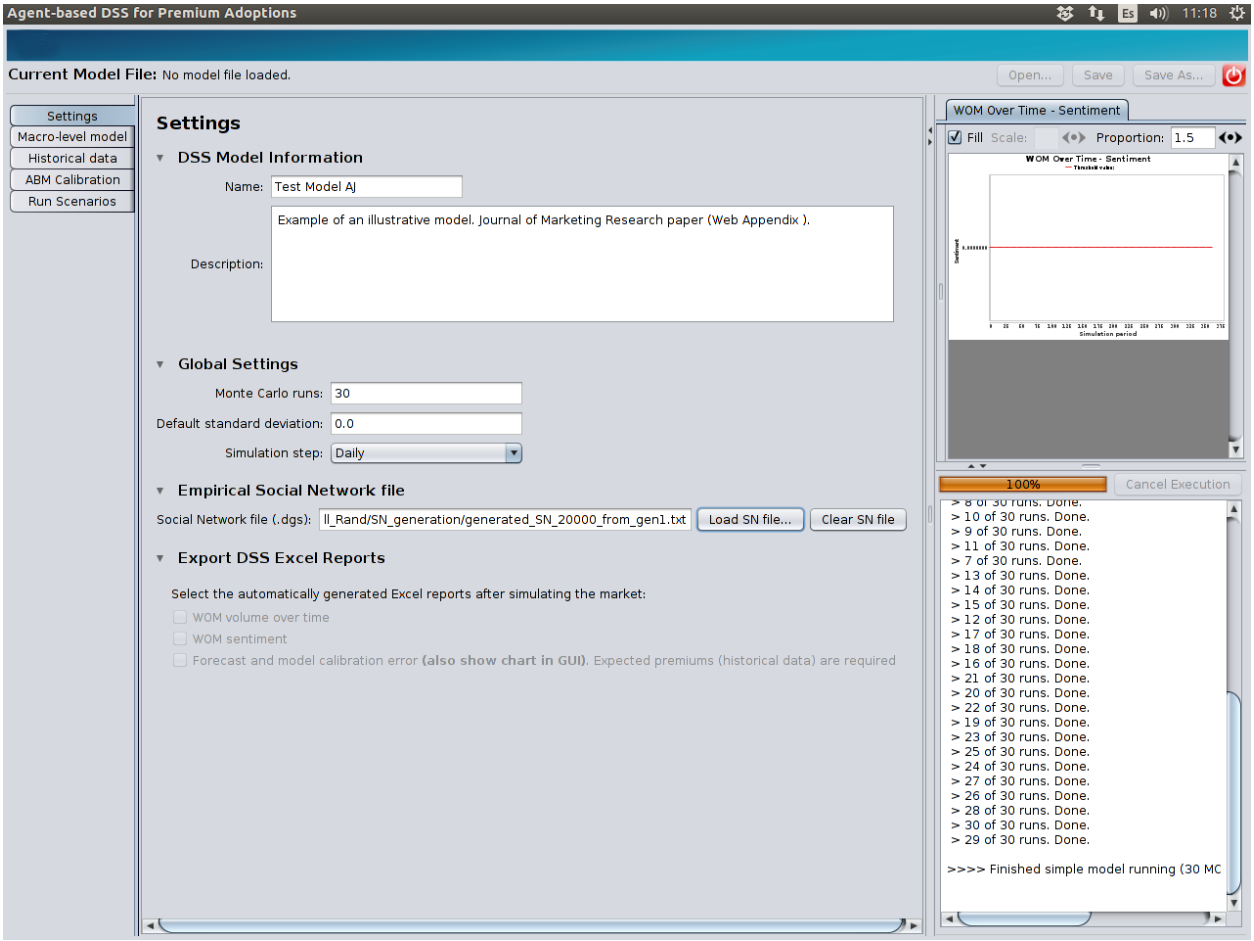

Figure 2: Screenshot of the main settings of the DSS for premium adoption.

Figure 2 shows the main graphical user interface (GUI) of the DSS for WOM programs. The modeler can save, open, and launch model files and to configure the global settings of the DSS as well as check the running progress and output charts. Additionally, exporting to a .csv file, which can be imported to Excel, is useful to independently analyze the output of the models. The

modeler can also define the main ABM parameters through the GUI of Figure 3. For instance, the granularity of the model (i.e., number of agents) can be defined which represents the ratio of real customers defined by a single agent. This screen also has the seasonality parameters for using the app, Bass-ABM, and complex contagion diffusion models.

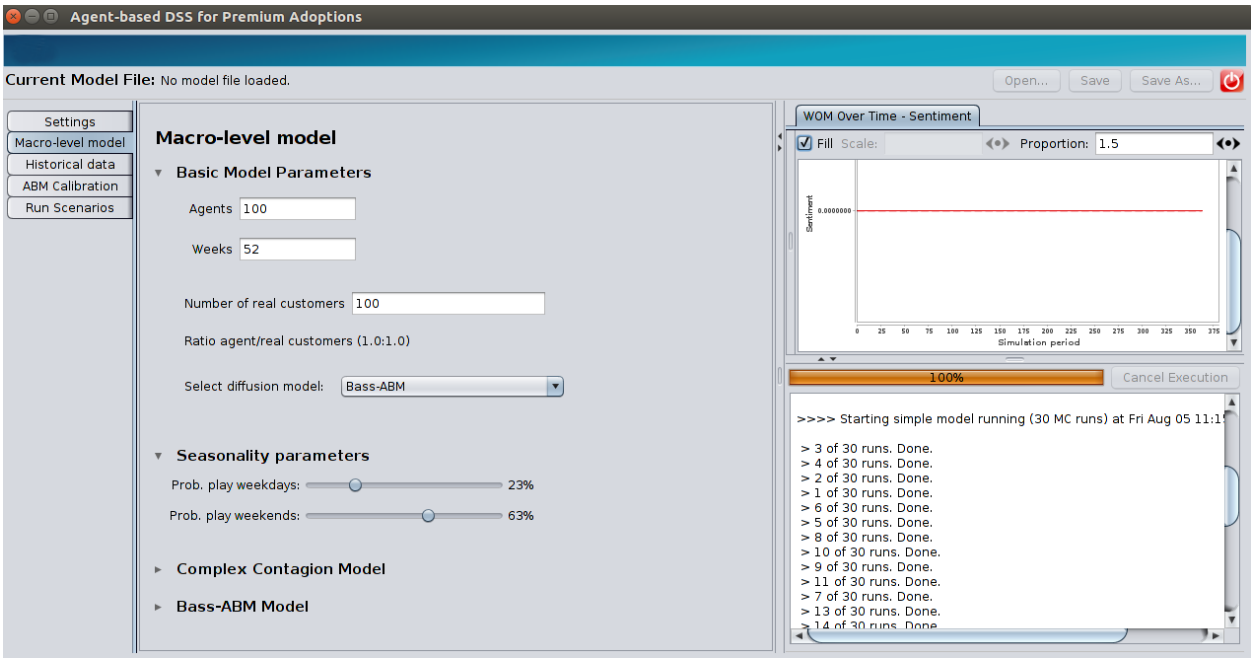

Figure 3: Definition of the models' parameters of the diffusion models and ABM macro-level simulation engine.

The calibration of the models, as defined by the guidelines and steps of the paper to build the DSS, can be performed using metaheuristics and by matching the output of the models with the historical data provided by the company. The GUI of Figure 4 guides this process. When data is loaded, the modeler can select those parameters to be calibrated as well as their minimum, maximum, and step values for the GA to search for the most convenient parameters' values.

Finally, when the modeler and marketers have validated the output results through the graphs and the output files, they can use the marketing scenarios of Figure 5, which are similar to those scenarios explored in the paper, to evaluate WOM programs for premiums expansion with the goal of increasing the overall adoption rates and revenue. This screen allows the user of the GUI to specify the type of model to run, and how to choose the set of basic users to target by rewarding policies. Also, a marketer can run sensitivity analysis over parameters such as the social influence

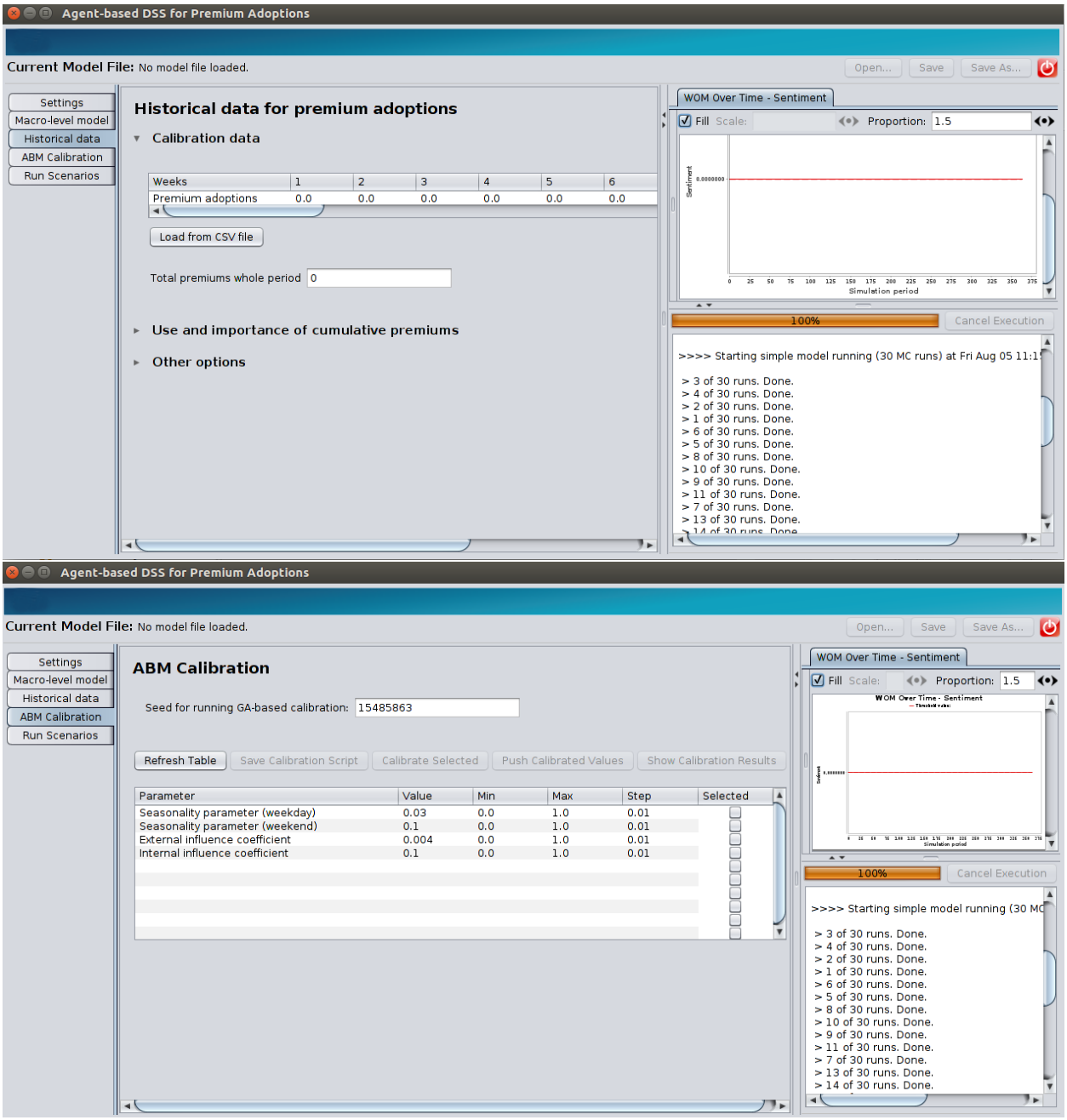

Figure 4: Two screenshots to select the historical premium adoptions of the app and the metaheuristic calibration method to calibrate the parameters of the macro-level DSS.

of the rewarded users and minimum number of premium friends to explore the effects on the *amplified* WOM and premium expansion.

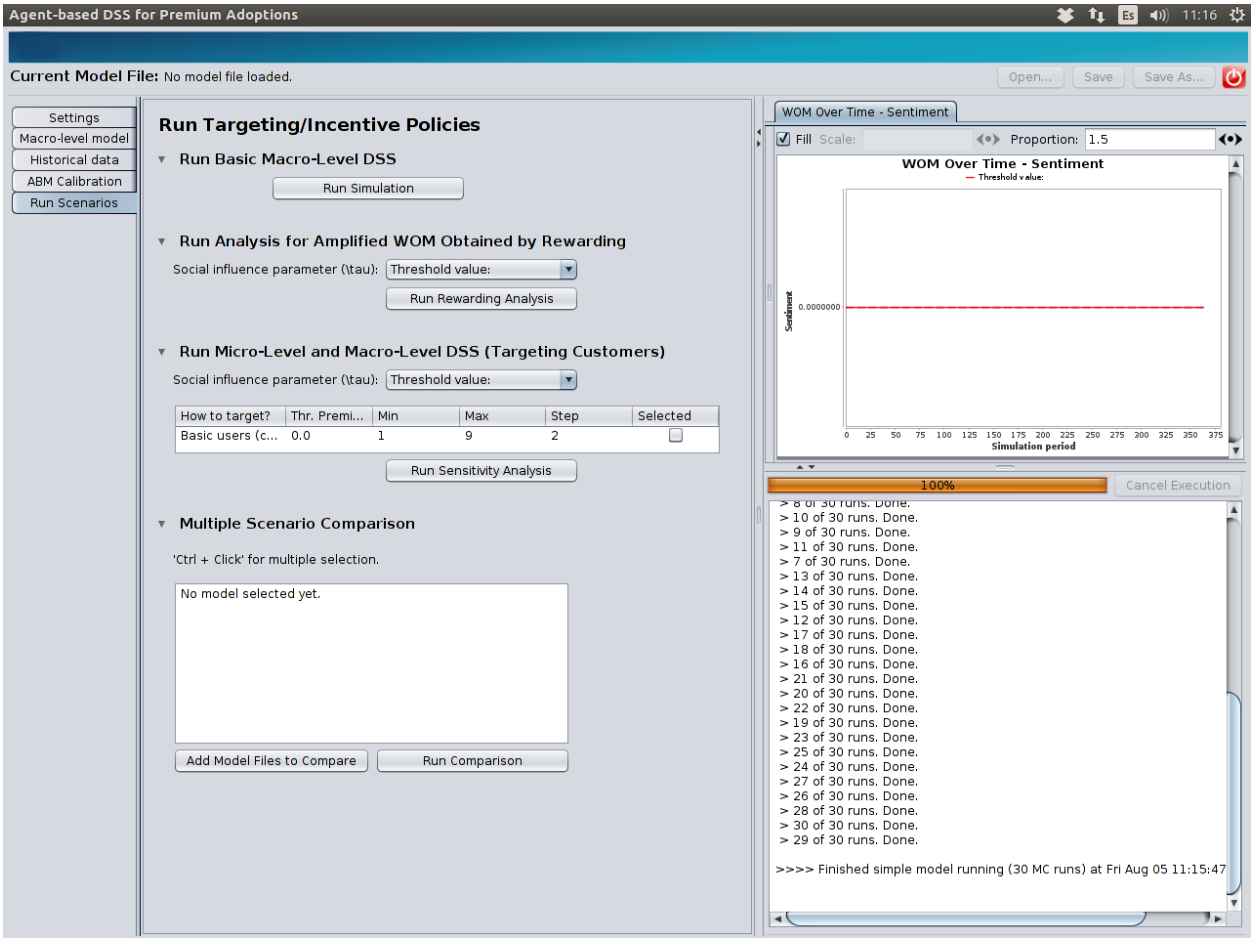

Figure 5: Main screenshot of the rewarding and incentive policies for the users of the app.

WEB APPENDIX D: DESCRIPTION OF THE GENETIC ALGORITHM FOR CALIBRATION

The metaheuristic (Talbi 2009) automated calibration method used in the paper is a standard generational genetic algorithm (GA) where every generation of the population replaces the previous one (Goldberg and Holland 1988, Back et al. 1997). The initialization process creates a population of individual solutions (chromosomes) by generating feasible values for their parameter values (genes). This process uses a uniformly distributed random procedure to select the initial value for each gene. Our implementation of the generational GA approach uses a k tournament selection mechanism (Talbi 2009). Additionally, the design of the GA has weak elitism, i.e., the best parent is always preserved through every generation.

Representation scheme

The set of calibrated optimal parameter values P^* for any particular instantiation depends on the specific adoption model. Parameter values can also have different characteristics (integer and real) and ranges. Hence, the coding scheme of the GA has a representation that allows the calibration of integer and real parameters depending on the diffusion model:

- Bass-ABM (four real parameters): two seasonality parameters (real-valued), $\gamma_0, \gamma_1 \in [0, 1]$, and two parameters for the adoption model, a external influence coefficient $\hat{p} \in [0, 1]$, and internal influence coefficient, $\hat{q} \in [0, 1]$.

- Complex contagion (three parameters): two seasonality parameters (real-valued), $\gamma_0, \gamma_1 \in [0, 1]$, and a bounded integer threshold adoption parameter $\phi \in [0, m]$ where m is the maximum number of neighbors of across all users of the social network.

- Complex contagion with external influence coefficient (four parameters): two seasonality parameters (real-valued), $\gamma_0, \gamma_1 \in [0, 1]$, a bounded integer threshold adoption parameter $\phi \in [0, m]$ where m is the maximum number of neighbors of across all users of the social network, and a fourth parameter which is equivalent to the external influence coefficient $\hat{p} \in [0, 1]$.

Crossover operator

The crossover operator plays a central role in GAs. In fact, it may be considered to be one of the algorithm's defining characteristics, and it has a dramatic effect on the GA's behavior. The crossover operator provides for the recombination of information that is drawn from individual solutions. It combines the features of two parent solutions to form two offspring solutions. It is applied to solutions in the population with probability, p_c per individual solution.

Our crossover operator is based on a classical two-points crossover: the PMX. PMX generates two offspring from two parents by means of the following procedure: (a) selection of two random cut points, (b) for the first offspring, copy the parameter values (genes) outside the random points directly from the first parent, and c) copy the parameter values (genes) inside the two cut points but in the order they appear in the second parent. The algorithm follows the same mechanism with the second offspring but with transposes the roles of the parents.

Mutation operator

The mutation operator arbitrarily alters one or more parameter values of a selected solution (chromosome) to increase the structural variability of the population. The role of mutation in GAs is to explore parameter values (genetic material) that is not currently being explored in the population. Mutation can also "re-discover" parameter values that have been lost from the population. Mutation is important in the GA because it prevents the premature convergence of GA to sub-optimal solutions by constantly perturbing the population. Moreover, it insures that the probability of reaching any point in the search space is never zero. Each parameter value (gene) of every individual solution (chromosome) in the population undergoes a random change according to a probability defined by the mutation probability, p_m.

In our GA design, when mutation occurs a parameter value (gene) of the individual solution (chromosome) is mutated by generating a new parameter value (gene value) c which is a uniform random number from the range $[c_{min}, c_{max}]$ depending on the valid range of parameter values.

Fitness function

The fitness function of the GA evaluates how good a solution is at solving a specified problem. In our case, this solution quality is given by examining the model fit to the historical training data of new premiums.

We take the whole dataset of premium adoptions, R, and divide it into R_{train} and R_{test} to follow a hold-out validation approach as done by Stonedahl and Rand (2014). Both datasets have their corresponding environmental variables E_{train} and E_{test}, which specify the conditions of the surrounding environmental situations. In our case, the only difference between E_{train} and E_{test} is the number of premium adoptions. Other elements such as the social network remain constant. The GA calibration is then accomplished by identifying the set of parameters P^* such that an error measure $\epsilon(R_{train}, M(P^*, E_{train}))$ is minimized. Any number of error measures, ϵ, such as, Euclidean distance and Pearson correlation coefficient can be used. Due to the stochastic nature of the model, one run is insufficient to evaluate each set of parameters (solution), and so the GA must run a MC simulation to obtain a robust error measure of the model. We utilized a MC simulation of 15 runs. Therefore, the error estimation $\epsilon(R_{train}, M(P^*, E_{train}))$ is the average of the fitting errors in these 15 model simulations.

In our case and after a preliminary experimentation with different measures we adopted the the L^2 or Euclidean distance for the ϵ error, computed by:

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

where n is the number of outputs (days of the historical evolution) and V_M and V_R are the output vectors (new premiums) of the model and the historical data, respectively. Also note that minimizing the L^2 norm is equivalent to minimizing either the mean square error (MSE) or root square error (RMSE).

Parameters of the GA

Table 1 summarizes the parameters for running the GA to calibrate the models.

Parameter	**Value**
GA parameters	
Number of GA runs (with different seeds)	15
Population size	100
Number of evaluations (stopping criteria)	$20,000$
k for the tournament selection scheme	3
Crossover probability (p_c)	1
Mutation probability (p_m)	0.1
ABM specific parameters	
MC simulation runs	15
Number of agents (n)	$20,000$
Rate of initial number of premiums (α)	0.0406
Weighted social influence (τ)	1

Table 1: Parameter values for the GA calibration method.

WEB APPENDIX E: ADDITIONAL RESULTS OF THE AGENT-BASED DSS FOR THE FREEMIUM APPLICATION

DSS calibration results of the macro-level adoption forecast

This section contains additional results about the validity and performance of the three diffusion models of the agent-based DSS when forecasting macro-level premium adoption with respect to the main paper. Plots of Figures 6 and 7 show the evolution of the output of the agent-based Bass model (Bass-ABM) and the historical app data for the training and test datasets. As can be seen, the model fits the data well for both training and testing datasets.

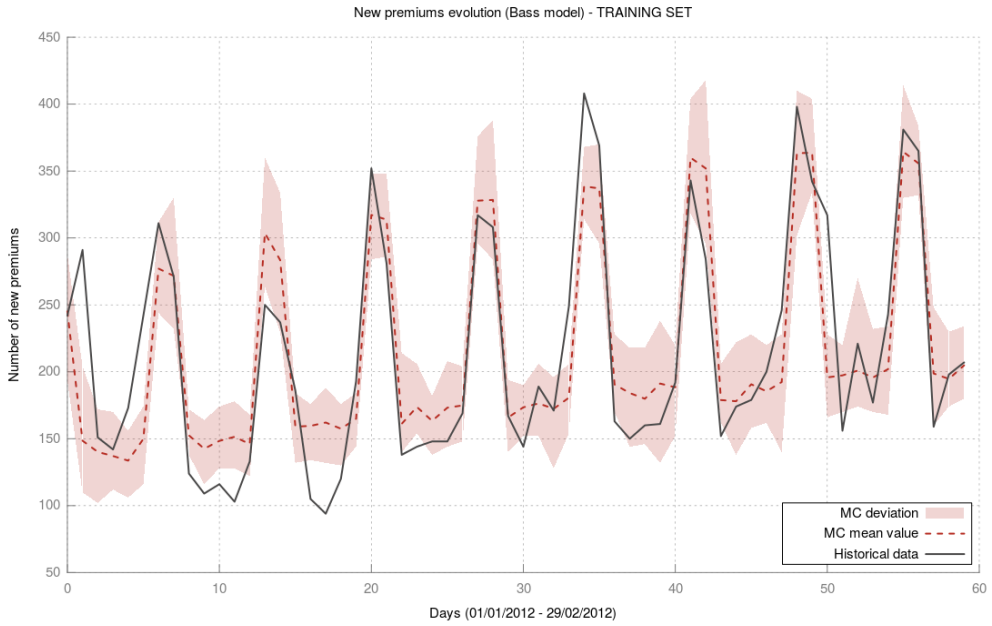

Figure 6: Agent-based DSS output when using the Bass-ABM diffusion model with respect to the historical training dataset (60 days). The Euclidean deviation mean values and error bars for the outputs of the 15 MC runs of the DSS are plotted by the solid line and the gray area, respectively.

Plot of Figure 8 shows the cumulative premium adoptions for test dataset where we see the cumulative adoptions of premium content for the three diffusion models. The Bass-ABM fits better than both complex contagion models at the end of the test period. Therefore, the Bass-ABM not only fits the daily patterns slightly better than complex contagion but also the final number of premium adopters when simulation ends. Differences between both complex models are low. This

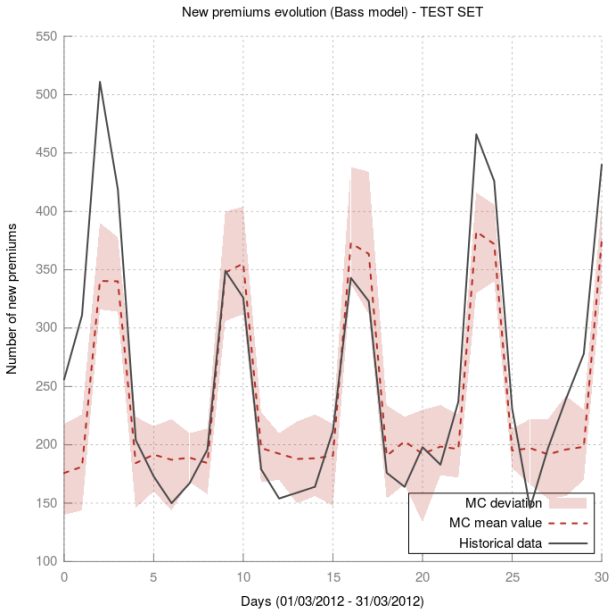

Figure 7: Agent-based DSS output when using the Bass-ABM diffusion model with respect to the historical test dataset (31 days). The Euclidean deviation mean values and error bars for the outputs of the 15 MC runs of the DSS are plotted by the solid line and the gray area, respectively.

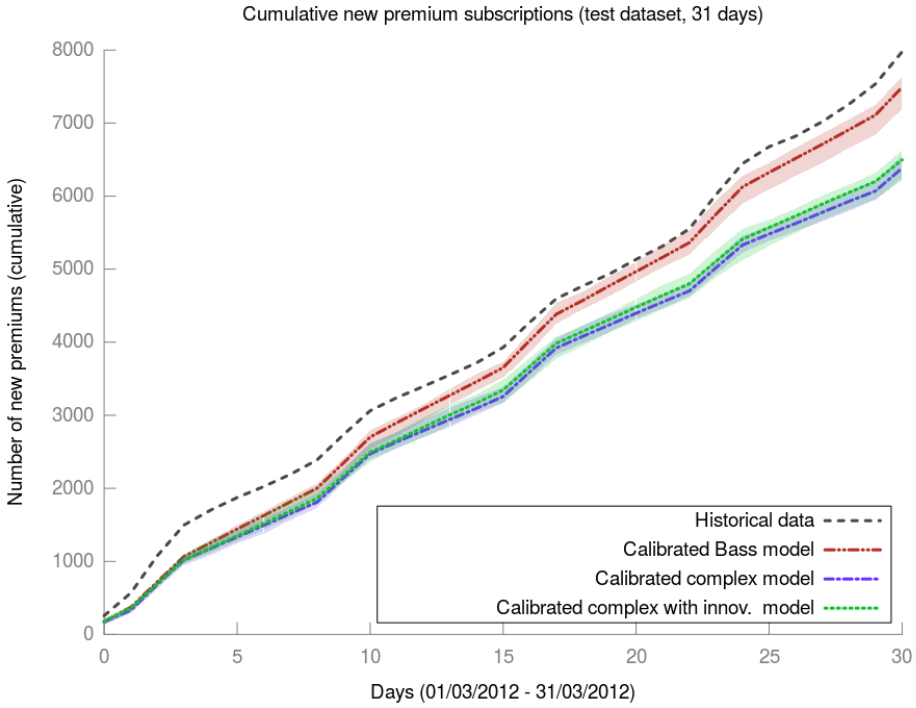

Figure 8: Cumulative premium adoptions predicted by the three implemented agent-based DSS with respect to the cumulative historical test dataset (31 days). The Euclidean deviation mean values and error bars for the outputs of the 15 MC runs of the DSS are plotted by the solid line and the gray area, respectively.

is also in line with the analysis of the main paper regarding the scarce impact of introducing the external influence coefficient to the traditional complex contagion of Centola and Macy (2007).

Sensitivity analysis of the models' parameters

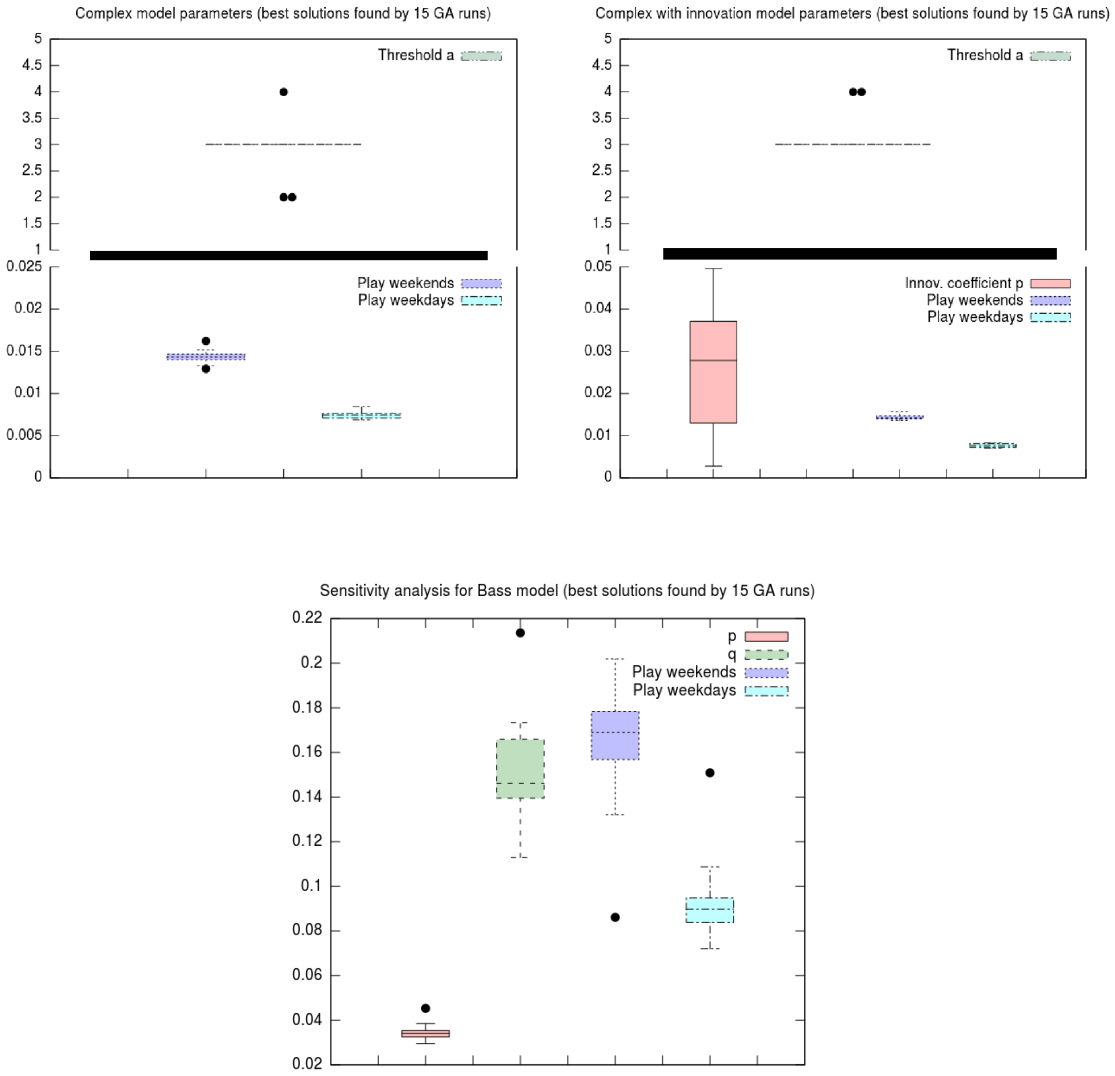

Figure 9: Boxplots showing all the values of the parameters,obtained from the best GA calibrated solutions, for the three diffusion models.

Boxplots of Figure 9 represent the parameter values of the best solutions found for the three models averaged over the 15 runs of the GA for each of the models. These best solutions used to plot the figures have similar error values for all three models, with the Euclidean distances between 354.28 and 361.57 for the Bass-ABM; and between 440.08 and 459.28 for the complex

contagion models.

Application of the targeting and incentive policies

Figure 10 shows three heat-maps of the results after applying the targeting and incentive policies discussed in the main text of the paper and running the DSS simulations. The values of the heatmaps are the additional premium adopters with respect to the baseline simulation, i.e., no incentive policy. Note that these values are the average of a number of MC simulations of the macro-level simulation forecast.

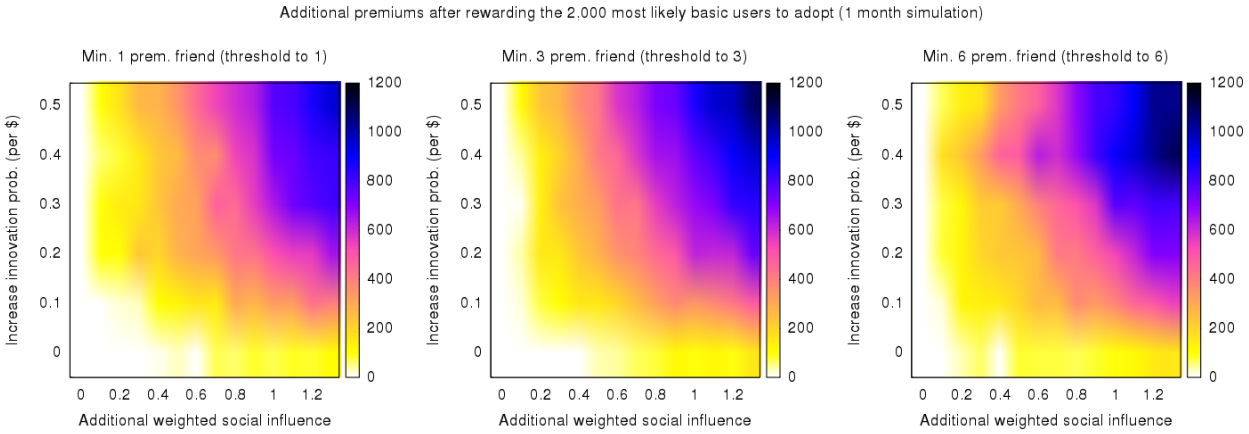

Figure 10: Additional premiums obtained after one month by targeting and rewarding three different groups of basic users (i.e., using the complex contagion threshold for 1, 3, and 6 premium friends) at the beginning of the simulation.