Web Appendices for Decomposing the Value of Word-of-Mouth Seeding Programs: Acceleration vs. Expansion

Web Appendix A: On agent-based modeling, verification, and validation

Agent-Based Models (ABMs) are especially suitable for cases in which a larger number of agents interact in a way that while simple to model on the individual level, are too complex to track using simple aggregate approaches (See Rust and Rand 2011 for a recent comprehensive review). Hence they fit well contagion processes among customers. Essentially, researchers build an individual- or network-level model of behavior, and use a simulation to examine how individual-level behavior aggregates to market-level aggregate consequences. Next we describe the basic features of our agent-based model, and how it corresponds to guidelines suggested recently by Rand and Rust (2011).

Basic features of the model

Basically one can conceptualize an agent-based model as a collection of connected cells, each representing a customer. In the agent-based environment, time is discrete and at each point in time, a cell can be in a finite number of possible states. A cell can change its state each period as a function of the state of the cells to which it is connected. In cellular automata application of ABM, the social system is a matrix with each cell connected to the cells immediately in its vicinity (i.e., typically the 8 cells that surround it). However, as we do here, to make the processes more realistic, researchers increasingly use more realistic structures of social systems as the base for the connectivity among individuals. The algorithm by which cells change their state is usually called “local rule” or “transition rule”. The collection of all states at a given point in time is called the “global state”. In each period, application of the local rule to a cell changes the global state of the grid. In the simple version, local rules are deterministic: A global state determines exactly the next global state. However, one can also use a stochastic model (as we do here), in which the state of the cells changes based on some probability function, which is a function of the state of the surrounding cells.

Consider the diffusion of innovations we discuss here. Two basic approaches have dominated the decision of how to model the transition from non-adopter to adopter. Since Granovetter’s (1978) seminal work, in disciplines such as sociology and communication, threshold models have been

largely used to model a variety of phenomena, including the basic diffusion process. In these cases, the assumption is that an individual adopts an innovation only when a certain number of others, sufficient in number so as to surpass her threshold, have already done so. In contrast, *cascade model*s, or *competing-risk models* such as the one used here (Leskovec, Adamic, and Huberman, 2007) take a stochastic approach that follows the basic diffusion-of-innovations Bass model and its extensions. Under this approach, in each period, a customer has a certain probability of adopting based on communications with her previously adopting peers or in response to marketing efforts. Here, we use a cascade approach as it offers a number of advantages: First, it incorporates external effects such as advertising that are not traditionally part of the threshold adoption approach. Second, it allows a more realistic stochastic approach, while the adoption threshold is deterministic. Third, it follows a well-established research tradition in marketing, which also allows us to build on past research when setting up and calibrating model parameters.

Our model is thus similar in essence to previous cascade-based agent-based models of the diffusion of innovations published recently in the marketing literature (Goldenberg et al. 2007; Goldenberg, Libai, and Muller 2010), with a notable difference: It is a competitive, brand-level model, and not a category-level model. Each potential adopter can be influenced to adopt either Brand A or Brand B, which depends on the number of adopters of each in her social network. In the body of the paper, the local rules are described in detail. Next we give an overview of dynamics of the basic process, where Brand A has a program and Brand B does not (without extensions such as customer defection of shorter time horizons). The process is independently run on each of the 12 network structures described in the article. While connectivity is assumed bi-directional, the level of influence is not necessarily symmetric.

Period 0: This is the initial condition; generally, consumers have not yet adopted any of the products (that is, activity of all network members is 0). Only the seeded group of customers with Brand A receives the value of A.

Period 1: The probabilities for each consumer *prob*(*t*) are realized. For Brand A, an individual member can obtain the activation value of A through the combination of external influence and internal influence from the seeded people, if s/he is connected to any of them. For Brand B only, external influence is still in place. A random number U is drawn from a uniform distribution in

the range [0,1]. If U< *prob*(*t*) (based on the algorithm described in the paper), then the consumer moves from non-adopter to adopter (receiving the value of A for Brand A, or B for Brand B). Otherwise the consumer remains a non-adopter.

Period n: The process continues as gradually more non-adopters obtain the value of either A or B, which changes the probability for the remaining non-adopters in the next period.

Period 30: The process ends. By that time, on average, more than 90% have become adopters of one of the brands. The program calculates the customer equity, i.e., the net present value of each brand, given the number of adopters and when each one adopted.

Guidelines for rigorous agent-based models

In a recent article, Rand and Rust (2011) proposed structured guidelines for rigorous agent-based modeling focusing on models especially relevant for marketers, such as those dealing with the diffusion of innovations. Specifically they emphasized two factors that should be considered: *verification* and *validation*. We next briefly describe how the approach presented here is consistent with their guidelines on these two matters.

Verification

As Rand and Rust (2011) note, verification deals with the match of the implemented model to the conceptual model. They suggest three verification steps: documentation, programmatic testing, and test cases. We followed these stages as described below:

Documentation: For each new module we add to the simulation, we start with a detailed written description of the theoretical conceptual model. Based on it, we generate a specifications document for programming. While programming, we closely follow the specifications document to verify that all functionality is covered, and there is no redundant functionality. We have a detailed work log in which we update what we are working on, and what functionality it serves. The code is documented so that even untrained assistants can easily find their way in it. The logical flow of each procedure and method is described, and classes are documented and explained.

Programming testing: Done through a series of sanity checks, the Visual Studio debugger, test prints, and scripts as follows:

Unit testing: Each method has a test method or test script associated with it, which provides a given input, and has a known given output. This output is printed to a file or to the standard output. Since many of our methods involve the Network class, which is complicated by nature, we conduct some of the tests on a simplified, toy network of 10 members, with homogenous ties and influence parameters, and a given, predefined realization matrix.

Code walkthroughs: Are done through scripts, each focusing on a part of the process (e.g., network building, activation, attrition, NP calculation), and the appropriate logical flow is run using all the relevant classes and methods.

Debugging walkthroughs – Are done using the same scripts, using the Visual Studio debugger. Note, that in projects where the researchers are not the programmers, code and debugging walkthroughs are very different in nature. In our case, they tend to overlap, since we are the ones who write the code.

Formal testing – Is done on the toy network using Excel and pen and paper.

Test cases: We invest a lot in building and running test cases, whenever a change is applied to the code, or a new module is added. *Corner cases* are used as sanity checks for extreme values (no connectivity, full connectivity, realization is all 0, realization is all 1, full attrition, zero attrition, etc.) *Sample cases* are used to test that we obtain a reasonable range of outputs for our parameters. Our standard sample case is the Keller-Fay network, 30 periods, and 0.1 discount rate, no attrition, and brands are similar in p and q. We already know what values to expect from this network, and use it as a basis for many of our tests. We also have a set of *specific scenarios* (uniform degree of 6, zero discount, zero periods, etc.), and sets of *relative value testing* that we used for network size, number of periods, attrition, and time horizon. For example, when adding the attrition functionality to the code, we ran test scenarios with differing attrition rates to verify that higher attrition indeed results in fewer customers).

Validation

In terms of micro validation, similar to the example central in Rand and Rust's (2011) paper, our model is a contagion model that follows the diffusion-of-innovation paradigm, which has been validated by numerous studies over the years. This relates to the parameters of word of mouth and mass media, as well as to the fact that agents possess a local social network that does not

allow them to discover information about the entire population. In addition, the seeding process is consistent with practice both in its nature (getting the product to consumers randomly or to opinion leaders with launch) and its extent, as a percentage of the population. At the macro level, the aggregated patterns of diffusion created by our model appear to suggest typical innovation adoption patterns of a sigmoid curve in which at first a few adopt, and then more and more until the entire population has adopted.

At the empirical input level, the range of parameters for p and q is consistent with previous cascade modeling research, which draws on the Bass model. While the value of the mass media parameter is in the range of the aggregate Bass model, there is a need to adapt the q parameter to the size of the network. This adaptation has been done in the past (e.g., Goldenberg, Libai, and Muller 2002) where the parameters are set to achieve curves similar in magnitude to the empirical observations of the Bass model research. Recent research has indeed pointed to the close relationship between the individual-level cascade models and the aggregate, empirically supported, Bass model diffusion approaches (Gibori and Fibich 2010).

In addition, we used a variety of real-world networks as the base for the social system structure in which the contagion process occurs. This social network structure is more realistic than models in which every consumer knows every other consumer. Because we examine a range of social network structures, we are able to cover differing social system scenarios.

In terms of empirical output, thus far there have been no published empirical studies that enable differentiation between acceleration and expansion in word-of-mouth programs. Thus, our ability to compare our results to previous findings is limited. However a number of stylized facts indicate consistency between outcomes of our model and industry practice. Our results support the emphasis placed by many programs on market expansion, and are consistent with the common practice of focusing on opinion leaders, which is common in word-of-mouth marketing. It can also be shown using our framework that in terms of seeding, going much above the widely used industry benchmark of 1% results in decreased profitability.

Additional References

Fibich, Gadi and Ro'i Gibori (2010), "Aggregate diffusion dynamics in agent-based models with a spatial structure," *Operations Research*, 59(5), 1450-1468.

Goldenberg, Jacob , Barak Libai, Sarit Moldovan and Eitan Muller (2007), "The NPV of Bad News," *International Journal of Research in Marketing*, 24, pp.186-200.

Granovetter, Mark S. (1978), "The strength of weak ties," *American Journal of Sociology*, 78(6), 1360-1380.

Leskovec, Jure, Lada A. Adamic, and Bernardo A. Huberman (2007), "The dynamics of viral marketing," *ACM Transactions on the Web*, 1(1), 1-39.

Web Appendix B: Detailed network descriptions

We use three types of data sources for the network structure, as follows:
The published social network to which we had access has been used in recent years for various studies of social networks. E-mail Network URV is the e-mail network that manages communications between faculty and graduate students at Spain's Universitat Rovira i Virgili (URV). URV eliminates bulk e-mails and considers a connection between A and B only if A sent mail and B replied (Guimera et al. 2003). PGP is the giant component of the network of users of the Pretty-Good-Privacy algorithm for secure information interchange (Boguña, Pastor-Satorras, and Diaz-Guilera 2004). Cameroon Tontines comm3 is a social network of women in Cameroon who were asked about their social communications as a part of a study on the use of contraceptives (Valente et al. 2007).

A second type of network is one for which we collected data for this study. Recently there is growing interest in the academic literature in online communities and their effects on customer loyalty, new product growth, and profitability in general. Here we had the cooperation of Lithium Technologies, a leading online community platform provider that manages online communities for various US brands. Customers who join the community exchange information on a host of issues including support, ideas for new products, and discussions about the brand / product. For our purposes, each discussion between individuals on a specific matter constitutes a "link". Assuming that the structure of social networks evolving may differ among subjects, we requested network information for various categories. We received data on information exchange in six such networks in areas including technological products, entertainment products, consumer products, retailing, and services (categories set by Lithium). The social networks presented here are those whose members surpassed a minimum level of involvement with the community, as defined by Lithium.

Another social network we used is YouTube, which enables those who upload a video to be part of a social network connecting with other members. The network presented was created using a "snowball technique": We first collected data on the users who uploaded the 25 most viewed videos in June 2009. We expanded our sample network by adding each YouTube user who was linked in a "friendship" connection to a member of the network, and had at least two additional

“friendship” connections with other users within the general YouTube social network. The final data set contained over 4,000 users.

The third type of network is the *empirical degree distribution*, where “degree” is the number of others connected to a given member of the work. Here we do not have the full network connections, but constructed a randomly assigned social network based on a reported degree distribution. The first one is a distribution based on *TalkTrack* by the Keller-Fay group, an award-winning, ongoing survey of American consumers ages 13-69, who report on WOM activity as well as social network size. The averages reported here are based on interviews with over 50,000 TalkTrackers who report size of close personal networks. A second empirical degree distribution is based on the reported averages of over 11,000 customers who visited the CNET site (Smith et al. 2007). Here the degree distribution is based on self-reporting on the number of others with whom respondents communicate at least once a month both online and offline.

Web Appendix C: Additional tables

Table C1. Parameter ranges

Parameter	Range
δ – external influence	0.001, 0.005, 0.01, 0.015, 0.02
q – normally distributed internal influence	Cameroon: Means of 0.16, 0.2, 0.24, 0.28, 0.32 (standard deviation = 0.08) CNET: Means of 0.005, 0.01, 0.02, 0.03, 0.04 (standard deviation = 0.0025) The rest: Means of 0.04, 0.08, 0.1, 0.12, 0.16 (standard deviation = 0.02)
Seeding program size - proportion of market	0.5%, 1%, 2%, 3%, 4%, 5%
Seeding program type	Random, Influential–hubs, Influential–experts

Table C2. Overestimation bias due to short time horizon

No.	Network	Social value measured after 5 periods	Long-term social value	Overestimation bias
1	URV e-mail network	138%	78%	34%
2	PGP	96%	58%	23%
3	Cameroon Tontines	163%	88%	40%
4	Retailer	129%	90%	21%
5	Services	125%	98%	14%
6	High-tech 1	123%	85%	20%
7	High-tech 2	120%	76%	25%
8	Entertainment 1	130%	76%	31%
9	Entertainment 2	124%	83%	22%
10	YouTube	140%	80%	33%
11	Keller-Fay	120%	65%	34%
12	CNET	136%	74%	36%
	Average	129%	**81%**	28%

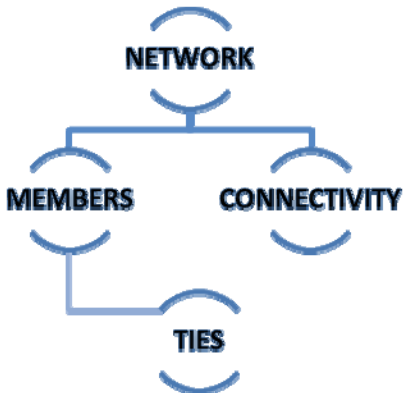
Web Appendix D: Pseudo code of the agent-based model

The simulation is an object-oriented structure comprised of over 5,000 code lines, programmed using C++. We describe below the main classes, data structures, and procedures.

Hierarchy

The main object of interest is Network, comprised of members and connectivity. Each member has a set of data associated with it (see below in the class description), plus, its systems of ties. Note that connectivity is not the overall structure of a tie; rather it is the set of methods used for tie generation, as well as statistics encapsulated from the member.

The hierarchy is described in the figure below. Note that the chart does not imply direct inheritance, but rather describes the general hierarchical structure.



Main classes:

```
CLASS Tie
ATTRIBUTES:
of_whom //the focal member's id
with_whom //the id of the member to which it is connected
influence_fromA //the influence from the focal member on the other
      member if the focal member adopted A
influence_onA //the influence on the focal member from the
      other member if the other member adopted A
influence_fromB //the influence from the focal member on the other
                      member if the focal member adopted B
influence_onB //the influence on the focal member from the
      other member if the other member adopted B
METHODS:
SetTie(id1, id2, inf_fromA, inf_onA, inf_fromB, inf_onB)
```

```
CLASS Member // A member in a network
```

ATTRIBUTES:

```
Id    // the serial number of the member
state      //did not adopt (0), adopted brand A (1) or brand B(2)
defect; //Did the member defect? Can only go with a zero
        activation state
Influential //1 influential, 0 a regular member
its_pA     //The amount of influence of brand A
its_pB     //The amount of influence of brand B
its_numties //The number of WOM ties (degree)
its_ties  // a linked list of ties
```

METHODS:

```
Activate(curr_status) //Update the results of the activation algorithm
SetDefect(defect) //Decide whether the unit will defect this period
IsConnected(member) RETURN BOOL //is the focal member connected
                                to the given member?
```

```
CLASS Network // A network is a collection of members
ATTRIBUTES:
its_members: MEMBER // Members in the network
connectivitymatx
ties_matxA            //connection strength if member adopted A
ties_matxB            //connection strength if member adopted B

METHODS:
IsConnected(memberid1,memberid2) RETURN BOOL //Is a pair
                                        of members connected?
SetTieStrength(qvalA[Networksize],qvalB[Networksize])
UpdateMembersties()
ActivateMember(new_state,isCompetition) //This is the algorithmic core
                                         (Section 4.2).
build_connections(numties[Networksize],connectivity[Networksize])
Activate(realization[Networksize],isCompetition) //Realization plus an
     indication whether this is a monopoly or competitive scenario
GetTotalActivity(activityA,activityB)
ResetActivation() //Keeping everything as was, only reset activation
                for rerun.
SetActivation(initial_act[Networksize]) //Set activation to certain
                values. It is used for setting initial activation.
PrintConnectivity()
PrintConnectivityToFile(Outfile) //Print connectivity to network
PrintActivity()
PrintActivityToFile(Outfile)     //Print activity to file
SetInfluentials(influentials_array[Networksize]) //decide who are the
                                                  influentials
GetActivity(ActivityMatx[Networksize])
CalcAveDeg() RETURN FLOAT
CalcAveDegTop10per() RETURN FLOAT
CalcClustering() RETURN FLOAT
```

```
CLASS Connectivity //Set of procedures which set the network's
connectivity

METHODS:
CreateMatrix(connectivity,tiesarray[],ties_matrix[],sizeofnetwork,
probsarray[6]) //connectivity parameter can get two possible values:
      External - Read an external connectivity, OR
      Generate_dist – Generate a random graph from a given
distribution.
      Switch(connectivity)
            Case generate_distribution:
                  CALL RandomTies
            Case external:
                  CALL ExternalTies
      End
End


RandomTies(numtiesarray[],ties_matrix[],sizeofnetwork)
      //Creates a random matrix according to a given distribution
      CALL GenerateRandomTies
      CALL SetConnectivityRandom
End


GenerateRandomTies(tiesarray[],sizeofnetwork)
      //Assigns number of ties for each member according to the Keller
      empirical distribution
      Read the distribution parameters from a file
      FOR i ← 0 to sizeofnetwork DO
            Draw a random number
            Find out between which values te number is
            Return the relevant number of ties
      END
END
```

```
SetConnectivityRandom(minmember,maxmember,numtiesarray[],ties_matrix[]
,sizeofnetwork)
      //Sets random connectivity between the members in the range
      minmember:maxmember given the number of ties for each member as
      input (numtiesarray)
      FOR each member in the range DO
            WHILE there are still ties to allocate DO
                  IF all other members are set THEN
                        Allocation is terminated
                  ELSE
                        //Since we allocate in order, all the previous
                        members are set
                        WHILE (i+1 < maxmember) DO
                              Randomly select a member to link to
                              IF selected member is free for allocation
                              and is not already linked to focal member
                              THEN
                                    BREAK
                        END
            END
      END
END



ExternalTies(tiesarray[],ties_matrix[],sizeofnetwork)
      //Reads the connectivity from an external file, generates the
      ties array and the matrix
      Reads the connectivity from an external file
      WHILE (!eof()) DO
            IF members are not already connected THEN
                  Connect the members
                  Increase the number of allocated ties
            END
      END
END
```

Main procedures:

```
void GenericExperiment(connectivity, float probsarray[6], float
Netparams[4]) //A generic experiment which compares:
     A case with no competition (pB=0 and qB=0)
     WOM program with random seed of adopters (only A or both A
     and B)
     WOM program with influential hub seed of adopters (only A
     or both A and B)
     WOM program with influential expert seed of adopters (only
     A or both A and B)


     Define Connectivity object
     CALL CreateMatrix
     Sort the ties array and find the ones with the highest
     number of connections - they will be the hubs

     Define Network object and create the network according to
     the connectivity

     READ all parameters from a file
     Assign the parameter q and mark the units with the highest
     q as experts

     WHILE (!eof()) DO
          Run the network X times with different realizations
          but same connectivity
               Run no WOM program network
                    CALL runsingleNetwork with suitable
                    parameters

               Run a network where only A has a WOM program with
               random seed of adopters
                    CALL runsingleNetwork with suitable
                    parameters

               Run a network     where only A has a hub
```

```
                influential program
                     CALL runsingleNetwork with suitable
                     parameters

                Run a network where only A has an expert
                influential program
                     CALL runsingleNetwork with suitable
                     parameters

          END
     END
END
void runsingleNetwork(Network *network, float discount, float
*npvA, float *npvB,int *finalA, int *finalB,
int seed[Networksize], bool av_slots[Networksize],
int seedsizeA, int seedsizeB, int t_entryB,
int TimeMatx[Networksize][2])
     //Run a single Network for a given number of periods.
     The function gets as input whether A or B have a program,
     and the entry time of B.
     The assumption is that A enters at time zero, and a brand
     can start a program only when it enters

     Determine the initial activation state based on the seed,
     entry times etc.

     Activate the network given the initialization vector and
     according to the entry time of the follower
     Calculate adoption time of each member
     Calculate NPV for A and for B
END


MAIN:
CALL GenericExperiment
```

Web Appendix E: Correlation matrix and results tables

Table E1. Correlation matrix

Variable	network size	average degree	clustering coefficient	discount rate	periods	brand strength	seed size	attrition rate
network size	1							
average degree	-0.304	1						
clustering coefficient	0.371	-0.093	1					
discount rate	0.005	-0.003	-0.001	1				
number of periods	0.000	-0.001	0.000	-0.001	1			
brand strength	0.011	-0.011	-0.006	-0.008	-0.001	1		
seed size	-0.031	-0.002	-0.014	0.003	0.000	0.006	1	
attrition rate	-0.016	0.010	0.006	0.009	0.001	0.027	-0.011	1

Table E2. Regression results: Influential–experts program

Independent variable	Coefficients	Standard Error
Network size	-2.2E-05	6.15E-07
Average degree	-0.00497	0.000155
Clustering coefficient	-0.69843	0.011948
Discount rate	0.013679	0.003185
Number of periods	-0.09692	0.003185
Seeding size	-3.97168	0.09235
Attrition rate	-0.1493	0.003202
Relative strength of focal brand	0.261454	0.003185
Adjusted R-Square	70.8%	

Dependent variable is acceleration ratio. All coefficients are significant at the 1% level.

Table E3. Regression results: Random program

Independent variable	Coefficients	Standard Error
Network size	-2.2E-05	7.29E-07
Average degree	-0.00458	0.00018
Clustering coefficient	-0.77353	0.013928
Discount rate	0.007571	0.003722
Number of periods	-0.10938	0.003721
Seeding size	-4.42184	0.108003
Attrition rate	-0.16614	0.003766
Relative strength of focal brand	0.28867	0.003729
Adjusted R-Square	67.5%	

Dependent variable is acceleration ratio. All coefficients are significant at the 1% level.