Investigating App Store Ranking Algorithms using a Simulation of Mobile App Ecosystems

Soo Ling Lim
Software Systems Research Centre
Bournemouth University
United Kingdom
slim@bournemouth.ac.uk

Peter J. Bentley
Department of Computer Science
University College London
United Kingdom
p.bentley@cs.ucl.ac.uk

***Abstract*—App stores are one of the most popular ways of providing content to mobile device users today. But with thousands of competing apps and thousands new each day, the problem of presenting the developers' apps to users becomes non-trivial. There may be an app for everything, but if the user cannot find the app they desire, then the app store has failed. This paper investigates app store content organisation using AppEco, an Artificial Life model of mobile app ecosystems. In AppEco, developer agents build and upload apps to the app store; user agents browse the store and download the apps. This paper uses AppEco to investigate how best to organise the Top Apps Chart and New Apps Chart in Apple's iOS App Store. We study the effects of different app ranking algorithms for the Top Apps Chart and the frequency of updates of the New Apps Chart on the download-to-browse ratio. Results show that the effectiveness of the shop front is highly dependent on the speed at which content is updated. A slowly updated New Apps Chart will impact the effectiveness of the Top Apps Chart. A Top Apps Chart that measures success by including too much historical data will also detrimentally affect app downloads.**

Keywords-mobile app ecosystems; Artificial Life; agent-based simulation; app store; top apps chart; new apps chart; evolving developer strategies

I. INTRODUCTION

Mobile applications are big business. From the pioneering iOS App Store created by Apple, to the App World of BlackBerry and the Android Market, today hundreds of apps are downloaded every second. The revenue generated from app sales surpassed $15 billion in 2011 and is estimated to reach $58 billion by 2014 [1].

Mobile app ecosystems, comprising developers, users, and apps, face challenges that are brand new to the software industry. App store owners face the challenges of presenting the rapidly increasing app store content to the users and encouraging users to download apps. Developers find it increasingly difficult to make their apps stand out among hundreds of thousands of other apps in the app store, achieve downloads, and make profit. App users have difficulty in finding good apps amongst the vast number of alternatives.

It is difficult if not impossible to create different app stores, or modify existing app stores, and experiment with millions of real users. Consequently for this work we use AppEco, an Artificial Life (Alife) agent-based model, as an experimental tool to address such challenges [2, 3]. Alife methods have proven their worth with many previous simulations of ecosystems. AppEco is a model of app ecosystems. It models developers (agents that build apps) and users (agents that download apps). It simulates the app store environment, which hosts and organises content created by the developers, and enables users to browse and download apps. Significantly, AppEco also models apps – artefacts produced by the developers and downloaded by users – and their features. AppEco allows us to conduct experiments, test hypothesis about various processes in the ecosystem, and ask "what if" questions, all of which are otherwise difficult if not impossible to conduct in a real-world setting.

In this paper we focus on the app store and how it should best provide content to users in order to encourage app downloads, which we quantify in terms of a *download-to-browse ratio*. We use AppEco to simulate Apple's iOS app ecosystem and investigate content organisation in the iOS App Store, specifically, how the Top Apps Chart and New Apps Chart should best be organised. We study the effects of different app ranking algorithms for the Top Apps Chart and the frequency of updates of the New Apps Chart on the download-to-browse ratio over time.

The rest of the paper is organised as follows. Section II describes existing work. Section III describes AppEco. Section IV describes the application of AppEco to simulate the iOS app ecosystem, the experiments and results. Section V provides our conclusions and discusses future work.

II. BACKGROUND

While the study of mobile app ecosystems is a current and significant topic for researchers, to date there has been little work focussing on the topic [4]. However there is much related work that contextualises and informs the current study.

One area related to app ecosystems is the study and prediction of app downloads and usage. For example, Garg and Telang developed strategies to infer the number of downloads for an app based on its ranking on Apple's iOS App Store Top Apps Chart [5]. Such work may enable investors to estimate likely profits should an app reach a specific rank, however there is no certainty that a new app will appear on the chart. Bohmer et al. developed a mobile app to collect mobile app usage information from over 4,100 users of Android devices [6]. Their research revealed interesting app usage behaviours among the users. For example, although users spend almost an

hour a day using their phones, an average session with an app lasts less than a minute. They also found that news applications are most popular in the morning and games are at night, but communication applications dominate through most of the day [6]. These studies are informative, but they are limited to studying what is already out there, and "what-if" questions cannot be answered.

In the fields of Alife, Evolutionary Computing and Agent-Based Simulation, there are a growing number of studies on the emergent effects of human interaction at the population level. For example, Kohler et al. used models to understand the environmental and social factors that led to the disappearance of the Puebloan peoples of the North American Southwest [7]. Wilkinson et al. used models to understand the urbanisation process in ancient Mesopotamia [8]. Huang et al. used models to study the spread of epidemic diseases over complex social networks [9]. Bosse and Gerritsen studied the interplay between the emergence and displacement of criminal hot spots and the reputation of the involved locations [10]. Lux and Marchesi showed that the scaling of financial prices arises from interactions between a large number of market participants [11]. App stores have large populations of apps, developers, and users, and can benefit from similar studies.

Alife researchers have also modelled various aspects of ecosystems such as evolutionary dynamics within interacting populations. For example, Holland created Echo, a generic ecosystem model in which evolving agents are situated in a resource-limited environment [12]. Olson and Sequeira developed an environment for producing and running artificial ecosystems [13]. Pachepsky et al. investigated the effect of ecological interactions between organisms on the evolutionary dynamics of a community [14]. Agent-based models are also widely used to study natural and man-made ecosystems. For example, Antona et al. modelled the economic exchanges between consumers and harvesters of renewable resources [15]. Alexandrova-Kabadjova et al. developed an agent-based model to study an artificial payment card market [16].

Indirect interaction through mechanisms such as stigmergy is commonly studied by Alife researchers. But in human society we use more complex ways to interact. Different kinds of objects and tools are often built, adopted, shared, and used to support people in their work. These entities, also known as artefacts, have a key role in determining the success or failure of human activities [17]. It is common for such artefacts to become media for communication (e.g., books, music, and software). One study relating to this topic is the use of robots to create music. In this work, Miranda developed a group of interactive autonomous singing robots that interact and imitate each other to create music [18]. Despite such studies, which often focus on the evolution of human culture with reference to artefacts [19], there is a lack of models that study the development and consumption of artefacts by agents, and how the success of those artefacts depends on the preferences of the agents.

III. APPECO

AppEco is an Artificial Life simulation of mobile app ecosystems developed in previous work [2, 20]. In the previous work, AppEco was used to study the effects of different developer strategies [20], and different publicity strategies [2]. This paper focuses on the app store itself and investigates for the first time different store ranking algorithms and their effects on app downloads. In this section, we describe elements of AppEco to set the context and provide the background for our study. Earlier versions of AppEco are described in [2, 20].

In a mobile app ecosystem, coevolving systems of apps, developers, and users form complex relationships, filling niches, competing and cooperating, similar to species in a biological ecosystem [4]. The health of the app ecosystem is largely determined by the communities of developers that create innovative solutions that users want to buy [21].

The AppEco model consists of agents that are abstractions of app users and developers, as well as artefacts that are abstractions of apps. Developer agents build and upload apps to the app store; user agents browse the store and download the apps, see Figure 1. A distinguishing feature of the AppEco model compared to more traditional agent-based models is the explicit modelling of artefacts as well as the agents that produce and use the artefacts. Different from agents, artefacts are not autonomous, they represent passive entities of the system that are intentionally created and used by agents. App artefacts are important in a model of an app ecosystem because the agents interact with one another via the apps.

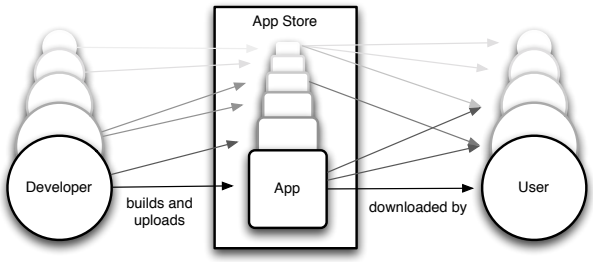

Figure 1. The interaction between developers, apps, and users in AppEco.

A. AppEco Components

AppEco consists of app developers, apps, users, and the app store. Each component is described as follows.

Developers. In AppEco, a developer agent represents a solo developer or a team of developers working together to produce an app. Each developer agent has a development duration (*devDuration*, a random value between [dev_{min}, dev_{max}]), which specifies the number of days it needs to build an app. Each developer also records the number of days it has already spent building the app (*daysTaken*). Each developer is initially active (it continuously builds and upload apps to the app store) but may become inactive (it stops building apps) with probability $P_{Inactive}$. This models part-time developers, hobbyists, and the tendency of developers to stop building apps[1]. Every developer records the number of apps it has developed and the number of downloads it has received.

In this work every developer uses an evolutionary app creation strategy of making a variation of its own best app (app with highest number of downloads) each time [20]. This

[1] http://t-machine.org/index.php/2009/06/11/may-2009-survey-of-iphone-developers/

models the ability of developers to learn from downloads and improve on their best app. This strategy is commonly used by developers who learn from their experience. An example is Rovio, who developed many game apps before hitting the jackpot with Angry Birds. They then built on their success, releasing new apps such as Angry Birds Seasons, and Angry Birds Rio[2].

Apps. Each app artefact is built and uploaded by a developer agent. The features of the app are abstracted as a 10x10 feature grid (**F**) for each app. If a cell in **F** is filled, then the app offers that particular feature. A grid is used so that feature similarity can be represented in the future, e.g., features that are similar can be represented as cells that are near to one another on the grid. The cells in **F** are filled probabilistically if this is the developer's first app. Otherwise, the developer fills **F** with copies of the features from his own best app (as determined by the highest daily average downloads) with random mutation. The choice of which app to copy occurs when the developer is starting to build the app. If no apps by this developer have downloads, the developer fills **F** with a copy of his most recent app. There is a 0.5 probability that mutation occurs during a copy. Mutation is implemented by randomly selecting a filled cell in **F** and randomly "moving" it to an empty cell in **F**.

For ranking purposes, each app keeps a record of the total number of downloads it has received to date and the number of downloads it has received on each of the previous seven days. Each app has a probability of $P_{Infectious}$ to be infectious. If the app is infectious, users who download the app recommend it to their friends. Apps can be infectious because they have exciting features (e.g., Angry Birds). Apps can also have infectious features. For example, WhatsApp Messenger[3] (No. 1 in 99 countries) is a mobile messaging app that allows users to exchange messages without having to pay for SMS. The user needs his friends to download the app to receive his messages. His friends will, in turn, ask their friends to download the app. For simplicity, the AppEco model currently assumes that all apps are sold at the same price; the model of variations in app pricing and categories of apps is left for future work. Each app also records the time when it was uploaded.

Users. Inspired by the recommender systems literature [22], each user agent has preferences (or taste information) that determine the app features that it prefers. Developers are unaware of the users' preferences. The preferences of a user agent are abstracted as a 10x10 preference grid (**P**). The top right quadrant in **P** is always empty, to model features that are undesirable to all users. For example, no users want an app to have the features of a difficult-to-use or malicious program. The top left and bottom right quadrant in **P** are filled probabilistically, such that each cell in the grid has a probability P_{Pref} of being filled, to model features that are desirable to some users. The bottom left quadrant in **P** is filled probabilistically, such that each cell in the grid has a probability $2xP_{Pref}$ of being filled, to model popular features desirable to many users. An example preference grid is illustrated in Figure 2 (right).

If a cell in **P** is filled, then the user agent desires the feature represented by that cell. If the feature grid **F** of an app has a cell in the same location filled, then it means the app offers a feature desired by the user agent (i.e. the user is susceptible to infection by that app). For example, in Figure 2, all four of the features offered by App 1 match the user agent's preferences, but only two of the features offered by App 2 match the user agent's preferences. Using the AppEco model, an app such as Angry Birds (to which many users are susceptible) can be abstracted as an app with **F** that matches **P** of many users, while an app to which few users are susceptible has **F** that matches few or no users' **P**. For simplicity, preference matching is binary: filled cells either match or do not match.

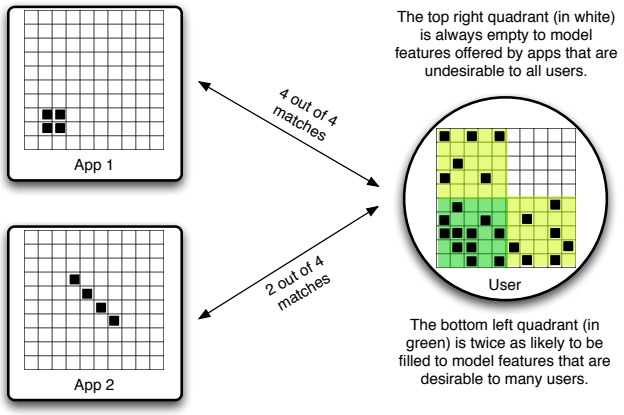

Figure 2. Matching app features with user preferences [2].

Each user agent keeps a record of the apps it has downloaded, the number of days between each browse of the app store (*daysBtwBrowse*, a random value between $[bro_{min}, bro_{max}]$), and the number of days that have elapsed since it last browsed the app store (*daysElapsed*). *daysElapsed* is recorded so that the user agent knows when to browse the app store next. When users are initialised at the start of the simulation, *daysElapsed* is set to be a random number between [0, *daysBtwBrowse*] so that users do not browse at the same time when they start. Users also record the number of friends they can influence and thus potentially "infect" (*numFriends*). The value of *numFriends* is a random number with a power law distribution in the range [0, 150]. (Many people will be able influence very few friends, but a few people can influence many friends.) The upper limit of this range is derived from the Dunbar number of 150 [23]. Dunbar [23] showed that the human brain is only capable of managing relationships with about 150 people (staying in contact at least once per year and knowing how friends relate to others). Dunbar suggests that this number remains the same despite new social networking technologies such as Facebook and Twitter[4].

App store. The app store is the environment used by the agents to store and access apps. Its primary function is to provide a shop front for users and enable them to locate and

[2]http://www.wired.co.uk/magazine/archive/2011/04/features/how-rovio-made-angry-birds-a-winner

[3] http://www.whatsapp.com/

[4]http://www.nytimes.com/2010/12/26/opinion/26dunbar.html?_r=2&ref=facebookinc

download apps that match their preferences. To achieve this, it provides three browsing methods: the Top Apps Chart, the New Apps Chart, and Keyword Search. These browsing methods provide changing subsets of apps to users; they are the "watering holes" of the ecosystem at which all users drink. As such, they provide a vital mode of transmission of apps to users. These three methods are modelled because they are common to many app stores, such as iOS, Android, and BlackBerry. The Top Apps Chart ranks apps based on the number of downloads the apps have received. The New Apps Chart displays apps that have recently been uploaded by developer agents; only a small subset of new apps is chosen for the chart. Keyword Search returns a list of apps that match the keyword entered by the user agent. In AppEco, Keyword Search is abstracted as a random search for a random number of apps. It is implemented in this way because keywords may not correspond to features, so a matching keyword does not mean the app has desirable features for the user.

B. AppEco Algorithm

The AppEco algorithm models the daily interactions between the AppEco components described in the previous section. Each timestep in the algorithm represents a day in the real ecosystem.

Inspired by the ecology literature [24], the population growth of user and developer agents is modelled using a sigmoid growth function commonly used to model the population growth in natural systems. The equation models the growth rate of user and developer agents in an app ecosystem declining as their population density increases, with the size of the ecosystem limited by the market share of the mobile platform. The population size at timestep t, pop_t, is defined by Equation 1 as follows.

$$pop_t = \text{MinPop} + \frac{(\text{MaxPop-MinPop})}{1 + e^{S*t-D}} \quad (1)$$

where MinPop is the minimum population, MaxPop is the maximum population, S determines the slope of the growth curve (S is negative for a growth curve), and D shifts the curve from left to right. Different growth formulas [24] can be used to model different ecosystems.

The AppEco algorithm is summarised in Figure 3 and detailed as follows.

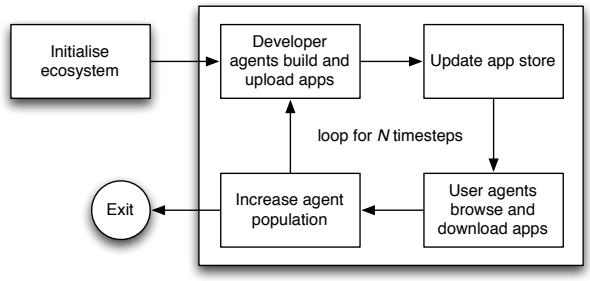

Figure 3. The AppEco algorithm.

Initialise ecosystem. This step launches AppEco with the population of developer and user agents as defined in Equation 1, with timestep $t = 0$. It is common for app stores to have apps before it is opened. For example, the iOS App Store had 500 apps the day it was launched[5]. As such, this step also creates an initial number of app artefacts ($N_{InitApp}$). The developers of these initial apps are randomly selected from the pool of initial developers. The attributes of initial developers, apps, and users are set as described in the previous section.

Developer agents build and upload apps. For each active developer, *daysTaken* is incremented by 1. If *daysTaken* exceeds this developer's *devDuration*, the app is completed. The developer then uploads the app to the store, resets *daysTaken* to 0. The *feature* attribute of the app is set such that each cell in the 10x10 feature grid has a probability P_{Feat} of being filled.

Update app store. The New Apps Chart is updated. When timestep $t = 0$, the New Apps Chart consists of a random selection of initial apps. In each following timestep, each new app has a probability $P_{OnNewChart}$ of appearing on the New Apps Chart. Apps are randomly selected here because the selection criteria are not the focus of this work and real app stores do not reveal how they select apps for the New Apps Chart. The maximum number of apps in the chart is defined by $N_{MaxNewChart}$. As newly selected apps are added to the chart, older apps appear lower in the chart and are no longer listed when their position exceeds the chart size. The Top Apps Chart is also updated. When timestep $t = 0$, the Top Apps Chart is empty because no apps have been downloaded yet. In each following timestep, each app is ranked based on the number of downloads it has received. The app ranking algorithm is configurable, and it is the purpose of this work to study different ranking algorithms. For example, if apps are ranked based on cumulative downloads, then apps with a higher number of cumulative downloads are ranked higher. If apps are ranked based on current downloads, then apps with a higher number of current downloads are ranked higher. The maximum number of apps in the chart is defined by $N_{MaxTopChart}$.

User agents browse and download apps. For each user, *daysElapsed* is incremented by 1. If *daysElapsed* exceeds *daysBtwBrowse*, then the user browses the app store and resets *daysElapsed* to 0. The user browses the New Apps Chart and the Top Apps Chart, and conducts Keyword Search (which returns a random number of apps between [key_{min}, key_{max}]). The user browses each app that it has not previously downloaded: the feature grid of the app is compared with the preference grid of the user. If all the features offered by the app match the user's preferences, then the user downloads the app. For example, in Figure 2, the user downloads App 1 but not App 2. If the user has downloaded an infectious app in the current timestep, the user will recommend the app to his friends who will then browse the app in the next timestep and download the app if it matches their preferences.

Increase agent population. This step increases the number of user and developer agents in the ecosystem for the next timestep, using Equation 1.

AppEco is implemented in C++ and the code can be requested from the authors via email. It is developed to be

[5] http://www.apple.com/pr/library/2008/07/10iPhone-3G-on-Sale-Tomorrow.html

highly configurable so that it can simulate various app ecosystems, such as iOS, Android, and BlackBerry.

IV. EXPERIMENTS

In order to investigate content organisation in the app store, we calibrate the simulation to match, as much as is feasible, the behaviour of a real app store. We use Apple's iOS App Store because it is one of the oldest and most established app stores. The calibration of AppEco to the iOS App Store is described in Section IV.A. We then investigate the effect of content organisation in the iOS App Store on app downloads. Two experiments are conducted.

Experiment 1 (Section IV.B) investigates the effect of using different app ranking algorithms to construct the Top Apps Chart. The Top Apps Chart is considered the most important chart, for it lists the most "successful" apps in order of their "success." Apps listed in this chart will be seen by many more users and will in turn receive more downloads – a positive feedback loop. Consequently this experiment examines how "success" should be calculated in order to maximise effectiveness of the app store.

Experiment 2 (Section IV.C) investigates the effect of the other major chart in the app store: the New Apps Chart (also known as the "New and Noteworthy" Chart in the iOS App Store). Only a selected few of the new apps are featured on this chart. While it is unclear how apps may be chosen, we are able to modify the rate at which apps appear on the chart and assess the result. Should the app store highlight apps on the New Apps Chart for several days at a time and maximise their visibility, perhaps pushing them into the Top Apps Chart? Or should the app store update the New Apps Chart frequently and give more apps an opportunity to appear? This experiment aims to answer these questions.

In both experiments, users browse the New Apps Chart and the Top Apps Chart, and conduct Keyword Search, in order to find apps that meet their preferences, as described in Section III.B. We introduce the download-to-browse (D-B) ratio as our performance metric. The D-B ratio is calculated as the total number of apps downloaded divided by the total number of apps browsed. A larger D-B ratio means a more effective shop front, for example, a D-B ratio of 0.2 means that for every five apps the user browses, one app matches the user's preferences and is downloaded. A D-B ratio of 1.0 means that the user finds the perfect app immediately. Since users only have a finite amount of time, a higher D-B ratio corresponds to a higher number of downloads from the app store, more profits for both the app store and the developers, and happier users.

A. Calibrating AppEco for iOS

We collected the following iOS data over a period of three years, from the start of the iOS ecosystem in July 2008 (Q4 2008) until the end of June 2011 (Q3 2011):

- **Number of iOS developers**. The number of iOS developers is based on the number of worldwide iOS developers month over month compiled by Gigaom[6].
- **Number of iOS apps and downloads**. The number of apps and downloads is based on statistics provided in Apple press releases and Apple Events[7]. For example, in the Apple Special Event on 9th September 2009, Apple CEO Steve Jobs announced the App Store to reach 75,000 apps and 1.8 billion downloads, and Apple's press release on 28th September 2009 announced that the App Store has achieved more than 85,000 apps and 2 billion downloads[8].
- **Number of iOS users**. The number of iOS users is based on the number of iOS devices (iPod Touch, iPhone, and iPad) sold by Apple over time. The sales figures are available from Apple's quarterly financial data[9], and for simplicity the calculation assumes that each user has one iOS device.

Using this and other publicly available data we calibrated AppEco to simulate the iOS app ecosystem. Table I summarises the calibrated values for the system constants. In order to match (curve-fit) the iOS user and developer growth rates, values such as D and S for users and developers were determined through tuning experiments.

TABLE I. CONSTANT VALUES RESULTING FROM IOS CALIBRATION

$[Pop_{minUser}, Pop_{maxUser}]$	[1500, 40000]	$[dev_{min}, dev_{max}]$	[1, 180]
D_{User}	-4.0	P_{Pref}	0.4
S_{User}	-0.0038	P_{Feat}	0.04
		$P_{OnNewChart}$	0.001
$[Pop_{minDev}, Pop_{maxDev}]$	[1000, 120000]	$P_{Infectious}$	0.0001
D_{Dev}	-4.0	$N_{MaxNewChart}$	40
S_{Dev}	-0.005	$N_{MaxTopChart}$	50
$N_{InitApp}$	500	$P_{Inactive}$	0.0027
$[bro_{min}, bro_{max}]$	[1, 360]	$[key_{min}, key_{max}]$	[0, 50]

It is computationally infeasible in terms of memory to simulate hundreds of millions of users. To ensure that the system is computationally feasible, one app represents one real app, and one developer agent represents one real developer, but one user agent represents 10,000 real users. As such, the value of *numFriends* for one user agent is the average *numFriends* for 10,000 real users. This coarse-grained simulation is necessary to enable the modelling of the entire app ecosystem using the available computing resources. Mobile app ecosystems are international ecosystems. App recommendations are not bounded by the users' physical location. For this reason, modelling just one country or a subset of app users would not provide an accurate simulation of the true app store ecosystem.

After calibration the behaviour of AppEco closely resembles the behaviour of the iOS ecosystem, including emergent rates such as the number of apps and downloads [2, 20]. A run of the simulation takes approximately 22 seconds CPU time on a MacBook Air with a 1.8GHz Intel Core i7 Processor and 4GB of 1333 MHz DDR3 memory. After three

[6] http://gigaom.com/apple/infographic-apple-app-stores-march-to-500000-apps/

[7] http://www.apple.com/apple-events/

[8] http://www.apple.com/pr/library/2009/09/28Apples-App-Store-Downloads-Top-Two-Billion.html

[9] http://www.apple.com/pr/library/

years (1080 timesteps assuming 30 days a month), the model typically contains more than 100,000 developer agents, 500,000 apps, 20,000 user agents (corresponding to 200m real users), and 1.5 million downloads (corresponding to 15bn real downloads).

B. Experiment 1: App Ranking Algorithm

1) Objective and Setup

The Top Apps Chart is perhaps the most significant chart for an app to be listed on. Appearing on this chart will trigger many more downloads from users. But how should apps be ranked on this chart? Developers are interested in knowing about the top app ranking algorithms because appearing on this chart increases the number of daily downloads by 23 times or even more [25]. Some developers even try to "game the system" to make their apps appear on the Top Apps Chart[10]. This work does not aim to help developers "cheat" – we focus on a larger question of interest to all: which ranking method would be most effective to improve the D-B ratio, and thus improving downloads and user satisfaction?

In some stores, such as iOS and Android, the top app ranking algorithms are not publicly available, and developers have hypothesised various algorithms. To study the effect of these different top app ranking algorithms on app downloads, we investigated the following:

- **Weighted 4-day Downloads** [26]. Apps are ranked in the order of decreasing score. The score of each app is:

$$score = 8D_1 + 5D_2 + 5D_3 + 3D_4$$

  where D_n is the number of downloads received by the app on the nth day before the current day.

- **Weighted 7-day Downloads**[11]. Apps are ranked in the order of decreasing score. The score of each app is:

$$score = 4D_1 + 3D_2 + 2D_3 + D_4 + D_5 + D_6 + D_7$$

  where D_n is the number of downloads received by the app on the nth day before the current day.

- **Current Downloads** [25]. Apps are ranked in order of decreasing total number of downloads they have received in the previous day.

- **Cumulative Downloads**[12]. Apps are ranked in order of decreasing total number of downloads they have received since they were uploaded.

AppEco was run for 1080 timesteps (corresponding to three years in the real world, assuming 30 days a month) using each of the four ranking algorithms. The experiment was repeated 100 times.

2) Results and Analysis

Table II shows the average download-to-browse (D-B) ratio and corresponding standard deviation for each ranking algorithm. The Current Downloads algorithm produced the highest average D-B ratio, followed by Weighted 4-day Downloads, Weighted 7-day Downloads, and Cumulative Downloads. Although the differences in the D-B ratio may appear subtle, with 15 billion downloads by the period ending Q3 2011, a small difference in the D-B ratio results in a large difference in downloads. For example, a difference of 0.0001 in the D-B ratio corresponds to a difference of more than 10 million downloads.

TABLE II. EXPERIMENT 1: D-B RATIO AT TIMESTEP $T = 1080$

App Ranking Algorithm	Average D-B Ratio	Standard Deviation
Weighted 4-day Downloads	0.0948	0.0034
Weighted 7-day Downloads	0.0918	0.0037
Current Downloads	**0.1020**	**0.0036**
Cumulative Downloads	0.0803	0.0046

Figure 4 illustrates the average D-B ratio for 100 runs for each algorithm over three years. The Cumulative Downloads algorithm performed poorly throughout. When this algorithm was used, older apps with a longer history in the app store tended to have higher cumulative downloads, which meant that they had a higher chance of appearing and staying on the Top Apps Chart. The Top Apps Chart changed very slowly as older apps kept dominating the chart. As a result, existing users were presented with the same apps repeatedly, which resulted in a low D-B ratio. At the other end of the spectrum, the Current Downloads algorithm produced a highly variable Top Apps Chart. This algorithm considered only the number of downloads on the previous day and so apps that appeared on the chart were likely to be replaced quickly.

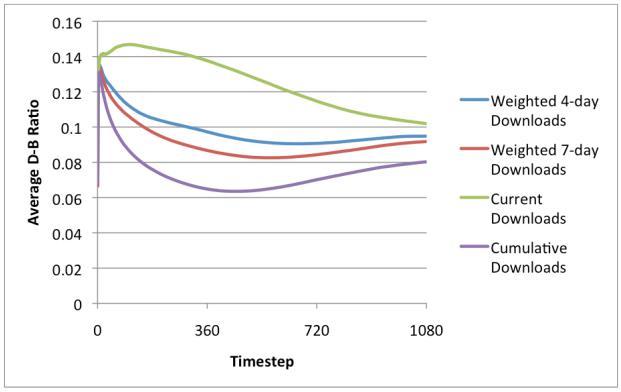

Figure 4. Experiment 1: Average D-B ratio over 100 runs.

The overall result from the first experiment seems clear: algorithms that produced a faster-changing Top Apps Chart tended to have a consistently higher D-B ratio. To assess the effect of the evolutionary developer strategy, the same experiments were performed with developers producing apps with random features. Fascinatingly, identical trends were seen, and the relative performances of the algorithms remained unaffected.

C. Experiment 2: New Apps Chart

1) Objective and Setup

As we have seen, a higher D-B ratio is achieved when the Top Apps Chart changes faster. In contrast, the New and Noteworthy Chart in Apple's iOS App Store changes slowly.

[10] http://www.pcworld.com/article/189973/apple_app_stores_dirty_little_secret.html

[11] http://tii.libsyn.com/index.php?post_id=522655

[12] http://www.iphonedevsdk.com/forum/promotion-techniques/89162-how-apple-rank-app-search-result.html

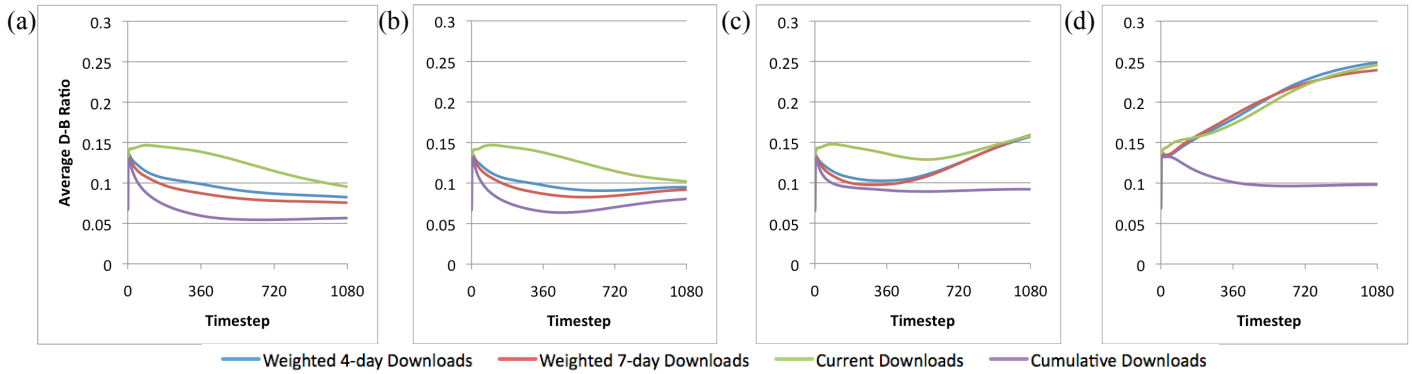

Figure 5. Experiment 2: Average D-B ratio for each ranking algorithm for $P_{OnNewChart}$ value of (a) 0.0001, (b) 0.001, (c) 0.01, and (d) 0.1. A higher $P_{OnNewChart}$ value means a more frequently updated New Apps Chart.

On average, only one to two new apps enter the chart each day, as evidenced by the data we gathered over several weeks on the frequency of changes to the New and Noteworthy Chart. The second experiment investigates the effect of the rate of change of the New Apps Chart to the D-B ratio. Four probabilities of new apps being selected for the New Apps Chart were investigated using AppEco:

- $P_{OnNewChart}$ = 0.0001 (On average over three years, approximately 2 new apps enter the New Apps Chart every month.)
- $P_{OnNewChart}$ = 0.001 (On average over three years, about 15 new apps enter the chart every month. This configuration most resembles the iOS App Store rate and is used in Experiment 1.)
- $P_{OnNewChart}$ = 0.01 (On average over three years, about 150 new apps enter the chart every month.)
- $P_{OnNewChart}$ = 0.1 (On average over three years, about 40 new apps enter the chart every day, i.e., the entire New Apps Chart is updated almost every day.)

Experiment 2 was run using the four ranking algorithms from Experiment 1. Similar to the previous experiment, Experiment 2 was also run for 1080 timesteps using different $P_{OnNewChart}$ values. Each experiment was repeated 100 times.

2) Results and Analysis

Table III shows the average download-to-browse (D-B) ratio and corresponding standard deviation for each ranking algorithm and $P_{OnNewChart}$ values. Regardless of the ranking algorithm, a faster-changing New Apps Chart (higher $P_{OnNewChart}$ value) produced a higher average D-B ratio. The results confirm that app stores will have higher downloads and customer satisfaction if they have a faster changing shop front.

Figure 5 illustrates the average D-B ratios for each algorithm with different $P_{OnNewChart}$ values for 100 runs over time. When the New Apps Chart changed very slowly, see Figure 5(a), the average D-B ratio for all four algorithms consistently dropped as the ecosystem matures. When the New Apps Chart was updated frequently, see Figure 5(c), three ranking algorithms (Weighted 4-day, Weighted 7-day, and Current Downloads) produced the same improving D-B ratios towards the end of the three years. Finally, when the New Apps Chart changed the most frequently, see Figure 5(d), Cumulative Downloads showed no improvement in the average D-B ratio, while the performance of the other algorithms rose to an impressive average D-B ratio of about 0.25. Again, to assess the effect of the evolutionary developer strategy, the same experiments were performed with developers producing apps with random features. Similar to the first experiment, identical trends were observed, and the relative performances of the algorithms remained unaffected.

TABLE III. EXPERIMENT 2: D-B RATIO AT TIMESTEP T = 1080

App Ranking Algorithm	$P_{OnNewChart}$	Avg. D-B Ratio	Std. Deviation
Weighted 4-day Downloads	0.0001	0.0824	0.0034
	0.001	0.0948	0.0034
	0.01	0.1570	0.0053
	0.1	**0.2490**	**0.0055**
Weighted 7-day Downloads	0.0001	0.0756	0.0037
	0.001	0.0918	0.0037
	0.01	0.1588	0.0052
	0.1	**0.2396**	**0.0045**
Current Downloads	0.0001	0.0953	0.0032
	0.001	0.1020	0.0036
	0.01	0.1579	0.0040
	0.1	**0.2458**	**0.0053**
Cumulative Downloads	0.0001	0.0566	0.0032
	0.001	0.0803	0.0046
	0.01	0.0920	0.0037
	0.1	**0.0980**	**0.0043**

These surprising findings suggest that a frequently updated New Apps Chart minimises the effect of different ranking algorithms on the Top Apps Chart – all algorithms, except Cumulative Downloads, eventually became equally effective. The findings also confirm that a faster changing shop front can improve downloads.

V. CONCLUSION

There may be an app for everything, but if the app store does not present its content effectively to the users, then users will never find the app they need. Juggling the Top Apps Chart and New Apps Chart is not an easy task. If apps are listed for too long on the New Apps Chart then few new apps can be featured. If "success" is measured poorly, then undesirable apps will linger, and download-to-browse ratios will fall.

In this work, we used AppEco, an Artificial Life evolutionary agent-based model that simulates app ecosystems to investigate these issues. AppEco demonstrated the complexity of the app store ecosystems that are now commonplace. Positive feedback within and across charts means that the effectiveness of the shop front is highly dependent on the speed at which content is updated. A slowly updated New Apps Chart will result in significant repercussions to the effectiveness of the Top Apps Chart. A Top Apps Chart that measures success by including too much historical data will lose downloads. These findings are valid regardless of developer strategy; the overwhelming effects of chart organisation on downloads mean that there is no advantage for "intelligent" evolutionary developer strategies compared to random development.

The findings have implications for app developers. The success of apps is highly dependent on how the app store chooses to present the apps. Even if the app has every feature that every user desires, if it never appears on an app store chart then it may be doomed to obscurity.

This paper studies mobile app ecosystems from the app store's perspective. There are many avenues for future work. Studies can be made from the user's perspective to understand how a user might best locate desirable apps and communicate their requirements and opinions about the apps back to developers. We have recently surveyed more than 10,000 people from 15 countries in order to collect data about their app usage behaviour to incorporate into our simulation. Through models such as AppEco, we anticipate that we can gain a deeper understanding of app ecosystems.

ACKNOWLEDGMENT

This work was supported in part by the European Commission Trans-Atlantic Research and Education Agenda in System of Systems Support Action INFSO-ICT-287593.

REFERENCES

[1] S. Baghdassarian and C. Milanesi, "Forecast: Mobile Application Stores, Worldwide, 2008-2014," Gartner, 2010.

[2] S. L. Lim and P. J. Bentley, "App Epidemics: Modelling the Effects of Publicity in a Mobile App Ecosystem," in *13th Int. Conf. on the Synthesis and Simulation of Living Systems (ALIFE)*, 2012, pp. 202-209.

[3] S. L. Lim and P. Bentley, "From natural to artificial ecosystems," in *Frontiers of Natural Computing*, 2012, p. 15.

[4] F. Lin and W. Ye, "Operating System Battle in the Ecosystem of Smartphone Industry," in *Int. Symp. on Information Engineering and E-Commerce*, 2009, pp. 617-621.

[5] R. Garg and R. Telang, "Estimating App Demand from Publicly Available Data," School of Information Systems and Management, Heinz College, Carnegie Mellon University, 2011.

[6] M. Bohmer, B. Hecht, J. Schoning, A. Kruger, and G. Bauer, "Falling asleep with Angry Birds, Facebook and Kindle: a large scale study on mobile application usage," in *MobileHCI 2011*, 2011, pp. 47-56.

[7] T. A. Kohler, G. J. Gumerman, and R. G. Reynolds, "Simulating ancient societies," *Scientific American,* vol. 293, pp. 76-84, 2005.

[8] T. Wilkinson, J. Christiansen, J. Ur, M. Widell, and M. Altaweel, "Urbanization within a dynamic environment: modeling Bronze Age communities in Upper Mesopotamia," *American Anthropologist,* vol. 109, pp. 52-68, 2007.

[9] C. Y. Huang, C. T. Sun, C. Y. Cheng, and Y. S. Tsai, "Resource limitations, transmission costs and critical thresholds in scale-free networks," in *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems*, 2008, pp. 1121-1128.

[10] T. Bosse and C. Gerritsen, "Agent-based simulation of the spatial dynamics of crime: on the interplay between criminal hot spots and reputation," in *Proc. of the 7th Int. Conf. on Autonomous Agents and Multiagent Systems*, 2008, pp. 1129-1136.

[11] T. Lux and M. Marchesi, "Scaling and criticality in a stochastic multi-agent model of a financial market," *Nature,* vol. 397, pp. 498-500, 1999.

[12] J. H. Holland, *Adaptation in Natural and Artificial Systems*. 2nd ed. Cambridge, MA: MIT Press, 1992.

[13] R. L. Olson and R. A. Sequeira, "An emergent computational approach to the study of ecosystem dynamics," *Ecological Modelling,* vol. 79, pp. 95-120, 1995.

[14] E. Pachepsky, T. Taylor, and S. Jones, "Mutualism promotes diversity and stability in a simple artificial ecosystem," *Artificial Life,* vol. 8, pp. 5-24, 2002.

[15] M. Antona, F. Bousquet, C. LePage, J. Weber, A. Karsenty, and P. Guizol, "Economic theory of renewable resource management: A multi-agent system approach," *Multi-Agent Systems and Agent-Based Simulation,* vol. 1534, pp. 61-78, 1998.

[16] B. Alexandrova-Kabadjova, E. Tsang, and A. Krause, "Competition is bad for consumers: Analysis of an artificial payment card market," *Journal of Advanced Computational Intelligence and Intelligent Informatics,* vol. 15, pp. 188-196, 2011.

[17] A. Omicini, A. Ricci, and M. Viroli, "Artifacts in the A&A meta-model for multi-agent systems," *Autonomous Agents and Multi-Agent Systems,* vol. 17, pp. 432-456, 2008.

[18] E. R. Miranda, "Emergent songs by social robots," *Journal of Experimental and Theoretical Artificial Intelligence,* vol. 20, pp. 319-334, 2008.

[19] M. Wheeler, J. Ziman, and M. A. Boden, Eds., *The Evolution of Cultural Entities*. OUP/British Academy, 2002.

[20] S. L. Lim and P. J. Bentley, "How to become a successful app developer? Lessons from the simulation of an app ecosystem," in *Genetic and Evolutionary Computation Conf. (GECCO)*, 2012, pp. 129-136.

[21] M. A. Cusumano, "Platforms and services: Understanding the resurgence of Apple," *Comms. of the ACM,* vol. 53, pp. 22-24, 2010.

[22] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Trans. on Knowledge and Data Engineering,* vol. 17, pp. 734-749, 2005.

[23] R. I. M. Dunbar, "Neocortex size as a constraint on group size in primates," *Journal of Human Evolution,* vol. 22, pp. 469-493, 1992.

[24] S. E. Kingsland, *Modeling Nature: Episodes in the History of Population Ecology*. University of Chicago Press, 1995.

[25] A. Ansar, "AppStore Secrets (What We've Learned from 30,000,000 Downloads)," Pinch Media, 2009.

[26] M.-C. Lanfranchi, B. Benezet, and R. Perrier, "How to Successfully Market your iPhone Application," faberNovel, 2010.