COMPETITION WITH OPEN SOURCE AS A PUBLIC GOOD

ERNAN HARUVY, ASHUTOSH PRASAD AND SURESH SETHI
School of Management
The University of Texas at Dallas
Richardson, TX 75080, USA

RONG ZHANG
Chongqing University
Chongqing 400044, China

(Communicated by Kok Lay Teo)

Abstract. The open source paradigm is often defined as a “collaborative effort,” implying that firms and consumers come together in a non-competitive climate. We show here that open source development can arise from a competitive climate. Under competition, we find that open source is the surplus maximizing outcome and can be in equilibrium if cost asymmetries are small. However, when cost asymmetries are large, contradictions between equilibrium and welfare maximization result. Considerations typical to public good problems arise, with issues of asymmetric contributions and free-riding. These issues should guide the firm’s as well as the society’s decisions to implement open source in particular environments. We analyze this problem in the framework of a dynamic duopolistic competition, with firms controlling their investments in software.

1. Introduction. The open source movement has gained much momentum since its early days. Open source projects such as Linux, Apache, and Perl have become successful, capturing market share from well established closed source competitors. This enormous success has made open source development an important economic phenomenon. Advantages to firms resulting from this form of development include savings on software purchases or software development, increased creativity associated with freedom from company restrictions, and greater development speed arising from rapid dissemination and higher relevance to end user needs [20].

In open source development, users of the software – the public as well as firms – collaborate on the development of the software by contributing code and bug reports and by sharing and disseminating the code. As such, open source can be considered a public good. A public good is defined as a good which is non-rival in consumption and non-excludable [23]. That is, one user’s enjoyment of the product does not interfere with another and no one can be excluded from using the good. Public goods are susceptible to free-riding (e.g., [12]), where firms and individuals would enjoy the benefits of the public good without contributing. Much of the research on open source is concerned with the incentives and why contributors freely

2000 Mathematics Subject Classification. Primary: 49N90, 90B50; Secondary: 91A23, 91A05, 49N70.

Key words and phrases. Differential games, public goods, open source, software.
give away their code rather than free-ride. Reasons proposed in the literature include signaling motives for future jobs or consulting opportunities, ego-gratification, political convictions, social concerns, and community motives. A few studies in the recent literature focus on empirical research into the motives behind open source development (e.g., [6], [11], [13], [14]). However, these studies focus on the motivations of unpaid volunteers (who often refer to themselves as hackers) who are not directly paid to work on open source projects. In the early days of the open source movement, most projects were driven by such hackers. Today, firms such as IBM, Apple, Hewlett Packard, Sun, Motorola, Nokia, and others sponsor open source development and have in-house teams of developers who are employed full-time to develop open source code. The power of hackers in determining the direction of the open source movement is fast eroding.

2. Background literature. The emerging shift in development sources does not change the fact the open source is a public good. It merely shifts the focus from developer motives to firm motives. Firms tend to be profit-motivated (with some notable exceptions). Given that the open source code is free, a firm which coordinates and invests its own resources in an open source project generally has a commercially sold product that would benefit from the open source code by either nesting the code within its own commercially sold product, or by exploiting the complementarities that are present between the code and one of its commercially sold products. Haruvy, Prasad and Sethi [9] evaluate the former model—where the open source code is sold as part of a commercial product. In that scenario, open source code may become a component of closed source code and the software can be made commercial. This can be done while keeping the source open by providing packaging, distribution, service or brand name or by closing the source, such as Microsoft’s use of the BSD (Berkeley Software Distribution) code, or the BSDi operating system (derived from Unix BSD).

In this work we examine the latter model in a competitive dynamic setting. There are various approaches in the literature to modeling competitive dynamic optimization in finite horizon. Bass et al. [1] model competitive advertising in finite horizon with closed loop Nash equilibrium solutions. While they obtain closed form solutions for symmetric firms, numerical solutions are employed for asymmetric firms. Mou and Yong [17] formulate an open loop two-player zero sum game in a linear quadratic form and this permits them to use a generalized differential Riccati equation to obtain a solution. Liou, Schaible and Yao [16] model competition...
between a buyer and a supplier in a Stackelberg framework where the supplier is the leader (declaring its strategy first) and the buyer is the follower (taking the supplier strategy as given) in a deterministic industry. The leader-follower structure allows for a solution with mixed integer bilevel programming.

The differential games approach taken in this work is ideal since the firms are able to vary the price of their commercial products and investment in open source as well as proprietary technology from period to period and since related variables (state variables) such as quality and network size vary over time. Such approach has been often taken in the literature to address dynamic pricing problems (e.g., [2]), dynamic quality problems (e.g., [18]) and dynamic advertising expenditure problems (for a survey of that literature see Feichtinger, Hartl and Sethi [5]). The approach presented here allows for dynamic pricing in the presence of network externalities (e.g., [4]) as well as for dynamic quality, where the former is a control variable and the latter is a state variable.

The open source code is free but complements another product, service, or software that is sold commercially. The open source business models discussed in the literature [20], [21] suggest several variations on this concept: The Loss-Leader Model uses the open source code to establish brand name and lead users to the firm’s other products. Netscape’s Mozilla web browser is an example for such a practice. Netscape’s early business model relied upon the sale of server software. The Compatible selling model has an open source software compatible with the firm’s commercially sold software, hardware or service. An example is the Apple-MaxOS X software which is compatible with Apple’s hardware. The Service and Support model, of which Red Hat is a prominent example, provides service and support for free to complement the open source code. Sun’s Star Office falls under the Brand Compatibility model wherein the open source software complements the firm’s product lines and brand image. Finally, under the Information model a firm charges for information associated with the open source code. Consulting services often charge for information relating to open source documentation, maintenance, etc. Notice that the models discussed here can also apply to a freeware software product – a software product which is free but whose code is not necessarily open to the public. However, the key difference lies in the development efforts which are less expensive and more rapid in the open source model. For a discussion of when the freeware model would be most appropriate, see [7] and [8].

| Business Model         | Example                                      |
|------------------------|----------------------------------------------|
| Loss-Leader            | Netscape’s Mozilla                           |
| Compatible Selling     | Apple MaxOS X                                 |
| Service and Support    | Red Hat                                      |
| Brand Compatibility    | Sun’s StarOffice                             |
| Information            | Wide variety of vendors, including IBM, Oracle, and consulting firms |

Note that an open source product in the categories discussed here need not be a full software product. It can be a more limited group of modules or classes for existing open source software. For example, IBM initiates and supports various
Apache-related projects and releases related code even when it is developed in-house. IBM’s main source of revenue in this case is derived from selling web servers, which benefit greatly from improvements and additions to Apache.

Haruvy et al. [10] examined a model of competition wherein each firm exploits complementarities between a potentially open source code and a commercially sold product. Their model postulated demands that were linear in qualities of the open source and proprietary products. While their work sheds light on some public good aspects such as free-riding, their demand functions (which are standard demand functions in the economics literature) cannot be used to shed light on firm choice between open and closed source formats since linear demand functions in the quality of the non-commercial code imply that each firm strictly prefers open source to closed source. In the present work in contrast, we study realistic demand functions with a bounded market potential that generates rivalry in all aspects of quality. This rivalry in turn implies preferences over open vs. closed source format that depend on environmental parameters.

The questions we are concerned with in the present work are (1) whether competing firms in a dynamic environment find it useful to pursue open source development as opposed to closed source, (2) which firms tend to contribute more and which to free ride, and (3) which firms benefit most from open source development (the developers or the free-riders). We also look at social issues such as (4) software quality under open and closed source and (5) welfare improvement due to open source.

3. Model and analysis. Two firms have competing software products. Each firm can invest in a common production component (potentially open source) to both firms. Each firm also invests in its own proprietary software, which is private. The question that we ask in this work is whether both firms will find it optimal to invest in the public good or whether one firm will find it optimal to free-ride on the other.

There are two stages (e.g., [15], [24]). In the first stage, firms must decide whether or not to switch to open source. In stage 2, the firms optimize over investment in open source and proprietary technology dynamically for the duration of the game. Figure 1 illustrates the decision sequence firms face.

![Figure 1. The Software Strategy Decision Sequence.](image-url)
Dynamic Variables are:

- \( S_i(t) \) Quality of common component software developed by firm \( i \) - state variable
- \( H_i(t) \) Quality of proprietary software for firm \( i \) - state variable
- \( x_i(t) \) The market share of firm \( i \)'s proprietary product - state variable
- \( a_i(t) \) Investment in common component software by firm \( i \) - decision variable
- \( u_i(t) \) Investment in proprietary software by firm \( i \) - decision variable

Model parameters are:

- \( p \) Price of software
- \( \eta_1, \eta_2 \) Productivity parameters
- \( \gamma_1, \gamma_2, \gamma_3 \) Demand parameters
- \( c_{S1}, c_{S2} \) Cost parameters for the common component software
- \( c_{H1}, c_{H2} \) Cost parameter for the private component software
- \( T \) Duration of the game

3.1. The open source model. As mentioned above, in the open source model, the common component is shared between the firms. Demand growth rates for the proprietary software products are dependent on the two firms’ qualities according to a linear relationship. We model firm \( i \)'s demand as:

\[
D_i(t) = \gamma_1(S_1 + S_2) + \gamma_2 H_i - \gamma_3 H_{3-i}, \quad i = 1, 2
\]

The \( i^{th} \) firm’s market share grows at the rate:

\[
\dot{x}_i(t) = D_i(t)(1 - x_1 - x_2) = (\gamma_1(S_1 + S_2) + \gamma_2 H_i - \gamma_3 H_{3-i})(1 - x_1 - x_2), \quad x_i(0) = x_{i0} \quad i = 1, 2
\]

This demand growth rate is a product of a standard demand function and the fixed market potential growth rate common in optimal control problems. The software products' qualities grow at a rate determined by the firms’ respective investments.

\[
\dot{S}_i(t) = \eta_i a_i(t), \quad S_i(0) = Q_{i0}^S \quad i = 1, 2
\]

\[
\dot{H}_i(t) = u_i(t), \quad H_i(0) = Q_{i0}^H \quad i = 1, 2
\]

The objective of each of the competing firms is to maximize its own profit over a finite planning horizon subject to the costs of investment in proprietary and common component software, \( c_{H1}, c_{H2}, c_{S1}, \) and \( c_{S2} \). Since there are two firms, there are two objective functions, one for each firm, to be solved simultaneously. Equation (5) defines the pair of objective functions faced by the firms and therefore the differential game to be played. Each firm must optimize its objective function taking into account that the other firm is optimizing as well. We use an open-loop approach, which means that each firm’s course of action evolves over time. Firm \( i \)'s problem is

\[
\max_{a_i, u_i} \int_0^T (px_i - c_{S_i}a_i^2 - c_{H_i}u_i^2)dt,
\]
In the closed source model, neither firm can benefit from the other firm’s common component quality. The market share growth rate is now:

\[ \dot{x}_i(t) = (\gamma_1 S_i + \gamma_2 H_i - \gamma_3 H_{3-i}) (1 - x_i - x_{3-i}), \quad i = 1, 2 \]  

(9)

The firm’s problem (to be solved for both firms) becomes:

\[ \max_{a_i, u_i} \int_0^T (px_i - cS_i a_i^2 - cH_i u_i^2) dt \quad i = 1, 2 \]  

(10)
subject to (3), (4) and (9).

The Lagrangians can be written as:

\[ L_1 = px_1 - c_{S1}a_1^2 - c_{H1}u_1^2 + \lambda_0 \eta_1 a_1 + \lambda_1 \eta_2 a_2 + \lambda_2 u_1 + \lambda_3 u_2 + \lambda_4 (\gamma_0 S_1 - \gamma_1 S_2 + \gamma_2 H_1 - \gamma_3 H_2)(1 - x_1 - x_2) + \lambda_5 (\gamma_0 S_2 - \gamma_1 S_1 + \gamma_2 H_2 - \gamma_3 H_1)(1 - x_1 - x_2) \] (11)

\[ L_2 = px_2 - c_{S2}a_2^2 - c_{H2}u_2^2 + \mu_0 \eta_1 a_1 + \mu_1 \eta_2 a_2 + \mu_2 u_1 + \mu_3 u_2 + \mu_4(\gamma_0 S_1 - \gamma_1 S_2 + \gamma_2 H_1 - \gamma_3 H_2)(1 - x_1 - x_2) + \mu_5(\gamma_0 S_2 - \gamma_1 S_1 + \gamma_2 H_2 - \gamma_3 H_1)(1 - x_1 - x_2) \] (12)

This problem can be represented by 18 equations of motion, shown below:

\[ \dot{S}_1 = 0.5n_1^2 \lambda_0/c_{S1} \]
\[ \dot{S}_2 = 0.5n_1^2 \mu_1/c_{S1} \]
\[ \dot{H}_1 = 0.5 \lambda_2/c_{H1} \]
\[ \dot{H}_2 = 0.5 \mu_3/c_{H2} \]
\[ \dot{x}_1 = (\gamma_0 S_1 - \gamma_1 S_2 + \gamma_2 H_1 - \gamma_3 H_2)(1 - x_1 - x_2) \]
\[ \dot{x}_2 = (\gamma_0 S_2 - \gamma_1 S_1 + \gamma_2 H_2 - \gamma_3 H_1)(1 - x_1 - x_2) \]
\[ \dot{\lambda}_0 = -\lambda_4 \gamma_0 (1 - x_1 - x_2) + \lambda_5 \gamma_1 (1 - x_1 - x_2) \]
\[ \dot{\lambda}_1 = \lambda_4 \gamma_1 (1 - x_1 - x_2) - \lambda_5 \gamma_0 (1 - x_1 - x_2) \]
\[ \dot{\lambda}_2 = -\lambda_4 \gamma_2 (1 - x_1 - x_2) + \lambda_5 \gamma_3 (1 - x_1 - x_2) \]
\[ \dot{\lambda}_3 = \lambda_4 \gamma_3 (1 - x_1 - x_2) - \lambda_5 \gamma_2 (1 - x_1 - x_2) \]
\[ \dot{\lambda}_4 = -p + \lambda_4 (\gamma_0 S_1 - \gamma_1 S_2 + \gamma_2 H_1 - \gamma_3 H_2) + \lambda_5 (\gamma_0 S_2 - \gamma_1 S_1 + \gamma_2 H_2 - \gamma_3 H_1) \]
\[ \dot{\lambda}_5 = \lambda_4 (\gamma_0 S_1 - \gamma_1 S_2 + \gamma_2 H_1 - \gamma_3 H_2) + \lambda_5 (\gamma_0 S_2 - \gamma_1 S_1 + \gamma_2 H_2 - \gamma_3 H_1) \]
\[ \dot{\mu}_0 = -\mu_4 \gamma_0 (1 - x_1 - x_2) + \mu_5 \gamma_1 (1 - x_1 - x_2) \]
\[ \dot{\mu}_1 = \mu_4 \gamma_1 (1 - x_1 - x_2) - \mu_5 \gamma_0 (1 - x_1 - x_2) \]
\[ \dot{\mu}_2 = -\mu_4 \gamma_2 (1 - x_1 - x_2) + \mu_5 \gamma_3 (1 - x_1 - x_2) \]
\[ \dot{\mu}_3 = \mu_4 \gamma_3 (1 - x_1 - x_2) - \mu_5 \gamma_2 (1 - x_1 - x_2) \]
\[ \dot{\mu}_4 = \mu_4 (\gamma_0 S_1 - \gamma_1 S_2 + \gamma_2 H_1 - \gamma_3 H_2) + \mu_5 (\gamma_0 S_2 - \gamma_1 S_1 + \gamma_2 H_2 - \gamma_3 H_1) \]
\[ \dot{\mu}_5 = -p + \mu_4 (\gamma_0 S_1 - \gamma_1 S_2 + \gamma_2 H_1 - \gamma_3 H_2) + \mu_5 (\gamma_0 S_2 - \gamma_1 S_1 + \gamma_2 H_2 - \gamma_3 H_1) \]

with the boundary conditions

\[ \lambda_i(T) = \mu_i(T) = 0. \]

4. **Numerical analysis and discussion.** By definition of the problem, both firms have to choose open source in order for the open source case to apply. As long as even one firm chooses to close its source, the two source codes for the common component are no longer shared and we assume that this reduces to the closed source case\(^3\). In other words, while there are four strategy combinations \(- (1)\) both

\(^3\) Alternatively, the firm which chooses to close its source could take code from the other firm’s open source to incorporate to its common component, without opening its source. This would never be in equilibrium in our setting so we can ignore this scenario.
firms pursue open source, (2) the first firm chooses open source and the second closed source, (3) the first firm chooses closed source and the second open source, or (4) both choose closed source – there are only two possible outcomes. Hence, the payoffs are the same in three of the four strategy combinations involving at least one firm pursuing closed source. As such, we need to solve the problem for two cases only. The 15 equations of motion for the open source case and the 18 equations of motion for the closed source case, along with the starting values for the states and final values for the co-states, give us two-point boundary value problems which are solvable numerically.

The terminal values of all state variables in both systems of equations are 0. Hence, we set the objective function to the sum of the squared terminal values and use the Nelder-Mead simplex method, also known as the amoeba algorithm, in Fortran-77 to minimize the objective function with respect to the initial values. The algorithm is based on evaluating a function at the vertices of a simplex, then iteratively shrinking the simplex as better points are found until some desired bound is obtained [19]. We use multiple starting values to ensure global convergence and ensure that the converged objective function is infinitesimally close to 0.

We generated different sets of parameter values for the two cases and note the results below. For illustration purposes, we present five sets of parameter values, each for two cases. The sets of parameter values are given in Table 2.

### Table 2. Parameter Values for Scenarios of Interest.

| Parameter | Symmetric | Moderately Asymmetric | Highly Asym.1 | Highly Asym.2 | Highly Asym.3 |
|-----------|-----------|-----------------------|--------------|--------------|--------------|
| $P_1$     | 1         | 1                     | 1            | 1            | 1            |
| $P_2$     | 1         | 1                     | 1            | 1            | 1            |
| $\gamma_1$ | 0.5      | 0.5                   | 0.5          | 0.5          | 0.5          |
| $\gamma_2$ | 0.5      | 0.5                   | 0.5          | 0.5          | 0.5          |
| $\gamma_3$ | 0.5      | 0.5                   | 0.5          | 0.5          | 0.5          |
| $\eta_1$  | 1         | 1                     | 1            | 1            | 1            |
| $\eta_2$  | 1         | 1                     | 1            | 1            | 1            |
| $c_{S1}$  | 5         | 5                     | 1            | 1            | 5            |
| $c_{S2}$  | 5         | 50                    | 200          | 50           | 200          |
| $c_{H1}$  | 5         | 5                     | 5            | 5            | 5            |
| $c_{H2}$  | 5         | 5                     | 5            | 5            | 5            |
| $T$       | 5         | 5                     | 5            | 5            | 5            |
| $S_{1}(0)$ | 0.25     | 0.25                  | 0.25         | 0.25         | 0.25         |
| $S_{2}(0)$ | 0.25     | 0.25                  | 0.25         | 0.25         | 0.25         |
| $H_{1}(0)$ | 0.5      | 0.5                   | 0.5          | 0.5          | 0.5          |
| $H_{2}(0)$ | 0.5      | 0.5                   | 0.5          | 0.5          | 0.5          |
| $x_{1}(0)$ | 0.1      | 0.1                   | 0.1          | 0.1          | 0.1          |
| $x_{2}(0)$ | 0.1      | 0.1                   | 0.1          | 0.1          | 0.1          |

The cumulative profits for each firm under each case under each set of parameter values are shown in Table 3. Our first observation from Table 3 (and from other simulations) is that open source maximizes joint surplus.
Table 3. Profits for the Different Scenarios.

| Case                                      | Firm 1 Profit | Firm 2 Profit | Total Profit |
|-------------------------------------------|---------------|---------------|--------------|
| Symmetric Open Source                     | 1.71          | 1.71          | 3.42         |
| Symmetric Closed Source                   | 1.21          | 1.21          | 2.42         |
| Moderately Asymmetric Open Source         | 1.65          | 1.68          | 3.33         |
| Moderately Asymmetric Closed Source       | 1.25          | 0.98          | 2.23         |
| Highly Asymmetric Open Source1            | 1.77          | 1.83          | 3.60         |
| Highly Asymmetric Closed Source1          | 1.81          | 1.13          | 2.94         |
| Highly Asymmetric Open Source2            | 1.70          | 1.94          | 3.64         |
| Highly Asymmetric Closed Source2          | 1.82          | 1.13          | 2.95         |
| Highly Asymmetric Open Source3            | 1.65          | 1.67          | 3.32         |
| Highly Asymmetric Closed Source3          | 1.32          | 0.92          | 2.24         |

**Result 1.** Open Source improves total firm surplus over closed source.

The above observation can be seen from the third column. For the symmetric case, the joint surplus improves from 2.42 to 3.42. In the moderately asymmetric case, open source improves joint surplus from 2.23 to 3.33, and in the highly asymmetric cases 1-3, joint surplus improves from 2.94 to 3.60, from 2.95 to 3.64, and from 2.24 to 3.32.

Next, we observe that free-riding is so substantial that in all asymmetric open source cases, the seemingly cost advantaged firm is actually worse off.

**Result 2.** Under open source, with asymmetric costs for the common component, the higher-cost competitor will earn higher profit than the lower-cost competitor. This is reversed under closed source.

This result can be seen in Table 3. In the moderately asymmetric case, the higher cost firm (Firm 2) earns 1.68, beating Firm 1’s profits of 1.65. The difference is higher in highly asymmetric case 1, where Firm 2 earns 1.83, relative to Firm 1’s 1.77. It is highest in highly asymmetric case 3, with a difference of 0.24 (1.94 for Firm 2 versus 1.70 for Firm 1).

This apparently counterintuitive result is actually fairly simple to explain. Since in all our examples, the two firms have identical costs and prices to the proprietary product, and since they have access to the same common component, there will be no difference between them on the revenue side. The difference will come from the cost side. On the cost side, the firm with the lower cost of the common component will produce more of the common component (by the marginal principle) and will therefore incur higher cost. The firm with the higher cost will free-ride and will incur lower cost, thereby having higher overall profit. To see that, Figure 2 shows the two firms’ investment in the common component in the moderately asymmetric case under open and closed source.

We next turn to equilibrium calculations. To do that, we break down Table 3 into smaller 2x2 matrices which are normal form representations the extensive form game depicted by Figure 1. For the moderately asymmetric scenario, the game’s normal form representation is shown in Table 4. This representation is a normal-form game matrix. The first number in each cell denotes the profit to Firm 1, who is the row player. The second number in each cell denotes the profit to Firm 2,
who is the column player. From the above representation, it is clear that both firms pursuing open source is in equilibrium and it is the Pareto efficient outcome. Note that both firms pursuing closed source is a weak equilibrium as well.

**Table 4. Normal Form: Moderately Asymmetric Case.**

|        | Firm 2 OS | Firm 2 CS |
|--------|-----------|-----------|
| Firm 1 OS | 1.65, 1.68 | 1.25, 0.98 |
| Firm 1 CS | 1.25, 0.98 | 1.25, 0.98 |

**Result 3.** When the firms are symmetric or moderately asymmetric, open source will be the equilibrium as well as Pareto efficient outcome.

To see the first claim of Result 3, regarding equilibrium, note that in the symmetric case and the moderately asymmetric case, open source is a weakly dominant strategy for both firms. If one pursues a closed source strategy, the other will be no worse off by pursuing an open source strategy. On the other hand, if either pursues an open source strategy, the other will be strictly better off by adopting an open source strategy. The second claim of Result 3 comes from open source being welfare maximizing (Result 1) as well as the open source being in equilibrium, resulting in open source being Pareto efficient.

In contrast to the symmetric and moderately asymmetric cases, open source is not in equilibrium in some of the highly asymmetric cases.

**Result 4.** In some cases where the firms are highly asymmetric, open source is not in equilibrium and closed source is the unique equilibrium.

Compare the game of Table 4 to the highly asymmetric case 1 in Table 5. It is clear that now both firms pursuing open source is not in equilibrium. Firm 1 would not find it optimal to pursue OS here.

Note that this is the case for highly asymmetric cases 1 and 2 but not for highly asymmetric case 3. Highly asymmetric case 3 is little different from the moderately
asymmetric case. Essentially, Firm 1 is happy to offer free-riding for Firm 2 when its cost is 5, but not when its cost is 1. To see why, one would have to look at the market shares of both firms under asymmetric case 1 and asymmetric case 3 as shown in Figure 3.

In asymmetric case 1, the closed source strategy brings Firm 1’s market share to 0.624. In asymmetric case 3, Firm 1 can at best, under a closed source strategy, reach a market share of 0.56. As such, Firm 1 has a greater incentive to compete under asymmetric case 1.

5. **Conclusions.** We have examined the question of why and how much commercial software firms should contribute to open source software development. We use a model of duopoly competition.

First, it is possible to model the market situation and dynamics using optimal control theory. The model is quite complex and involves six state and four control variables. Following this, numerical analysis was conducted for a wide range of parameter values representing different firm parameters (e.g., symmetric and asymmetric firms).

Second, we find that from a purely welfare maximizing point of view, open source is always optimal. That is, open source maximizes joint surplus by the competing firms. This has implications for social planners, such as government agencies, interested in improving societal welfare. It appears that encouraging open source development is not a mere redistribution of the (profit) pie but rather results in a larger pie to be distributed.
Third, we find that the higher cost competitor, in terms of common component development cost, will be strictly better off than the lower cost competitor under open source. This creates disincentives to develop cost-lowering production capacity. This is the age-old free riding phenomenon at its worst. Those who invest in capacity and technology will find themselves at a disadvantage later on since the bulk of the burden of investment in the open source component will fall on them. As expected from public-goods literature, we find that weaker firms contribute less.

Fourth, we find that equilibrium outcomes need not be the welfare maximizing outcomes. In some case, where firms are highly asymmetric, the cost advantaged firm will opt to close its source code for the common component, thereby increasing its market share and profits vis-à-vis its competitor. In these cases, it is advantageous for a firm to be the low cost software manufacturer and investment in cost reduction technology is rewarded.

In cases where the closed source outcome is the unique equilibrium, it is nevertheless welfare reducing relative to open source. Future research should investigate regulations such as price controls (e.g., [3]) as a possible remedy as well as revenue sharing programs to help move the equilibrium back to open source. Another possible shortcoming of the above analysis is its focus on the profit motive. Though it is reasonable to assume that companies are profit driven, there are also thousands of contributors who contribute freely. Future research should extend the above results to include the effect of an outside community or OS programmers who contribute without regard to profit. Finally, open source initiatives are often led by a large first mover such as IBM or Sun and an alternative modeling framework could be a Stackelberg game as in [16].

Acknowledgements. Rong Zhang was a Visiting Scholar at the University of Texas at Dallas from April 2004 to April 2005. This work was sponsored by the NSFC (Grant No. 70371030 and 70771118) and by UTD. We would like to thank the referees very much for their valuable comments and suggestions.

REFERENCES

[1] F. M. Bass, A. Krishnamoorthy, A. Prasad and S. P. Sethi, Advertising competition with market expansion for finite horizon firms, J. Industrial and Management Optimization, 1 (2005), 1–19.
[2] P. K. Chintagunta and V. R. Rao, Pricing strategies in a dynamic duopoly: A differential game model, Management Science, 42 (1996), 1501–1514.
[3] P. Daniele, S. Giuffrè and S. Pia, Competitive financial equilibrium problems with policy interventions, J. Industrial and Management Optimization, 1 (2005), 39–52.
[4] A. Dhebar and S. Oren, Optimal dynamic pricing for expanding networks, Marketing Science, 4 (1985), 336–351.
[5] G. Feichtinger, R. F. Hartl and S. P. Sethi, Dynamic optimal control models in advertising: Recent developments, Management Science, 40 (1994), 195–226.
[6] A. Hars and S. Ou, Working for free? Motivations for participating in open-source projects, Int. J. Electronic Commerce, 6 (2002), 25–39.
[7] E. Haruvy and A. Prasad Optimal product strategies in the presence of network externalities, Information Economics and Policy, 10 (1998), 489–499.
[8] E. Haruvy and A. Prasad Optimal freeware quality in the presence of network externalities: An evolutionary game theoretical approach, J. Evolutionary Economics, 11 (2001), 231–248.
[9] E. Haruvy, A. Prasad and S. P. Sethi, Harvesting altruism in open source software development, J. Optimization Theory and Applications, 118 (2003), 381–416.
[10] E. Haruvy, A. Prasad, S. P. Sethi and R. Zhang, Optimal firm contributions to open source software: Effects of competition, compatibility and user contributions, in “Optimal Control
and Dynamic Games: Applications in Finance, Management Science, and Economics” (eds. C. Deissenberg and R.F. Hartl), Springer, (2005), 197–212.

[11] G. Hertel, S. Niedner and S. Hermann, Motivation in open source projects: An Internet-based survey of contributors to the Linux kernel, Research Policy, 32 (2003), 1159–1177.

[12] O. Kim and M. Walker, The free rider problem: Experimental evidence, Public Choice, 43 (1984), 3–24.

[13] B. Kogut and A. Metiu, Open-source software development and distributed innovation, Oxford Review of Economic Policy, 17 (2001), 248–264.

[14] K. R. Lakhani and E. von Hippel, How open source software works: ‘Free’ user-to-user assistance, Research Policy, 32 (2003), 923–943.

[15] R. Lal and R. Rao, Supermarket competition: The case of every day low pricing, Marketing Science, 16 (1997), 60–80.

[16] Y.-C. Liou, S. Schaible and J.-C. Yao, Supply chain inventory management via a Stackelberg equilibrium, J. Industrial and Management Optimization, 2 (2006), 81–94.

[17] L. Mou and J. Yong, Two-person zero-sum linear quadratic stochastic differential games by a Hilbert Space method, J. Industrial and Management Optimization, 2 (2006), 95–117.

[18] E. Muller and Y. Peles, The dynamic adjustment of optimal durability and quality, Int. J. Industrial Organization, 6 (1988), 499–507.

[19] J. A. Nelder and R. Mead, A simplex method for function minimization, Comput. J., 7 (1965), 308–313.

[20] E. S. Raymond, “The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary,” O’Reilly, Sebastopol, CA, 2001.

[21] A. Schiff, The economics of open source software: A review of the early literature, Review of Network Economics, 1 (2002), 66–74.

[22] S. P. Sethi and G. L. Thompson, “Optimal Control Theory: Applications to Management Science and Economics,” Applications to management science and economics, 2nd edition, Kluwer Academic Publishers, Boston, MA, 2000.

[23] H. Varian, “Microeconomic Analysis, 3E,” W. W. Norton and Co., New York, 1992.

[24] D. Vila, A. Martel and R. Beauregard Taking market forces into account in the design of production distribution networks: A positioning by anticipation approach, J. Industrial and Management Optimization, 3 (2007), 29–50.

Received May 2007; 1st revision August 2007; 2nd revision September 2007.

E-mail address: eharuvy@utdallas.edu
E-mail address: aprasad@utdallas.edu
E-mail address: sethi@utdallas.edu
E-mail address: zhangrong@cqu.edu.cn