Firefly-inspired Synchronization for Improved Dynamic Pricing in Online Markets

Janyl Jumadinova, Prithviraj Dasgupta
Computer Science Department
University of Nebraska, Omaha, NE 68182, USA
E-mail: {pdasgupta,jjumadinova}@mail.unomaha.edu

Abstract

We consider the problem of dynamic pricing by sellers in an online market economy using software agents called pricebots. In previous research on dynamic pricing algorithms, each seller's pricebot employs either heuristics-based or learning-based techniques to determine and update the profit maximizing price for itself at certain intervals in response to changes in market dynamics. In these dynamic pricing techniques, each seller's pricebot uses only its private information such as past prices and profits to update its price in successive intervals. In this paper, we posit that the profits obtained by a pricebot can be improved if each pricebot incorporates its competitors' pricing information along with its private price and profit information in its price-update calculations. However, incorporating competitors' pricing information accurately into a pricebot's dynamic pricing algorithm is a challenging problem because competing sellers(pricebots) update their prices asynchronously and by an amount determined by each seller's private pricing strategy. Our contribution in this paper is a novel dynamic pricing algorithm that uses a distributed synchronization model observed in nature to align each seller's price with its competitors' prices. Our analytical and simulation results show that a combination of a heuristics-based pricing mechanism that uses only a seller's private information, and, the synchronization-based mechanism that aligns its prices with its competitors, enables a seller's pricebot improve its profits by as much as 78% *as compared to previous dynamic pricing algorithms.*

1 Introduction

Over the past few years, self-adaptive technologies have emerged as a simple and efficient means for designing autonomous behavior in large-scale, distributed systems comprising several initially uncoordinated units. In this paper, we consider such a distributed scenario consisting of an online market economy with uncoordinated, competing sellers facing an unknown number of buyers having unknown and temporally varying product preferences. The problem we consider here is to provide each seller with an appropriate mechanism to improve its profit (or utility) by updating the price(s) it charges for the products it sells, in a distributed manner. The pricing decision facing a seller in this scenario is referred to as the *dynamic pricing problem*. It is solved using a computational mechanism called the *dynamic pricing algorithm* which is usually implemented using an automated software agent associated with the seller, called the seller's *pricebot*.

Currently, several online merchants employ dynamic pricing algorithms to improve their profits in response to changes in buyer demand for the items they sell. However, most of the existing dynamic pricing algorithms either ignore [2, 3, 4] or naively assume[6, 9] the effects of pricing strategies of competing sellers into the dynamic pricing mechanism. Online buyers regularly use comparison shopping services[12, 14] to determine the prices offered by all sellers in the market and are well-informed about the seller that is making the best offer for the item they are interested in. In such a scenario, if a seller does not consider and possibly match the attractive offers made by its competitors, it is likely to lose significant revenue to competitors that make more attractive offers to buyers. Therefore, it makes sense to investigate dynamic pricing mechanisms that enable a seller to obtain more profits by considering the prices from other sellers in the market and aligning its own price with those of its competitors, to make an attractive offer to buyers.

The problem of aligning a seller's price with that of other sellers in an online market is challenging because every seller in the market changes its prices dynamically and asynchronously. Each seller also uses its private pricing strategy to update its price, which is not revealed to other sellers. To address these challenges, we describe an emergent synchronization- based mechanism inspired by nature, which enables a seller's pricebot to align its prices with its competitors without incurring significant computation overhead. We analyze the performance of our emergent price

synchronization technique and show that it can enable a seller to improve its profits. Our simulation results within a simulated market economy using consumer data obtained from online seller ratings Website[15] show that our synchronization based dynamic pricing algorithm outperforms comparable dynamic pricing algorithms by $10 - 78\%$ in terms of the profits obtained by sellers using the synchronized pricing strategy.

2 Related Work

Over the past decade, dynamic pricing techniques have gained in prominence as several online merchants dynamically adjust their prices in response to consumer demand and competitors' pricing. Among the earliest agent-based dynamic pricing mechanisms, Kephart and Greenwald [6, 9] analyzed dynamic pricing as a Nash equilibrium using Varian's[16] model of sales and proposed several dynamic pricing strategies including Game Theoretic(GT), Myopic(MY), Derivative Following(DF) and Q-learner. However, the first two strategies have practical shortcomings as they require a pricebot to have *a priori* and accurate knowledge of market parameters such as number of buyers, buyer demand and competitors' prices. These parameters might be difficult, if not impossible to know accurately in any real-life market. In the derivative following(DF) and Q-learner strategies, each seller uses a machine learning technique to update the price of a product at certain intervals based on previous transactions of the seller itself with different buyers. DiMicco[4] describes another dynamic pricing algorithm called goal-derivative pricing where each seller considers the past demand of its items from buyers, the inventory available with it, and the number of its remaining days in the market to update its price. Our previous work[8] uses a preference elicitation-based technique for the pricebot's dynamic pricing algorithm when each product is differentiated along multiple attributes. However, in contrast to the work described in this paper, most of these dynamic pricing algorithms do not incorporate the acquisition of competitors' pricing information by each seller and then using the competitors' price to calculate its own updated price more effectively.

Emergent computing based techniques have become an attractive technique for designing large-scale, distributed systems with loosely coupled components. The emergent synchronization observed in firefly flashes was expressed using mathematical models by Mirollo-Strogatz [11] and Ermentrout[5]. Recently, these models has been adapted for synchronization in wireless sensor networks [10] and p2p networks[1, 19]. To the best of our knowledge, emergent synchronization has not been previously applied to the decentralized price synchronization in online economies. Complementary to the synchronization-based approach described in this paper, machine learning techniques based on Bayesian learning[13] and reinforcement learning[18] have also been used (although not for dynamic pricing in online markets) to enable an agent determine an optimal response policy to a non-stationary opponent.

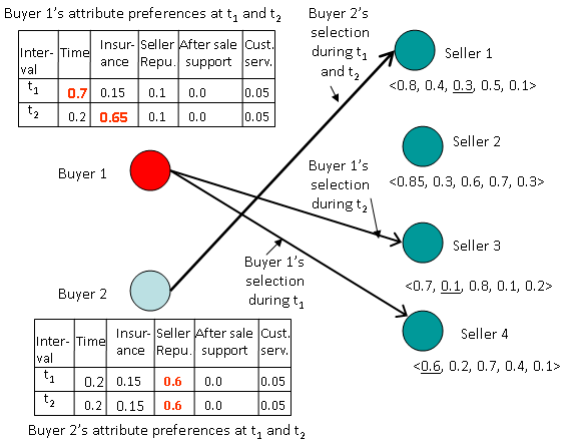

Figure 1. A market showing two buyers and four sellers. The tuple $< p_{a_1}, p_{a_2}, p_{a_3}, p_{a_4} >$ below each seller denotes the normalized price offered by each seller on the different product attributes. The preferred attribute of buyer 1 changes during interval t_2.

3 Market Model and Operation

Internet economies involve complex interactions between several buyers, sellers and possibly brokers that facilitate trading. We have made certain simplifying assumptions of an online economy to simplify analysis while retaining the essential features of the market. Following the information economy model of Kephart and Greenwald [9], we consider a market that consists of a set of S sellers who compete to provide a set of B buyers with a single homogeneous product. The number of buyers in the market is several degrees greater than the number of sellers, i.e., $\mid B \mid \gg \mid S \mid$. Each seller behaves as a profit maximizer and has a sufficient supply of the product to last the lifetime of the buyers. Buyers come back to the market repeatedly to re-purchase the product. A product is differentiated by buyers and sellers on multiple attributes such as expedited delivery time, insurance coverage for delivery, seller reputation, after-sales support, availability of customer service while purchasing the product, etc. Here, we assume that each product has a set of A different attributes, and, the buyers and the sellers in the market are aware of this set of product attributes.

Every buyer in the market selects one of the product's attribute as its preferred attribute. In online markets, buyers also change their preferred attribute for a product dynamically [17]. For example, when buyers purchase prod-

ucts under time constraints, they prefer the 'delivery- time' attribute of the product. On the other hand, when buyers do not consider time as a critical factor, seller reputation or after-sales support are possible attributes that determine their purchase decision. The choice of the preferred product attribute of a buyer can also be affected by exogenous factors such as time of the year, previous purchase experiences, etc.[17]. To model this, we assume that every buyer in our market has one of the product's attributes as its preferred attribute and the preferred attribute of a buyer can vary with time. Sellers are unaware of buyers' preferred attributes and the temporal variation of the preferred attributes for each buyer. A seller offers a slightly different price for the product along each product attribute $a_i \in A$. To improve its profits in a market with multiple sellers offering the same product, each seller has to dynamically adjust its posted prices along the different product attributes so that it can continue to offer a competitive price and attract buyers. When a buyer requests a price quote, the buyer's preferred attribute is not known to a seller. Therefore, the objective of a profit maximizing seller is to determine the buyer's preferred attribute in response to the buyer's quote request. The seller can then calculate a competitive price of the product along the buyer's preferred attribute and make an attractive offer to the buyer.

An illustration of the operation of our market is shown in Figure 1. A seller j enters the market with an initial price $p^0_{a_i,j}$ for a unit of the product under attribute a_i. Each seller has a unit production cost p_{co} below which it is not willing to sell the product. A buyer wishing to purchase a product first requests a price quote from the sellers. We assume that buyers use comparison shopping services [14] to discover sellers and are aware of all the sellers in the market. Each seller j receiving a buyer's quote request responds with a price vector $\bar{P}^t_j =< p^t_{a_i,j} >$, where $p^t_{a_i,j}$ represents the price charged by seller j during interval t along product attribute a_i. This price is updated by the seller's pricebot at intervals τ_j using a dynamic pricing algorithm. Different sellers update their product prices asynchronously and each seller uses its own pricing strategy. When a buyer that had made quote requests receives the offers from different sellers, it compares the offers made by the different sellers. Each buyer b has a reservation price for the product p_{r,b,a_i} along attribute a_i, above which it is not willing to buy the product. The utility of the product to a buyer b from seller j along attribute a_i is given by $U_{b,a_i,j} = p_{r,b,a_i} - p^t_{a_i,j}$, where $p^t_{a_i,j}$ is the posted price of the product offered by seller j along attribute a_i during interval t. The purchase decision is made by buyer b by comparing the utilities from the different sellers along the preferred attribute a_i of the product, and, selecting seller S_k given by $S_k = \arg_S \max U_{b,a_i,S}$. In the case when more than one seller offers the same lowest price along the buyer's preferred attribute, one of the sellers offering the highest utility is chosen randomly by the buyer. Buyer b then pays seller S_k the posted price of the product and the seller delivers the product. Payment and product delivery are not discussed any further here as we concentrate on seller's preferred attribute prediction and dynamic pricing strategies.

4 Emergent Price Synchronization

Ω	Natural frequency of emitting a flash
Δ	Natural cycle length for flashing, $\Delta = \frac{1}{\Omega}$
$\Omega_u(\Omega_l)$	Min.(Max.) frequency of a flash
$\delta_u(\delta_l)$	Max.(Min.) cycle length of a flash, $\delta_u(\delta_l) = \frac{1}{\Omega_u(\Omega_l)}$
ϵ	Convergence limit of synchronization between multiple flashes
ϕ	Phase of the flashing signal

Table 1. Parameters used by Ermentrout Model

The main problem we consider in this paper is to investigate mechanisms that can be employed in a pricebot's dynamic pricing mechanism to improve its sellers profits. We posit that beyond using learning-based mechanisms based on analysis of historical sales information[4, 6, 8], a seller can improve its profits by observing its competitors' prices and positioning its own price strategically with respect to its competitors. However, obtaining accurate information about prices charged by other competing sellers is a challenging problem. A seller can indeed send a quote request to all other competing sellers and receive a price quote from each of them. However, the seller obtaining the prices is unaware of it competitors' pricing strategies, and, consequently, is unable to determine when and more importantly by how much, the competing sellers will update their prices. As a result, the price information obtained from the competing sellers can be rendered stale and unusable if appropriate 'lookahead' mechanisms are not used. We envisage that a seller obtaining the price information from its competitors could estimate the price to be adapted by its competitors if it is able to synchronize its price with its competitors using a synchronization model. Because every seller in the market updates the price it charges dynamically at certain intervals, a pair of sellers could synchronize their prices with each other if they match each others prices *and* also match each others' price-changes (increment or decrement) subsequently. Matching two sellers' prices is trivial because each seller can obtain the identity of all its competitors through a comparison shopping service [12] and then request a quote from each of its competitors. However, matching each other's price *changes* is challenging because each seller's price change is determined by that seller's private pricing strategy which is neither revealed nor can be estimated trivially by the other seller. Therefore, we adapt an

emergent synchronization model to align the price changes of sellers and thereby synchronize their prices, as described below.

Our emergent synchronization mechanism is based on Ermentrout's synchronization model[5] used by a set of unsynchronized oscillators, such as a group of fireflies, to align their flashing frequencies and coordinate their flashes in a distributed and emergent manner. The Ermentrout model is expressed by the parameters given in Table 1. Each firefly is modeled as an oscillator with a variable frequency Ω. During each cycle length of the duration $\Delta = \frac{1}{\Omega}$, the phase of the oscillator increases from 0 (or the initial phase) towards 1. When the phase reaches 1, the firefly emits a signal in the form of a flash. Other fireflies observe this flash and adjust their oscillation frequencies so that all fireflies converge to the same oscillation frequency. The fireflies achieve this frequency synchronization using the following equation:

$$\omega' = \omega + \epsilon(\Omega - \omega) + g^{+}(\phi)(\Omega_l - \omega) + g^{-}(\phi)(\Omega_u - \omega) \quad (1)$$

where, $g^{+}(\phi) = max(\frac{sin\,2\pi\phi}{2\pi}, 0)$ and $g^{-}(\phi) = -min(\frac{sin\,2\pi\phi}{2\pi}, 0)$.

To adapt this model to our dynamic pricing domain, we assume that each seller(pricebot) synchronizes with other sellers(pricebots), the amount by which it changes its price, or its *price step*, at the end of each interval. We also assume that a pricebot using price synchronization has a hypothetical oscillator within it. The frequency of this oscillator corresponds to the *price step* of the pricebot. The oscillator's phase varies from 0 to 1. When the phase reaches 1, the pricebot emits a signal (or a flash) to the market indicating that it is ready to change its price by the amount given by its current price step and resets its phase to 0. Other sellers can perceive the signal emitted by the pricebot and use use the Ermentrout model given in Equation 1 to adapt their own price steps. When all sellers achieve synchronization, every seller has the same price-step. Therefore, without knowing each others' pricing strategies explicitly, each seller can adjust its prices in tandem with its competitors.

The synchronized pricing algorithm used by a pricebot is shown in Figure 2. For clarity, we have illustrated the algorithm and its analyses only along one product attribute a_i. The pricebot uses the same algorithm for adjusting the price along each product attribute. As shown in Figure 2, as long as a seller participates in the market, its pricebot updates its phase ϕ towards 1 (line 2). If ϕ reaches 1, it gets reset to 0 and the pricebot sends a signal (or a flash) to the market, which is forwarded to other sellers via the market (lines 3-5). The pricebot then obtains the price being charged by other sellers in the market and determines the minimum market price for the product from any seller (line 6). If the price charged by the pricebot's seller does not correspond to the minimum market price, the pricebot

function *synchronizePrice*() returns void
  variables: double ϕ; //phase
    int t; // normalizing time const.
    double delayTime; // update interval for ϕ;
    double ω; // current price step
    double Ω; // default price step, set to $0.5 \times (\Omega_u + \Omega_l)$
    double Ω_u; // upper bound on price step
    double Ω_l; // lower bound on price step
    double $minMarketPrice^t_{a_i}, p^t_{a_i}, syncPrice^t_{a_i}$;
      // min. price of other sellers in the market, current
      // price and synchronized-price charged by seller
      // during interval t along attribute a_i
1. while ($aliveInMarket$)
2. $\phi \leftarrow \phi + \omega \times \frac{delayTime}{t}$; // update phase
3. if ($\phi = 1$)
4. $\phi \leftarrow 0$; // reset the phase
5. sendFlash(); // send flash to all other sellers;
    // Next check and update synchronized price
6. $minMarketPrice^t_{a_i} \leftarrow$ getMinMarketPrice(a_i);
7. if ($minMarketPrice^t_{a_i} != p^t_{a_i}$)
8. $syncPrice^t_{a_i} \leftarrow minMarketPrice^t_{a_i} - \omega \times \gamma$;
9. if (gotFlashMessage() = true)
    // Update the step size for price changes
    // when a flash is received from another seller
10. $g^{+}(\phi) \leftarrow max(\frac{sin\pi\phi}{2\pi}, 0)$;
11. $g^{-}(\phi) \leftarrow -min(\frac{sin\pi\phi}{2\pi}, 0)$;
12. $\phi \leftarrow \phi + \epsilon(\Omega - \omega) + g^{+}(\phi)(\Omega_l - \omega) + g^{-}(\phi)(\Omega_u - \omega)$;

Figure 2. Algorithm used by a pricebot to synchronize its price step with other pricebots.

decreases its price by the current price step (ω) (lines 7-8). The constant γ is used to normalize the step size ω to the allowable range of price step sizes. Finally, in lines (9-12), when the seller perceives a signal (or flash) from the market, it adjusts its phase ϕ, using the Ermentrout model given in Equation 1. Next, we discuss some mathematical analyses of our synchronized pricing model.

Proposition 1. *The synchronized pricing algorithm given in Figure 2 allows initially unsynchronized pricebots to synchronize their price steps with each other in equilibrium.*

Proof. Let ω^t_1 and ω^t_2 denote the unsynchronized price steps of two pricebots during interval t. Let us also assume without loss of generality that the first pricebot's time step, ω^t_1, is lower than the default price step Ω, while the second pricebot's price step, ω^t_2, is above the default price step Ω. That is, $\Omega_l < \omega^t_1 < \Omega$ and $\Omega < \omega^t_2 < \Omega_u$. Let us also assume that the first pricebot's phase $\phi^t_1 < 0.5$ and the second pricebot's phase $\phi^t_2 > 0.5$. Note that because of these relative values of ω^t_1, ω^t_2, ϕ^t_1 and ϕ^t_2, the term with $g^{+}(\phi)$ is

negative and the term $g^-(\phi)$ vanishes for the first pricebot, while the term with ϵ is negative and the term $g^+(\phi)$ vanishes for the second pricebot. Using the Ermentrout model from Equation 1, we then can write the price steps for these two pricebots during the following interval $(t+1)$ as:

$$\omega_1^{t+1} = \omega_1^t + \epsilon(\Omega - \omega_1^t) + g^+(\phi)(\Omega_l - \omega_1^t) \quad (2)$$

$$\omega_2^{t+1} = \omega_2^t + \epsilon(\Omega - \omega_2^t) + g^-(\phi)(\Omega_u - \omega_2^t) \quad (3)$$

Following the parameters used in Ermentrout's model [5], $\epsilon > g^{\pm}(\phi)$. Also, following the definition of $g^{\pm}(\phi), 0 < g^{\pm}(\phi) < \frac{1}{2\,\pi}$. Taken together, the values of ϵ and $g^{\pm}(\phi)$ imply that $\omega_1^{t+1} > \omega_1^t$ while $\omega_2^{t+1} < \omega_2^t$. In other words, the price step of the first pricebot increases towards the default price step, while the price step of the second pricebot decreases towards the default price step. As the price step of the first pricebot increases in successive intervals, it receives the flash from the second pricebot at earlier phases than it received during interval t. Consequently, both the phase at which the first pricebot sees the second pricebot's flash and its value of $g^+(\phi)$ in Equation 2, approach zero. Similarly, because the second pricebot increases its frequency, the phase at which it sees the first pricebot's flash approaches $2\,\pi$ and its value of $g^+(\phi)$ in Equation 3 also approaches zero. In equilibrium, the $g^{\pm}(\phi)$ terms vanish both from Equations 2 and 3, while both ω_1 and ω_2 approach Ω. Therefore, both pricebots are able to synchronize their price steps with each other in equilibrium and the process converges. In general, when multiple pricebots use the Ermentrout model, the pricebots with price step values below the default price step increase their price step sizes, while those with price step values below the default price step decrease their step sizes and converge at the default price step to synchronize their price steps with each other in equilibrium. Hence proved.

Proposition 2. *Using the synchronized pricing algorithm, a pricebot increases the probability of making the best offer to buyers in the market.*

Proof. Recall from Section 3 that buyers select the pricebot(seller) that offers the lowest price for the product along the buyers' preferred product attribute. Therefore, to improve its performance(profit), a pricebot should update its price dynamically so that it charges the lowest price for the product in comparison to its competitors. Consider a scenario where a pricebot PB_i has $\mid PB_u \mid$ pricebots with prices higher than it, and another $\mid PB_l \mid$ pricebots with prices lower than it during interval $(t-1)$. Let p_i^{t-1} denote the price offered by PB_i during interval $(t-1)$. We are interested in calculating the probability that PB_i charges the minimum price during interval t, so that it can get the highest profits. For PB_i - the minimum priced pricebot during interval t, the $\mid PB_l \mid$ pricebots with prices below PB_i during interval $(t-1)$ should increase their prices during interval t by a price step that positions their price above PB_i, while the $\mid PB_u \mid$ pricebots with prices above PB_i during interval t should decrease their prices during interval t by a price step that positions their price below PB_i. Mathematically, we can represent the probability of PB_i being the minimum priced pricebot during interval t as:

$$P(\delta_{u_1}^t > \Delta_{u_1} - \delta_i^t)P(\delta_{u_1}^t < 0)P(\delta_{u_2}^t > \Delta_{u_2} - \delta_i^t)P(\delta_{u_2}^t < 0)...$$
$$...P(\delta_{l_1}^t > \Delta_{l_1} - \delta_i^t)P(\delta_{l_1}^t < 0)P(\delta_{l_2}^t > \Delta_{l_2} - \delta_i^t)P(\delta_{l_2}^t < 0)$$

where, $\delta_{u_j}^t$ is the price step taken by the j-th pricebot in PB_u during interval t, $\delta_{l_k}^t$ is the price step taken by the k-th pricebot in PB_l during interval t, Δ_{u_j} represent the difference in prices between the j-th pricebot in PB_u and PB_i during interval $(t-1)$ and Δ_{l_k} represent the difference in prices between the k-th pricebot in PB_l and PB_i during interval $(t-1)$. This expression can be rewritten as:

$$P(p_i^t = p_{min}^t) = \Pi_{j \in PB_u} P(\delta_{u_j}^t + \delta_i^t > \Delta_{u_j}) P(\delta_{u_j}^t > 0) \times$$

$$\Pi_{k \in PB_l} P(\delta_{l_k}^t - \delta_i^t > \Delta_{l_k}) P(\delta_{l_k}^t > 0)$$

Now the price steps δ_i^t, $\delta_{u_j}^t$ and $\delta_{l_k}^t$ are all drawn from the distribution of price steps given by $U[\delta_{min}, \delta_{max}$. So, the expression above can be considered as the probability of the sum (difference) of two numbers $\delta_{u_j}^t(\delta_{l_k}^t)$ and δ_i^t, drawn from the same distribution is greater than the value $\Delta_{u_j}(\Delta_{l_k})$. With this understanding, the probability of PB_i charging the minimum price among its competitors during interval t can be approximated as:

$$P(p_i^t = p_{min}^t) =$$

$$[\frac{(N - \Delta_{u_j}^t)(N - \Delta_{u_j}^t + 1) + \Delta_{u_j}^t(\Delta_{u_j}^t - 1)/2}{N^2}]^{|PB_u|} \times$$

$$P(\delta_{u_j}^t > 0) \times [\frac{(N - \Delta_{l_k}^t - 1)(N - \Delta_{l_k}^t)}{N^2}]^{|PB_l|} P(\delta_{l_k}^t > 0)$$

where $N = \delta_{max} - \delta_{min}$ is the difference between the maximum and minimum limits of the price step distribution. As shown in Proposition 1, when all pricebots use synchronized pricing, $\Delta_{u_j}^t(\Delta_{l_k}^t)$, the differences between the price steps of PB_i and that of other pricebots in $PB_u(PB_l)$ approach 0. When $\Delta_{u_j}^t(\Delta_{l_k}^t) \rightarrow 0$, the non-probability terms in the above expression approach their maximum values. Also, because all pricebots synchronize their price steps, the probability that a pricebot from $PB_u(PB_l)$ will reduce(increase) its price from the previous interval approaches 1 (i.e.,$P(\delta_{u_j}^t < 0) \rightarrow 1$ and $P(\delta_{l_k}^t > 0) \rightarrow 1$). Therefore, the highest probability of PB_i being the pricebot with the minimum price during interval t happens when pricebots use the synchronized pricing algorithm. Hence proved.

Proposition 3. *The synchronized pricing algorithm converges more accurately but less rapidly than a dynamic pricing algorithm without synchronization.*

Proof. Consider a market with two pricebots(sellers). One pricebot charges a fixed price p_F for the products it sells and does not change this price over time. The other pricebot in the market can either use a dynamic pricing algorithm without synchronization, or use the synchronized pricing algorithm. First, consider that the latter pricebot uses a dynamic pricing algorithm without synchronization. Suppose that this pricebot enters the market with an initial price of p_{DP} that is greater than p_F. To improve its profit, the dynamic pricing pricebot must reduce its price below p_F so that it can offer the best price for the product to the buyers. In other words, the dynamic pricing pricebot must reduce its price by at least $p_{DP} - p_F$ to improve its profits. To achieve this, the dynamic pricing pricebot selects its price step from the price step distribution $U[\delta_{min}, \delta_{max}]$, using a heuristics-based strategy. The expected number of intervals that the dynamic pricing pricebot will require to reduce its price below p_F is given by $\frac{p_{DP}-p_F}{E(price\ step)} = \frac{2(p_{DP}-p_F)}{(\delta_{max}-\delta_{min})}$.

Next, consider a pricebot that uses the synchronized pricing strategy and starts at the same price p_{DP}. Let us also suppose that the initial phase of this pricebot's oscillator is greater than 0.5 and it uses the Ermentrout equation in Equation 3. The synchronized pricing pricebot will be able to reduce its price below the fixed price pricebot's price p_F when it synchronizes its price step to ω. We can rewrite the equation for the Ermentrout model by rearranging the terms of Equation 3 in the following manner:

$$\omega^{t+1} = \omega^t(1 + \epsilon + g^-(\phi)) + \epsilon\Omega + g^-(\phi)\Omega_u$$

This gives a recurrence relation of the form $\omega^{t+1} = A\omega^t + B$, where A and B are constants given by $A = (1 + \epsilon + g^-(\phi))$ and $B = \epsilon\Omega + g^-(\phi)\Omega_u$. Solving the above recurrence relation, we get $\omega(n) = A\omega_0 + B\frac{A^n-1}{A-1}$, where ω_0 is the initial price step of the synchronized pricing seller. Here, n corresponds to the number of intervals taken by the synchronized pricing seller to synchronize its price step. Solving for n, we get:

$$n = log_A[\frac{\frac{p_{DP}-p_F}{\omega_0} + \frac{B}{(A-1)\omega_0}}{1 + \frac{B}{(A-1)\omega_0}}].$$

The smallest value that A can get is $(1 + \epsilon)$ (when $\phi = 0$ or 2π). Substituting this value for A, we get $n > \frac{2(p_{DP}-p_F)}{(\delta_{max}-\delta_{min})}$, the expected number of intervals required by the pricebot using dynamic pricing without synchronization to improve its profit. Therefore, a pricebot using a dynamic pricing algorithm without synchronization can outperform a seller using the synchronized pricing algorithm. Hence proved.

The inferior performance of a pricebot using the synchronized pricing algorithm only against a competing seller using a simple dynamic pricing algorithm without synchronization is also observed in the simulation results shown in Figure 5. We envisage that this problem of inferior performance by the synchronized pricing sellers can be alleviated if the coarse price adjustments of the dynamic pricing algorithm without synchronization could be combined with the accurate price step synchronization of the Ermentrout model. To address this problem, we have used a combined algorithm called the synchronized dynamic pricing (SDP) algorithm as shown in Figure 3. In the SDP algorithm, the pricebot maintains two pricing algorithms - an active pricing algorithm that it uses to update its price and calculate its actual profits, and a latent pricing algorithm that it uses only to calculate expected profits. At the end of every h intervals, the pricebot compares the profit it obtained by using its active pricing algorithm with the expected profit it would have obtained by using its latent pricing algorithm. The pricebot then selects the pricing algorithm that yielded higher profits for the next h intervals. h is called the *strategy selection interval* of the pricebot.

```
function synchronizedDynamicPricing() returns void
    variables: int h; // No. of intervals after which
        // pricing strategy performance is checked
        int t; // Current time interval
if (t < h) // initially use dyn. pricing
    strategy^{t+1} = DynamicPricing;
else if (t mod h = 0)
    if (expectedCumulativeProfit^t_{a_i} >
          currentCumulativeProfit^t_{a_i})
    // Switch strategies
    if (strategy^t == DynamicPricing)
        strategy^{t+1} = SynchronizedPricing
    else if (strategy^t == SynchronizedPricing)
        strategy^{t+1} = DynamicPricing
```

Figure 3. Algorithm used by a pricebot to select between the synchronized and dynamic pricing strategies.

5 Simulation Results

To quantify the performance of the SDP algorithm against other dynamic pricing algorithms, we have verified the performance of the SDP algorithm in a market with different number of buyers and sellers, using different dynamic pricing algorithms. For simplicity, we have reported results for one product attribute. Other product attributes show similar dynamics in terms of price and profit profiles. All prices are normalized to a range of [0,1]. All results were averaged over 10 simulation runs. The following is a list of parameters and the corresponding values we have used in our simulations:

Variation of Buyer Attribute Preferences. To simulate the temporal variation of attribute preferences by buyers, we have used preference data reported by online consumers

Parameter	Value
Number of buyers	500 or 1000
Number of sellers	3 or 5
Number of product attributes	5
Rate at which buyers send quote requests to sellers	5000 ms
Unit production cost for seller	0.1
Entry price of sellers in market	$U[p_{co}, 1.0]$
No. of intervals after which price strategy selection occurs(h)	75
Interval for price updates for seller	40 quote requests from buyers
Initial phase for pricebot's oscillator (for Ermentrout model)	$U[0, 1.0]$
Max. cycle length of oscillator(Δ_u) (max. price step of pricebot)	0.2
Min. cycle length of oscillator(Δ_l) (min. price step of pricebot)	0.01

Table 2. Parameters used in our simulation experiments

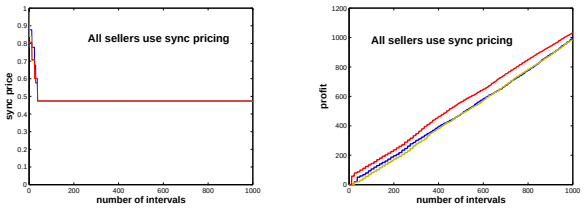

Figure 4. Price (left) and profit (right)dynamics in a market with 3 pricebots and 500 buyers, where all pricebots use synchronized pricing to update their prices.

along different product attributes obtained from an online ratings Website[15]. We have represented this data as a set of discrete probability distributions P_n from which buyers determine their attribute preferences. When a buyer enters the market, it randomly selects one of the probability distributions, $p_n \in P_n$. Each probability distribution consists of $\mid A \mid$ probabilities, $p_n =< p_{a_i} >\mid i = 1, ..., \mid A \mid$. Each p_{a_i} corresponds to a buyer's probability of selecting that attribute as its preferred attribute. To model the temporal variation in preferred attributes, each buyer changes its selected probability distribution from p_n to $p_{n'} \in p_{-n}$ at random times.

5.1 Synchronized Dynamic Pricing(SDP) Strategy Performance

For our first set of simulations, we tested that multiple pricebots using the synchronized pricing strategy converged to a single price. The results are reported for a market with 500 buyers and 3 pricebots, where all pricebots use the synchronized pricing strategy to set their prices. Figure 4 shows that after about 40 intervals, sellers using the synchronized pricing algorithm are able to synchronize their prices with each other and share profits more or less equally. However, as mentioned earlier, the synchronized pricing strategy alone performs poorly when compared to a simple dynamic pricing strategy called derivative following, as shown in Figure 5.

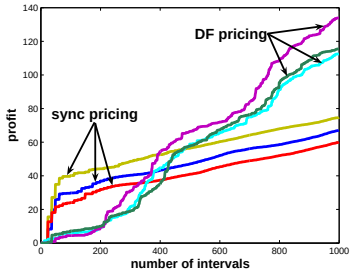

Figure 5. Profit profile of 3 sellers using synchronized pricing only, against 3 sellers using derivative following.

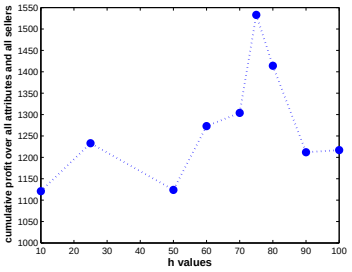

Figure 6. Cumulative profit obtained by 3 sellers for different values of the strategy selection interval(h)

For our next set of simulations we verified the impact of the strategy selection interval parameter h on the performance of the SDP pricing strategy. The results of our simulations are reported for a market with 3 sellers and 500 buyers in Figure 6. We observed that for a lower range of h values between $10 - 50$, the profits obtained by sellers are adversely affected because the sellers switch frequently between the synchronized and dynamic pricing strategies. This prevents the profits from either strategy to accrue sufficiently and the seller is forced to make a strategy switching decision with insufficient performance(profit) information from its last used strategy. On the other hand, even when a seller uses larger h values (100) its profits are lower. This is because, with larger h values, once a seller selects a strategy, it does not evaluate its performance for a considerably large number of intervals. This resilience to evaluate its performance results in the seller using a poorly performing strategy over a longer time, and, adversely af-

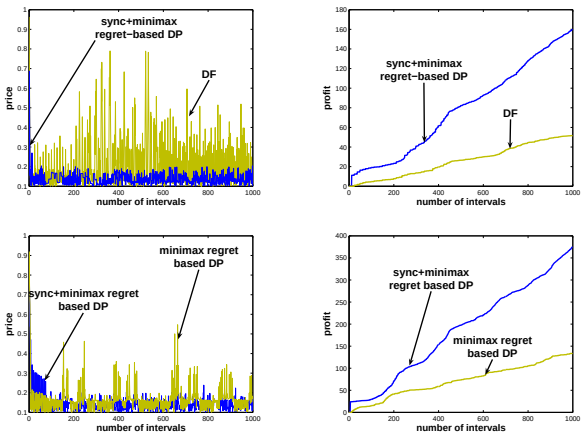

Figure 7. Market with 2 sellers and 500 buyers, SDP-MMR vs. DF (top) and SDP-MMR vs. MMR(bottom). The graphs on the left show the price profiles while the graphs on the right show the profit profiles of each seller.

Seller Strategies	Ave. Cum. Prof. (SDP-X)	Ave. Cum. Prof.$_{\text{(opponent)}}$	% Perf. Improv.
1 SDP-MMR vs. 2 DF	661.53	342.07	48.29
1 SDP-MMR vs 2 MMR	555.91	373.39	32.83
1 SDP-MMR vs 2 MY	1412.29	442.87	68.64
1 SDP-MMR vs 2 GT	1089.25	287.79	73.58
2 SDP-MMR vs 1 DF	451.53	302.57	32.99
2 SDP-MMR vs 1 MMR	1137.92	979.38	13.93
2 SDP-MMR vs 1 MY	644.69	346.93	46.19
2 SDP-MMR vs 1 GD	257.68	237.37	7.88
1 SDP-DF vs 2 DF	577.1	422.71	26.75
2 SDP-DF vs 1 DF	488.05	397.52	18.55
1 SDP-DF vs 2 MMR	672.97	461.01	31.49
2 SDP-df vs 1 MMR	467.64	368.82	21.13

Table 3. Average cumulative profit comparisons for different pricing strategies in the market with 500 buyers and 3 sellers

fects the seller's performance(profits). As shown in Figure 6, the highest profits of the sellers are obtained for a value of $h = 75$. This is the value of h we have used for the rest of our simulations.

Next, we compared the performance of the SDP strategy with existing dynamic pricing algorithms. The list of comparison strategies is as follows: **1) Fixed Pricing(FP).** This is the simplest strategy where a seller does not change the price of the item. **2) Myoptimal Pricing Strategy(MY).** Myoptimal pricing (MY) uses information about the buyer population such as the number of buyers and their reservation prices, as well as the number of competing sellers and their prices [6, 9]. The MY seller searches for the best price that maximizes its expected profit. The best price is guaranteed to be either equal to the buyer's valuation, or some small δ value below another seller's price. In our simulations, we used $\delta = 0.002$ as the value by which one sellers may undercut another. **3) Game-Theoretic Pricing Strategy(GT).** The game-theoretic(GT) pricing strategy also requires full information about the buyer population and the number of competing sellers, but it does not use any historical observations [6]. The seller computes a distribution $f(p)$ from the buyer parameters and the number of sellers, and then chooses a random price from this distribution. **4) Goal Directed Strategy(GD).** In the goal-directed(GD) pricing strategy described by DiMicco *et al.*[4], a seller calculates the average number of a units of the product that it should sell per interval so that it can clear the inventory by its last interval in the market. It then observes the number of units it is able to sell during the current interval. If the actual number of units sold is above (below) the expected clearance value, the seller responds by raising(lowering) the price of the product for the next interval. **5) Derivative Follower Pricing Strategy(DF).** In the derivative follower (DF) strategy, a seller determines the price for the next interval based on the profits obtained from the price charged during the current interval. A derivative-follower seller j uses the following equation to update its price along attribute a_i: $p^{t+1}_{a_i,j} = p^t_{a_i,j} + sign(\pi^t_{a_i,j} - \pi^{t-1}_{a_i,j}) * \delta$, where $\delta \in U[0.1, 0.2]$. **6) Minimax Regret Strategy (MMR)** Minimax regret(MMR) is a preference elicitation based technique where each seller uses a minimax regret-based technique to calculate buyers' preferred attribute as well as to update its own price based on competitor prices. A seller using the MMR strategy observes a buyer's purchase decision and uses this decision to select a preferred attribute for the buyer using the minimax regret criterion. After determining the buyer's preferred attribute, a seller updates the posted price of the item along the different attributes using a weighted average of its last k prices ($k = 10$ for our simulations), and a regret-based price. Further details of the MMR algorithm are given in [8]. Because the SDP strategy uses a combination of synchronized pricing with dynamic pricing, we have used the notation SDP-X in our comparisons, where X denotes the dynamic pricing strategy being used within with SDP.

Our first scenario consists of 500 buyers and 2 sellers, where one seller uses the SDP-X algorithm while the other uses another dynamic pricing algorithm without synchronization. Figure 7(top) shows the price and profit profiles of a seller using the SDP-MMR strategy vs. another seller using DF strategy only. The seller using SDP-MMR strategy is able to adjust its prices more strategically compared to the seller using DF only. As a result, the SDP-MMR

Seller Strategies	Ave. Cum. Prof. (SDP-X)	Ave. Cum. Prof.$_{(opponent)}$	% Perf. Improv.
1 SDP-MMR vs 4 DF	347.89	230.84	50.71
2 SDP-MMR vs 3 DF	274.46	284.2	−3.55
3 SDP-MMR vs 2 DF	237.4	237.91	−0.21
4 SDP-MMR vs 1 DF	236.08	26.89	3.89
1 SDP-MMR vs 4 MMR	773.58	321.96	58.38
2 SDP-MMR vs 3 MMR	451.92	355.67	21.29
3 SDP-MMR vs 2 MMR	538.24	466.68	13.29
4 SDP-MMR vs 1 MMR	398.15	384.8	3.36
2 SDP-MMR vs 3 GD	139.75	127.45	8.85
2 SDP-MMR vs 3 GT	361.33	253.91	29.73
2 SDP-MMR vs 3 MY	506.88	110.98	78.11
1 SDP-DF vs 4 DF	264.42	221.73	16.14
2 SDP-DF vs 3 DF	224.57	201.92	10.09
3 SDP-DF vs 2 DF	211.74	189.71	10.4
4 SDP-DF vs 1 DF	202.79	152.85	24.63
1 SDP-DF vs 4 MMR	316.98	208.67	34.17
2 SDP-DF vs 3 MMR	190.9	187.64	1.71
3 SDP-DF vs 2 MMR	172.73	178.12	−3.12
4 SDP-DF vs 1 MMR	188.79	194.25	−2.89

Table 4. Average cumulative profit comparisons for different pricing strategies in the market with 1000 buyers and 5 sellers

seller receives 76 % of the total profits vs. 24 % by the DF-only seller. Figure 7(bottom) illustrates a similar scenario with the two sellers using SDP-MMR and MMR-only respectively. We observe that the seller using SDP-MMR once again outperforms the seller with MMR-only and gets 73 % of the total market profit share. Clearly, in both cases, the ability of the SDP strategy to select between competition through dynamic pricing, and, price-matching through synchronization with its competitors, increases its profit margin by as much as three times. Further simulations when one of the sellers used any of the other dynamic pricing algorithms without synchronization yielded higher profit shares for the seller using the SDP-X algorithm.

Next we compare the performance of sellers using SDP-X strategy against multiple sellers using different dynamic pricing strategies. Table 3 summarizes the average cumulative profit comparisons in the market with 500 buyers and 3 sellers. The first column indicates the pricing strategies used by the sellers in the market, the second column shows the average cumulative profit of the seller using the SDP-X strategy, the third column shows the average cumulative profit of the seller using the dynamic pricing without synchronization, and the last column indicates the amount of performance(profit) improvement of the SDP-X strategy over the dynamic pricing only strategy, expressed as a percentage. In every case, we see that the combination of synchronized pricing with dynamic pricing in the SDP-X strategy outperforms the dynamic pricing without synchronization strategy. Among all of the compared strategies a seller uses, the GD strategy does the least worst (by about 7%) as compared to the SDP-MMR strategy. This can be attributed to the fact that GD is the only strategy that considers future expected sales assuming a finite horizon while updating the price. Although accurate information about future inventory might be a valid assumption in the setting of [4] for which GD was developed, this information might not be available in several online markets. In contrast, the SDP-MMR strategy is able to perform comparably with GD without knowing any such future information.

Table 4 summarizes our simulations for a similar comparison of SDP-X strategy vs. strategies using dynamic pricing without synchronization in a market with 5 sellers and 1000 buyers. Interestingly, we observe that in some cases the SDP-X strategy performs worse, although by a small amount (3.55% or lesser). An analysis of the results indicates that the poorer performance of SDP-X happens only in scenarios where multiple sellers using an SDP-X strategy compete with sellers using a dynamic pricing strategy (without synchronization) other than X. For example, SDP-MMR-s perform worse than DF-only in some cases and SDP-DF performs worse than MMR-only in some cases. From the results, we also observe that when 1 SDP-X seller competes with any number of sellers using non-synchronized pricing strategies, the SDP-X sellers always perform better. This observation leads us to the explanation that although a single seller using SDP-X outperforms multiple sellers using non-synchronized dynamic pricing strategies, when multiple sellers simultaneously use SDP-X, they compete with among themselves to reduce each others' profits and consequently end up performing poorer than the non-synchronized pricing strategy opponent. The better performance of multiple SDP-X sellers competing with other non-synchronized sellers using strategy X (e.g., SDP-MMR vs.MMR-only, or, SDP-DF vs. DF-only) also indicates that the poorer performance of SDP-X occurs only when the non-synchronized competitors employ a dynamic pricing strategy other than X (e.g SDP-MMR vs. DF or SDP-DF vs MMR). Another aspect of our SDP-X strategy vis-a-vis MY strategy is that, while the latter requires information about both the buyer population (number of buyers and reservation prices), SDP-X can outperform MY by as much as 78% without requiring any such knowledge about the buyer population. Overall, the results in Table 4 indicate that our SDP-X dynamic pricing algorithm is able to perform significantly better than non-synchronized pricing strategies in most settings, and performs slightly poorer than a few non-synchronized dynamic pricing strategies in certain specific cases.

Table 5 shows the cumulative profit comparisons in two

SDP-MMR	MMR	GD	DF	FP
354.16	262.64	256.8	251.27	78.57
SDP-MMR	MMR	DF	MY	GT
575.76	357.49	350.14	257.64	105.24

Table 5. Cumulative profit comparisons in two scenarios with 1000 buyers and 5 sellers. Each seller adopts a different pricing strategy.

scenarios with 1000 buyers and 5 sellers, where each seller adopts a different pricing strategy. In the first scenario, the seller using SDP-MMR strategy ends up with the largest market share, 29.4%, while the seller using MMR-only earns 21.8%, the GD seller earns 21.4%, the DF seller earns 20.9%, and the FP seller earns only 6.5% of the total profits. In the second scenario, the seller using the SPD-MMR strategy earns 35% of the profits, while the sellers using the MMR-only, DF, MY and GT end up with 21.7%, 21.3%, 15.6%, and 6.4% of the profits respectively. These results once again indicate that sellers using the synchronized pricing based SDP-X strategy outperforms other sellers using non-synchronized dynamic pricing by significant margins.

6 Conclusions and Future Work

In this paper, we have described a novel algorithm called synchronized dynamic pricing (SDP) that enables pricebots determine a profit maximizing price in response to changes in prices of other competitors as well as to changes in buyer demand and preferences. Our simulation results show that a pricebot using the SDP algorithm always outperforms another competing pricebot using a different dynamic pricing algorithm. With multiple pricebots, the SDP algorithm can still perform significantly better than existing dynamic pricing algorithms. However, when multiple pricebots use the SDP algorithm against each other, they compete with each other for profits and perform only slightly poorly (-0.21% to -3.55% in a few cases when compared to pricebots not using the SDP algorithm.

This is our first step in exploring algorithms that combine emergent synchronization with pricing algorithms. In the future, we plan to investigate several directions extending this model. A very relevant problem in online markets is the revelation of inaccurate price information by sellers to their competitors. A related problem is the presence of noisy communication in the market that results in some sellers not being able to send or receive signals(flashes) required for synchronizing their price steps. We plan to investigate techniques to address both of these problems. There are several dimensions to the dynamic pricing problem that are yet to be investigated and can be possibly modeled and solved using self-adaptive techniques. These problems include buyers sharing information about sellers with each other and buyers colluding with each other to share profits. We plan to investigate these problems in the future.

References

[1] O. Babaoglu, T. Binci, M. Jelasity and A. Montresor, "Firefly-inspired Heartbeat Synchronization in Overlay Networks," Proc. SASO 2007, pp. 77-86.

[2] P. Dasgupta and R. Das, "Dynamic Pricing with Limited Competitor Information in a Multi-Agent Economy," Proc. 7th Intl. Conf. on Cooperative Information Systems (CoopIS), Israel, 2000, pp. 299-310.

[3] P. Dasgupta and Y. Hashimoto, "Multi-Attribute Dynamic Pricing for Online Markets Using Intelligent Agents," Proc. 3rd Intl. Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS), New York, NY, 2004, pp. 277-284.

[4] J. DiMicco, A. Greenwald and P. Maes, "Dynamic Pricing Strategies under a Finite Time Horizon," Proc. 3rd ACM Conf. E-Commerce(ACM-EC), Tampa, , 2001, pp. 95-104.

[5] B. Ermentrout, "An adaptive model for synchrony in the firefly *Pteroptyx malaccae*," J. Math. Biol. , vol. 29, 1991, pp. 571-585.

[6] A. Greenwald, J. Kephart and G. Tesauro, "Strategic Pricbot Dynamics," Proc. 1st ACM Conf. on Electronic Commerce (ACM-EC), Denver, CO, 1999, pp. 58-67.

[7] A. Greenwald and J. Kephart, "Probabilistic pricebots," Proc. 5th Intl. Conf. on Autonomous Agents, Montreal, Canada, 2001, pp. 560-567.

[8] J. Jumadinova and P. Dasgupta, "Multi-attribute Regret-based Dynamic Pricing,", Proc. 10th Agent Mediated E-Commerce Workshop (AMEC), Estoril, Portugal, May 2008.

[9] J. Kephart, J. Hanson and A. Greenwald, "Dynamic pricing by software agents," Computer Networks, vol. 32, no. 6, Elsevier, 2000, pp. 731-752.

[10] D. Lucarelli and I. Wang, "Decentralized synchronization protocols with nearest neighbor communication," Proc. 2nd Intl. Conf. Embedded Networked Sensor Systems(SenSys), New York, NY, 2004, pp. 62-68.

[11] R. Mirollo and S. Strogatz, "Synchronization of pulse-coupled biological oscillators," SIAM J. of Applied Mathematics, vol. 50, no. 6, 1990, pp. 1645-1662.

[12] My Simon, http://www.mysimon.com

[13] V. Narayanan and N. Jennings, "Learning to negotiate optimally in non-stationary environments," Lecture Notes in Comp. Sci., vol. 4146, (Proc. 10th Cooperative Information Agents (CIA) Workshop), Springer, 2006, pp. 288-300.

[14] PriceGrabber, http://www.pricegrabber.com

[15] Reseller Ratings, http://www.resellerratings.com

[16] H. Varian, "A model of sales," Amer.. Econ. Review, Papers and Proc., vol. 70, no. 4, 1980, pp. 651-659.

[17] Y. Wei, "Future Orientation, Chronological Age and Product Attributes Preference," PhD Thesis, Georgia State University, 2007.

[18] M. Weinberg and J. Rosenchein, "Best response multiagent learning in non-stationary environments," Proc. AAMAS, New York, NY, 2004, pp. 506-513.

[19] G. Werner-Allen, . G. Tewari, A. Patel, M. Welsh and R. Nagpal, " Firefly inspired sensor network synchronicity with realistic radio effects," Proc. 3rd Intl. Conf. Embedded Networked Sensor Systems, New York, NY, 2005, pp. 142-153.