Particle Swarm Optimization in Dynamic Pricing

Patrick B. Mullen
Computer Science Department
Brigham Young University
mullenp@cs.byu.edu

Christopher K. Monson
Computer Science Department
Brigham Young University
c@cs.byu.edu

Kevin D. Seppi
Computer Science Department
Brigham Young University
kseppi@cs.byu.edu

***Abstract*— Dynamic pricing is a real-time machine learning problem with scarce prior data and a concrete learning cost. While the Kalman Filter can be employed to track hidden demand parameters and extensions to it can facilitate exploration for faster learning, the exploratory nature of Particle Swarm Optimization makes it a natural choice for the dynamic pricing problem. We compare both the Kalman Filter and existing particle swarm adaptations for dynamic and/or noisy environments with a novel approach that time-decays each particle's previous best value; this new strategy provides more graceful and effective transitions between exploitation and exploration, a necessity in the dynamic and noisy environments inherent to the dynamic pricing problem.**

I. Introduction

The popularity of the Internet somecommand as a medium for commerce creates unique opportunities to alter prices rapidly in response to changes in markets. While not responsible for its existence, this fluid medium lends increased importance to an interesting real-time learning problem known as *dynamic pricing*, "...the problem of setting prices dynamically to maximize expected revenues in a finite horizon model in which the demand parameters are unknown. [1]" In dynamic pricing, training examples are available in the form of one set of price-revenue pairs per time period.

In addition to being a real-time learning problem, dynamic pricing also has measurable associated costs that must be taken into account when determining an appropriate balance between exploration and exploitation. A pure exploitation strategy may produce good results for a time, but the dynamic nature of the environment may eventually cause its performance to degrade. Alternatively, exploration is likely to provide useful information about the true nature of the market, information that may facilitate more effective future exploitation. Because exploration and exploitation are indistinguishable from the perspective of a buyer (as both involve setting a price), liberal exploration carries with it the risk of *opportunity cost*, revenue lost at the current time period because the price was set away from the optimum.

One frequently applied approach to the problem is to model demand using a parametric model. As the true market demand is not known *a priori*, the parameters of the model are considered to be hidden; a price is chosen and a revenue observed, but the nature of the demand at every price point is generally unknown. Such a model lends itself well to Bayesian reasoning, making a case for the use of the Kalman Filter to track its hidden parameters [2].

The Kalman Filter is provided a prior (and generally subjective) belief about the hidden parameters, which it combines with observations to adjust and track its understanding of those parameters. This information is typically used to set a new price that optimizes the expected revenue during the next time period. After setting the new price, the actual revenue is observed and the process is repeated. This "myopic pricing strategy" was enhanced by Carvalho and Puterman, who estimate the value of different prices using a one-step look ahead function and choose the price with the highest total expected revenue for a given number of time steps [3].

The Kalman Filter is not unique in its ability to track hidden parameters, however, and these modifications, while allowing it to explore, do not provide compelling evidence that an optimal exploration/exploitation balance has been achieved. Because the dynamic pricing problem requires striking such a balance, PSO is an interesting alternative; it tends to naturally operate at the boundary between stability and chaos [4]. It is essentially an optimization technique based on social behavior that modifies each "particle" in a "swarm" by combining its current position in the search space $\vec{x}_i$, velocity $\vec{v}_i$, best remembered location $\vec{p}_i$, and the best location known among its neighbors $\vec{g}$ [5]. This is typically done in the following way:

$$\vec{v}_i = \chi\left(\vec{v}_i + \varphi_1\vec{U}_1 \otimes (\vec{p}_i - \vec{x}_i) + \varphi_2\vec{U}_2 \otimes (\vec{g} - \vec{x}_i)\right) \quad (1)$$

$$\vec{x}_i = \vec{x}_i + \vec{v}_i \quad (2)$$

where each $\vec{U}_i$ is a vector whose elements are drawn from a standard uniform distribution at each time step, and the $\otimes$ operator performs element-wise multiplication. The constant χ is called the "constriction coefficient" and is calculated thus:

$$\chi = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|} \quad (3)$$

usually with $\varphi = \varphi_1 + \varphi_2 > 4$ [4].

Unfortunately, standard unmodified PSO is not even suitable for application to static pricing problems due to the presence of observation noise; one abnormally favorable observation far from the optimal price can fix a particle's $\vec{p}$ (and often $\vec{g}$) to an overly optimistic value, causing the entire swarm to converge quickly on that erroneous point. This occurs because PSO is essentially greedy: only the best position is remembered by each particle, and the swarm is generally attracted to the best of those (assuming the

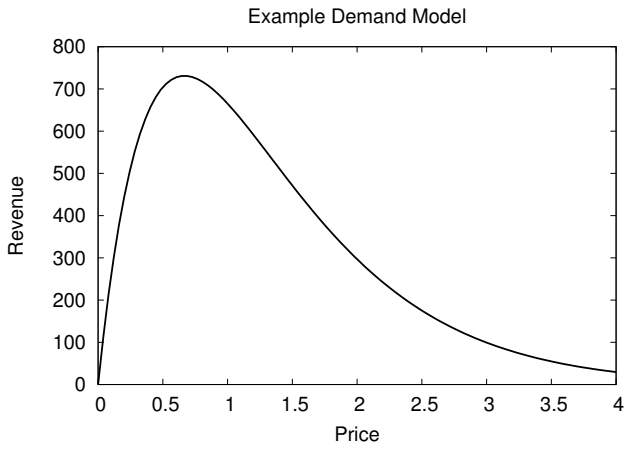

Fig. 1. Log-linear demand model, $\alpha = 8.0, \beta = -1.5$ with no noise

commonly used fully-connected sociometry). Noise alone is sufficient to cause problems for a particle swarm, but the situation worsens further in the presence of dynamic demand parameters. Adaptations to PSO must therefore be considered in the context of dynamic pricing.

This work begins with a more detailed description of the dynamic pricing problem, including the formulation of a demand model. Approaches employing the Kalman Filter are then described prior to a discussion of popular adaptations of PSO to noisy and dynamic environments. This discussion motivates the creation of a new algorithm, the P-Best Decay PSO (PBDPSO). After presenting the experimental setup, the results of applying these various algorithms are shown and discussed.

II. Pricing Model

Various economic models are in common use, and this work will focus on Kalyanam's log-linear demand model [6]. In this model, demand is represented using the parameters α and β, and the equation

$$d = e^{\alpha + p\beta + \epsilon} \tag{4}$$

where d is the quantity demanded for the chosen price p, β is strictly negative, and ϵ represents Gaussian noise with parameters μ and σ^2. The noise represents small changes in demand that occur on an irregular basis and can be due to any factor that shifts the d, such as money spent on advertising, competitor's advertising, or seasonal shifts in demand.

Interestingly, in this simplified demand model the optimal price is $p^* = -1/\beta$ and is therefore independent of α. It should be noted that maximization is performed over the *revenue* $r = pd$ in the pricing problem, not over the *quantity demanded* d (since the only requirement for maximization of the latter is a reduction of the price p to 0). An example of the shape of the log-linear demand curve is given in Figure 1.

While simple, the log-linear model admits more sophisticated and realistic scenarios by simply assuming that α and β vary with time. The following situations are of particular interest in this work:

- The parameters stay close to the original values,
- An event causes a sudden shift in the parameters, or
- A long-term trend is evident in the demand.

Results will be presented for experiments in which each of these situations is captured.

III. Kalman Filter and Pricing

The Kalman Filter estimates the hidden state of a system based on noisy observations over time using the following equations:

$$\mu_{t+1} = F\mu_t + K_{t+1}(z_{t+1} - HF\mu_t)$$
$$\Sigma_{t+1} = (I - K_{t+1}H)(F\Sigma_t F^\top + \Sigma_x)$$
$$K_{t+1} = (F\Sigma_t F^\top + \Sigma_x)H^\top (H(F\Sigma_t F^\top + \Sigma_x)H^\top + \Sigma_z)^{-1}$$

Where $\mu = (\hat{\alpha}, \hat{\beta})^\top$ is the mean of the filtered estimates of the true demand parameters, μ_0 and Σ_0 parameterize the prior Gaussian distribution over those parameters, z is the log of the observed quantity demanded, Σ_z is its covariance, F is the system transition matrix, and Σ_x is the covariance of the model parameters. Additionally, in this pricing model:

$$H = \begin{pmatrix} 1 \\ p \end{pmatrix} \qquad F = I \ .$$

This choice of F and Σ_x assumes the demand parameters α and β follow a random walk.

Using the filtered estimate $\hat{\beta}$, it is possible to attempt to maximize revenue $r = pd$ by setting the price to the estimate of the optimal price $\hat{p}^* = -1/\hat{\beta}$. This approach tends to select prices within a narrow range, a strategy that provides little information about the true nature of demand; if the initial estimate of β is incorrect, this strategy will never discover the error because β essentially defines a slope, requiring more diverse samples to achieve a good estimate. Choosing sufficiently diverse samples, however, incurs *opportunity cost*: the difference between what *might have been* earned through exploitation (letting $p = \hat{p}^*$) and what was *actually* earned during exploration (by setting it to something else).

Carvalho and Puterman address this information issue by using a one-step look ahead function [3]:

$$F_t(p_t) = p_t e^{\alpha_{t-1} + p_t \beta_{t-1}} M_{t-1} + \frac{G(t)}{2} \frac{M_{t-1} e^{\alpha_{t-1} - 1}}{\beta_{t-1}^3} \sigma_{B_t}^2(p_t) \ . \tag{5}$$

The price is chosen to maximize the objective function: $p_t = \arg\max_p F_t(p)$, where the first term is interpreted as the present estimated revenue and the second term defines the future estimated revenue (corresponding to minimizing the variance) given the price p_t. The variables α_{t-1} and β_{t-1} are set to $\hat{\alpha}$ and $\hat{\beta}$ from the Kalman Filter, respectively. Furthermore, the value $M_{t-1} = e^{\sigma_{t-1}^2/2}$, where σ_{t-1}^2 is the estimate of σ^2 for time t before demand is observed.

This algorithm assumes the availability of data points for $t \in \{-2, -1\}$. Even given these, however, at $t = 0$ sufficient information is lacking for a good estimate of σ^2. It is therefore initially set to a value known to be wrong for

the purposes of experimentation: $\sigma_0^2 = \sigma^2/2$ at $t = 0$. The algorithm is not particularly sensitive to this choice [3].

After observing the revenue at $t = 1$, the ordinary least squares method is used to estimate $\sigma_t^2 = \frac{1}{t}[\hat{\epsilon}_{-2} + \hat{\epsilon}_{-1} + \sum_{k=1}^{2} \hat{\epsilon}_k^2$ where $\hat{\epsilon}_k = log(d_k) - \alpha_t - \beta_t p_k$, $k = -2, -1, 1, \ldots, t$. The term $G(t)$ in equation 5 defines the weight given to exploration, and several different functions are used in the original work [3]. In this paper the piecewise linear function is used

$$G(t) = \begin{cases} T_c - t & \text{if } t < T_c \\ 0 & \text{otherwise} \end{cases} . \quad (6)$$

Note that T_c is not necessarily the end of the considered time horizon, and $G(t) = 0$ produces the myopic pricing strategy.

IV. Particle Swarms and Pricing

The unmodified Kalman Filter has some obvious deficiencies in this setting. First, it does not naturally explore: in using (5), Carvalho and Puterman were able to address this issue to some extent, forcing the Kalman Filter to explore. Second, the Kalman Filter requires precise knowledge or assumptions about the demand function: since it estimates the demand parameters, it must know exactly how those parameters relate to the quantity demanded; success within a log-linear model will not translate into success within other models (which may have many more parameters).

These particular deficiencies are not shared by PSO, which explores naturally and does not require explicitly-stated prior information about the target function for successful operation. Assuming that the revenue curve (e.g., Figure 1) is generally unimodal and smooth, PSO can be expected to work well in a variety of demand environments without problem-specific tuning. Even so, standard PSO has its own deficiencies in dynamic and noisy contexts, and these must be addressed before it can be successfully applied to the dynamic pricing problem.

Standard PSO has occasionally been tested in noisy environments, particularly in comparison with other evolutionary algorithms and differential evolution on common benchmarks [7], as well as with non-linear least squares algorithms on problems of determining parameters for traditional system identification tasks [8]. In both cases PSO was competitive with existing methodologies for solving those problems. Parsopoulos and Vrahatis claim that noisy environments allow particles to avoid local optima while converging on the global optimum. Their study, however, uses very small amounts of additive noise [9]; the pricing problem, in contrast, is subject to much larger amounts of noise.

Another study by Parsopoulos and Vrahatis applies PSO in situations with higher noise levels, but PSO requires thousands of iterations to converge even on two-dimensional problems [10]. The dynamic pricing problem requires significantly more agility than this, where convergence is expected to occur in a much smaller number of price settings or function evaluations (< 1000).

One adaptation stands out that improves PSO performance in noisy environments: Noise-Resistant PSO [11]. This algorithm re-evaluates all $\vec{p}$ locations after each iteration and either averages or takes the min of the new values. It appears to work well in a robotic obstacle avoidance setting but requires too many additional function evaluations to be suitable for dynamic pricing; another approach is needed.

Two things must be detected by PSO in a noisy and dynamic environment:

- Premature convergence due to noise, and
- Environmental changes that move the global optimum.

Once detected, an appropriate response to these changes must be developed that allows PSO to track the location of the optimum [12].

A. Detection Methods

Several detection strategies have been developed for dynamic environments, all of which make use of a variable N, describing a number of iterations. One approach triggers tracking if $\vec{g}$ has not moved but its *value* has changed after N iterations (NewGN), while another triggers if the *location* of $\vec{g}$ has *not* changed after N iterations (FixedGN) [13]. Yet another approach employs specialized *sentry particles* in the same manner as NewGN, re-evaluating these points during search to detect changes in the function [14]; two variants of this approach involve choosing random points as sentries (Sentry$N$) and choosing the $\vec{p}$ of a random particle as a sentry point (SentryPN). A simpler but more blunt detection tool is sometimes applied, triggering a change notification after N iterations without regard to the state of the swarm (Fixed$N$).

In consideration of the noisy pricing environment in which the algorithms will be running, these detection algorithms have been altered. When employing NewGN, Sentry$N$, or SentryPN in noisy environments, it is highly unlikely that re-sampling a position will produce the same value twice even if the environment has not changed, making the unmodified re-evaluation technique trigger too often. Therefore, instead of merely detecting a change in value, notification is only triggered when a value change exceeds some minimum percentage of variation, here taken to be 20%. Additionally, the addition of the N parameter is new to many of the previously described methods (with the exception of FixedGN), as by default they do their evaluations after every iteration. These altered approaches are outlined in Table I.

B. Response Methods

Response methods vary as well, but generally fall into one of two categories. The first involves recalculating the value at each $\vec{p}_i$: if the value is better at the particle's current position $\vec{x}_i$, then $\vec{p}_i$ is replaced with $\vec{x}_i$ (Reset) [14], [15]. Another strategy is to re-randomize a subset of particle locations and velocities (possibly including a reset of $\vec{p}$), thereby selectively erasing some particles' memory (Rand) [13], [16]; a variant of this method only re-randomizes particle i if $\vec{p}_i = \vec{g}$ (RandG). These approaches are summarized in Table II. In this table a '+' is used when the reset methodology is used in conjunction with other response techniques. For comparison purposes we also include NoResp where no changes are made to the swarm.

TABLE I

METHODS USED TO DETECT CHANGES IN THE PRICING ENVIRONMENT

Label	Detection Method Description
Fixed1	No Detection Method, Response called every time
NewG10	Re-evaluate global best, check every 10 iterations, 20% threshold
NewG20	Re-evaluate global best, check every 20 iterations, 20% threshold
FixedG10	Monitor time since global best changed, 10 iterations without change triggers response
FixedG20	Monitor time since global best changed, 20 iterations without change triggers response
FixedG30	Monitor time since global best changed, 30 iterations without change triggers response
Sentry10	Sentry, check every 10 iterations, 20% threshold
Sentry20	Sentry, check every 20 iterations, 20% threshold
SentryP10	Sentry-p-best, check every 10 iterations, 20% threshold
SentryP20	Sentry-p-best, check every 20 iterations, 20% threshold
Fixed10	Fixed-iteration, 10 iterations
Fixed20	Fixed-iteration, 20 iterations
Fixed30	Fixed-iteration, 30 iterations

TABLE II

METHODS USED TO RESPOND TO CHANGES IN THE PRICING ENVIRONMENT

Label	Response Method Description
NoResp	No Response Method
Rand1	Re-randomize 1 particle
Rand2	Re-randomize 2 particles
Rand1+	Re-randomize 1 particle, reset others
Rand2+	Re-randomize 2 particles, reset others
Reset	Reset all particles
RandG	Re-randomize g-best
RandG+	Re-randomize g-best, reset others

While interesting and useful in many dynamic situations, these approaches still suffer in the presence of noise; as mentioned above, re-evaluation of any location will almost always produce a different value. Additionally, if the locations in need of re-evaluation are far from the optimum, re-evaluation can incur significant opportunity cost with little potential for acquiring useful information. Worse still, in the pricing environment the majority of changes are gradual, so resetting the particles in the system discards all previously-obtained information about the state of the system; given the short time constraints under which the swarm is operating, throwing away so much information is unwise: the new optimum is likely to be near its previously location, implying that previous information may still have value. These issues are addressed by a new PSO variant, described below.

V. P-BEST DECAY PSO

The P-Best Decay PSO (PBDPSO) is designed to address the issues with adaptations of PSO for noisy and dynamic environments. It is essentially constricted PSO with the addition that the stored value $y_{\vec{p}_i}$ of each $\vec{p}_i$ is decayed by multiplying it by a decay rate γ. In the case of maximization, γ is within the interval $(0, 1)$. PBDPSO will replace each stored $y_{\vec{p}_i}$ with $\gamma y_{\vec{p}_i}$ (thus automatically decaying $\vec{g}$) when response is triggered by any of the various detection algorithms (Table I).

The assumption behind this approach is that each $y_{\vec{p}_i}$ is likely to be either abnormally favorable or invalid because the function has changed. By decaying $y_{\vec{p}_i}$, PBDPSO allows particles to be attracted to new areas of the space even though the noisy samples in those areas may not appear to be as good as the previous, potentially abnormal values. This allows the particles to simultaneously make use of previous information while discounting possibly noisy or dynamic data.

VI. EXPERIMENTAL SETUP

The experiments that follow require the selection and setting of various strategies and parameters. The true demand (as well as the Kalman demand model, whose parameters are learned over time) is log-linear, and its equation is initialized with $\alpha = 8.0$, $\beta = -1.5$, $\mu = 0$ and $\sigma^2 = 4.0$. To this model, one of the following four dynamic contexts is applied to the parameters of (4), described below. In all of the following scenarios, α and β vary with time and have a random component represented as additive Gaussian noise, parametrized by $(\mu_\alpha, \sigma_\alpha)$ and $(\mu_\beta, \sigma_\beta)$. Unless otherwise specified, this is represented by the following equations:

$$\alpha_t = \alpha_{t-1} + N(\mu_\alpha, \sigma_\alpha) \tag{7}$$

$$\beta_t = \beta_{t-1} + N(\mu_\beta, \sigma_\beta) \ . \tag{8}$$

In all cases, $\sigma_\alpha^2 = .05$ and $\sigma_\beta^2 = .015$. The scenarios follow.

Parameters remain near original values:

$$\mu_\alpha = \mu_\beta = 0$$

Large change at $t = 300$:

$$\alpha_{300} = \alpha_{299} + 1 + N(\mu_\alpha, \sigma_\alpha)$$
$$\beta_{300} = \beta_{299} + 0.4 + N(\mu_\beta, \sigma_\beta)$$

Overall upward trend:

$$\mu_\alpha = 0.025 \qquad \mu_\beta = 0.0075$$

Overall downward trend:

$$\mu_\alpha = -0.025 \qquad \mu_\beta = -0.0075$$

Sellers usually have a prior belief about the optimal price for their products. They also know the lowest price at which they are willing to sell and generally have a reasonable estimate of a maximum supportable price in their market, and these data will be supplied to the algorithms where needed. It is also assumed that revenue observations for the latter two price points are available, supplying the Kalman Filter with needed seed values and the particle swarm with necessary initialization bounds. The upper and lower bounds are always set at 3.0 and 0.33, respectively.

Parameters for (5) are taken from Carvalho and Puterman [3], with exploration time $T_c = 30$. They found that after the exploration period is complete, some random exploration of the pricing system can improve the Kalman Filter's ability

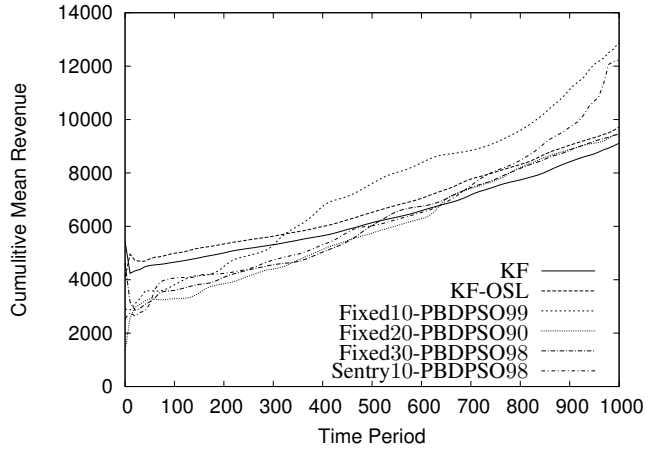

Fig. 2. Mean cumulative revenue earned with demand parameters changing with time

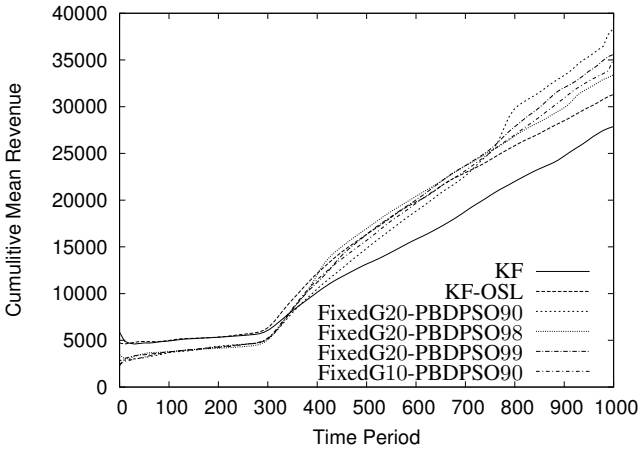

Fig. 3. Mean cumulative revenue earned with demand parameters jumping at $t = 300$

to track the parameters. Therefore, when $t > T_C$ a price is chosen according to a uniform distribution between the established bounds with probability 0.01.

Given the short time horizon, the PSO swarm size is set to 4 in all PSO algorithms, and a fully-connected sociometry is used. The chosen swarm size is large enough to admit testing responses that require more information while being small enough to avoid too many function evaluations. Additionally, $\varphi_1 = \varphi_2 = 2.05$ in all PSO experiments.

Each simulation proceeds for 1000 function evaluations, and all results shown are the mean of 1000 independent experiments. While total revenue for each algorithm is particularly important, in the interests of space the cumulative mean revenue is depicted instead, defined as $\sum_{n=1}^{t} r_n/t$, where t is the current time step and r_n is the revenue earned at $t = n$. For PSO, each detection method was run in conjunction with each response method, and these results are compared to the one-step look ahead function and unmodified Kalman Filter.

VII. Results

In the graphs presented here, the use of a Kalman Filter is denoted "KF", and the Kalman Filter employing the one-step look ahead function is denoted "KF-OSL". The remainder of the lines are indicated using the detection and response abbreviations in tables I and II. The additional notation PBDPSOn indicates that PBDPSO is applied with $\gamma = n/100$. In order to simplify the presentation, only the best-performing PSO algorithms are presented in each graph. For the curious reader more complete cumulative results are included in Tables III, IV, and V for all experiments except those involving trends.

PSO is able to overtake and surpass the performance of the Kalman Filter in the majority of the dynamic environments. Figure 2 indicates that the Kalman filter is able to quickly identify the demand parameters, but once the parameters begin to deviate from their original values the particle swarms are better equipped to adapt. The best response algorithms in

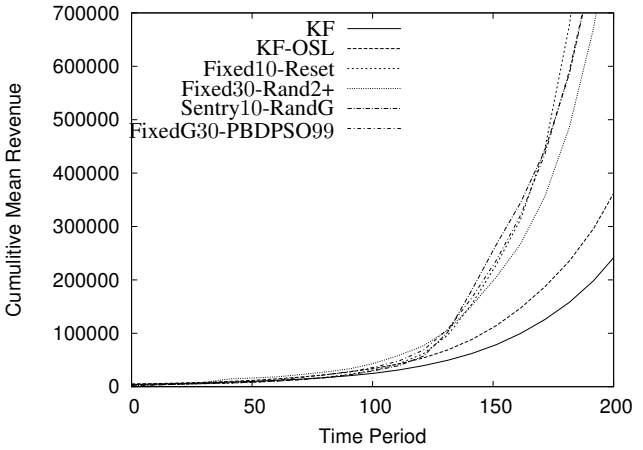

Fig. 4. Mean cumulative revenue earned with demand parameters changing with time with overall upward trend

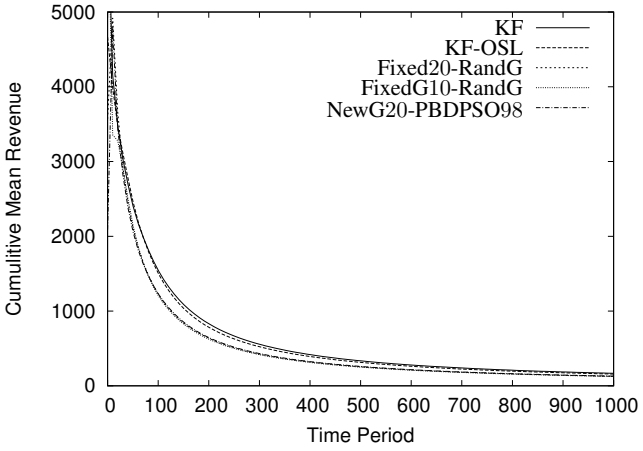

Fig. 5. Mean cumulative revenue earned with demand parameters changing with time with overall downward trend

TABLE III

COMPLETE PSO RESULTS FOR STATIC DEMAND. NUMBERS ARE AVERAGE HUNDREDS OF DOLLARS EARNED AT EACH TIME PERIOD.

	Fixed1	*NewG10*	*NewG20*	*FixedG10*	*FixedG20*	*FixedG30*	*Sentry10*	*Sentry20*	*SentryP10*	*SentryP20*	*Fixed10*	*Fixed20*	*Fixed30*	***Avg.***
NoResp	40	39	38	36	38	39	39	38	38	39	38	37	40	38
Rand1	32	33	33	33	33	31	32	32	33	31	33	32	33	32
Rand2	31	31	32	32	33	30	31	31	32	31	32	31	31	31
Rand1+	33	34	33	33	33	34	34	33	34	33	33	33	34	33
Rand2+	32	32	32	32	32	35	33	33	32	32	34	32	31	32
Reset	38	40	40	39	39	41	42	39	39	39	40	40	40	40
RandG	29	29	31	29	30	28	29	27	28	29	29	30	29	29
RandG+	31	31	31	31	31	31	31	30	32	30	30	31	31	31
PBDPSO99	41	38	39	41	37	39	42	41	39	42	39	39	40	40
PBDPSO98	39	39	41	41	41	40	39	41	40	40	43	38	41	40
PBDPSO90	41	42	39	41	40	42	41	44	41	40	43	41	41	41
Avg.	35	35	35	35	35	35	36	35	35	35	36	35	36	

TABLE IV

COMPLETE PSO RESULTS FOR CHANGING DEMAND WITH TIME. NUMBERS ARE AVERAGE HUNDREDS OF DOLLARS EARNED AT EACH TIME PERIOD.

	Fixed1	*NewG10*	*NewG20*	*FixedG10*	*FixedG20*	*FixedG30*	*Sentry10*	*Sentry20*	*SentryP10*	*SentryP20*	*Fixed10*	*Fixed20*	*Fixed30*	***Avg.***
NoResp	75	60	67	104	70	67	77	98	74	93	92	76	86	80
Rand1	62	59	69	61	76	76	72	78	68	72	80	62	66	69
Rand2	76	69	60	74	65	54	95	58	70	55	79	64	48	67
Rand1+	79	69	81	64	63	62	61	60	74	67	71	89	66	70
Rand2+	73	56	73	86	62	57	95	76	68	53	69	68	54	68
Reset	72	100	72	78	72	75	76	89	70	78	65	78	87	78
RandG	73	53	84	55	69	60	72	62	80	77	65	72	81	70
RandG+	73	71	54	68	82	62	63	63	93	51	82	62	75	69
PBDPSO99	74	83	103	102	84	87	87	77	95	73	129	75	96	90
PBDPSO98	66	94	114	90	101	87	72	82	69	91	96	70	95	87
PBDPSO90	64	92	85	64	65	93	100	108	68	93	75	94	75	83
Avg.	72	73	78	77	74	71	79	77	75	73	82	74	75	

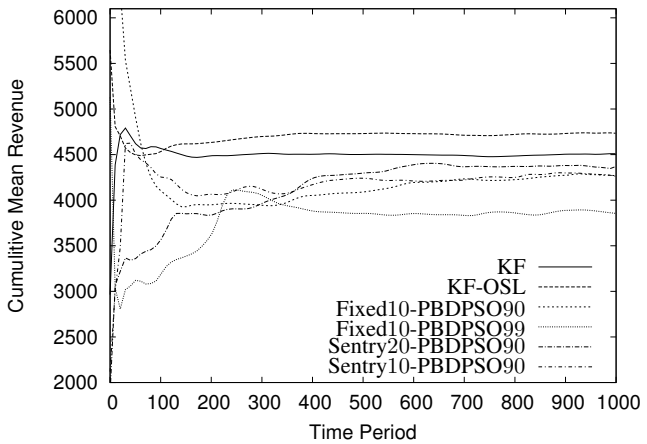

Fig. 6. Mean cumulative revenue earned with static demand

this scenario are those employing the PBDPSO as a response methodology. The best-performing detection algorithms are the Fixed$N$ and Sentry$N$ methods.

Figure 3 depicts the results of applying various algorithms to "sudden shift" scenario, where parameters change sharply at $t = 300$. In this case FixedGN is the best detection method and PBDPSO provides the best response. It is interesting to note that immediately after the parameter shift the particle swarm quickly overtakes the Kalman Filter algorithms.

In the presence of a constant upward trend, PSO significantly outperformed the Kalman Filter variants, as shown in Figure 4. Different response methods appear to be appropriate in this setting than in the previous experiments. Among response methods, the top performers are variations on Rand and Reset, with PBDPSO remaining highly competitive. When tested on the downward trend scenario, Figure 5 shows

TABLE V

COMPLETE PSO RESULTS FOR CHANGING DEMAND WITH TIME AND SUDDEN SHIFT IN DEMAND AT TIME PERIOD 300, REPORTED IN AVERAGE HUNDREDS OF DOLLARS EARNED AT EACH TIME STEP.

	Fixed1	*NewG10*	*NewG20*	*FixedG10*	*FixedG20*	*FixedG30*	*Sentry10*	*Sentry20*	*SentryP10*	*SentryP20*	*Fixed10*	*Fixed20*	*Fixed30*	***Avg.***
NoResp	289	303	327	326	294	312	341	320	299	306	345	323	313	315
Rand1	256	259	287	257	254	267	271	255	247	257	279	251	259	262
Rand2	266	288	242	272	243	277	286	287	329	257	264	281	277	274
Rand1+	313	279	265	270	289	307	269	253	250	300	269	247	296	277
Rand2+	283	266	264	280	275	264	288	288	278	256	239	274	245	269
Reset	309	283	291	279	322	297	300	322	309	288	276	312	296	299
RandG	256	247	231	234	262	236	263	256	275	250	237	245	242	249
RandG+	280	256	270	258	285	263	259	264	247	276	262	255	254	264
PBDPSO99	351	314	291	309	356	320	325	311	310	318	291	312	310	317
PBDPSO98	342	344	323	320	334	308	340	326	285	326	297	317	319	322
PBDPSO90	325	307	309	350	384	327	312	313	336	297	302	332	330	325
Avg.	297	286	282	287	300	289	296	290	288	285	278	286	285	

that PBDPSO does no better nor worse on average than any of the other top performers. In fact, none of the top performers are easily distinguished in this scenario.

Figure 6 displays the results in a noise-free and purely static demand context. In this scenario, PBDPSO is unable to approach the performance of the either of the Kalman Filter variants. It does, however outperform the other response algorithms in this setting.

Table III shows the result of applying the PSO variants to a static demand scenario, and has some interesting characteristics. In general, PBDPSO is the best response method applied, but unmodified PSO (Fixed1-NoResp) performs surprisingly well in comparison.

VIII. CONCLUSIONS

PBDPSO performs well against the Kalman Filter variants as well as outperforming existing PSO adaptations for noise and dynamic functions, except in the static scenario. While Carvalho and Puterman developed the one-step look ahead Kalman variant with the express purpose of improving performance in a dynamic setting, their published results are limited to a noisy but purely static environment. Unchanging demand parameters, while providing a good algorithm testing ground, do not represent a realistic pricing scenario, and this is the only setting in which the Kalman Filter variants dominate the results.

The results in this work indicate that PBDPSO is a well-rounded algorithm for application to the dynamic pricing problem. The exploratory nature of particle swarms, especially with the proposed adaptations, allows them to track changes in this noisy and dynamic system, thereby earning higher total revenues over time.

That no PSO variant's performance is distinguishable from that of the other methods outlined here for the case with a downward trend is unsurprising after deeper consideration. Even with the proposed decay method, PSO remains a fundamentally greedy algorithm: it continues to favor attractors with higher value, and therefore struggles with functions whose maximum is constantly decreasing. In an attempt to to improve the performance of PBDPSO in this situation, more aggressive detection and response methods have been implemented, including lower thresholds and iteration constants for the detection methods, and larger re-initialized subsets and decay values as low as 0.75 in the response methods. None of these approaches improved performance. Additionally, drawing random φ_i values from a uniform distribution on the interval $[1, 3]$ at each time step failed to improve its ability to track a downward trend.

Many promising directions are under consideration for future research. The first relates to the dynamic pricing problem itself. More experiments are needed in the downward trend scenario to determine whether any kind of particle swarm may be effectively applied. For example, rather than completely re-randomizing particles, it may help to retain their current positions while randomizing only their velocities. The problem of the downward trend is especially interesting because improvements to PSO in that environment are likely to extend to the others; noise as defined in these experiments is symmetric about the true revenue, and PBDPSO was designed exclusively for the overly optimistic case. That noise represents abnormally favorable *and* abnormally unfavorable values suggests that improvement in the presence of a downward trend will translate to improvement elsewhere.

Another opportunity for future research is a more direct and less application-centric comparison of PBDPSO with other PSO variants designed to cope with noisy or dynamic functions.

ACKNOWLEDGMENTS

This work was supported in part by the BYU bookstore and the Rollins Center for eBusiness at Brigham Young University.

REFERENCES

[1] A. X. Carvalho and M. L. Puterman, "Learning and pricing in an internet environment with binomial demands," *Journal of Revenue and Pricing Management*, vol. 3, no. 4, pp. 320–336, 2005.

[2] A. C. Harvey, *Forecasting, Structural Time Series Models and the Kalman Filter*. Press Syndicate of the University of Cambridge, 1989.

[3] A. X. Carvalho and M. L. Puterman, "Dynamic pricing and learning over short time horizons," 2003.

[4] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space." *IEEE Trans. Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[5] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *IEEE International Conference on Neural Networks (ICNN 1995)*, vol. 4, Perth, Australia, 1995, pp. 1942–1948.

[6] K. Kalyanam, "Pricing decisions under demand uncertainty: A Bayesian mixture model approach," *Marketing Science*, vol. 15, no. 3, pp. 207–221, 1996.

[7] T. Krink, B. Filipic, G. B. Fogel, and R. Thomsen, "Noisy optimization problems - a particular challenge for differential evolution?" in *Proceedings of the 2004 IEEE Congress on Evolutionary Computation (CEC 2004)*. Portland, Oregon, USA: IEEE Press, 20-23 June 2004, pp. 332–339.

[8] M. S. Voss and X. Feng, "A new methodology for emergent system identification using particle swarm optimization (PSO) and the group method data handling (GMDH)," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002)*, New York, New York, USA, July 2002, pp. 1227–1232.

[9] K. E. Parsopoulos and M. N. Vrahatis, "Particle swarm optimization for imprecise problems," in *5th international workshop on mathematical methods in scattering theory and biomedical technology*, Corfu, Greece, 2001.

[10] ——, "Particle swarm optimizer in noisy and continuously changing environments," in *IASTED International Conference on Artificial Intelligence and Soft Computing*, Cancun, Mexico, 2001, pp. 289–294.

[11] J. Pugh, Y. Zhang, and A. Martinoli, "Particle swarm optimization for unsupervised robotic learning," in *IEEE Swarm Intelligence Symposium*, 2005, pp. 92–99.

[12] X. Li and K. H. Dam, "Comparing particle swarms for tracking extrema in dynamic environments," in *IEEE Congress on Evolutionary Computation (CEC 2003)*, vol. 3, Newport Beach, California, USA, 2003, pp. 1772–1779.

[13] X. Hu and R. C. Eberhart, "Adaptive particle swarm optimisation: Detection and response to dynamic systems," in *IEEE Congress on Evolutionary Computation (CEC 2002)*, vol. 2, Honolulu, Hawaii, USA, 2002, pp. 1666–1670.

[14] A. Carlisle and G. Dozier, "Tracking changing extrema with adaptive particle swarm optimizer," in *5th Biannual World Automation Congress*, Orlando, Florida, USA, 2002, pp. 265–270.

[15] ——, "Adapting particle swarm optimization to dynamic environments," in *International Conference on Artificial Intelligence (ICAI 2000)*, Las Vegas, Nevada, USA, 2000, pp. 429–434.

[16] R. C. Eberhart and Y. Shi, "Tracking and optimizing dynamic systems with particle swarms," in *IEEE Congress on Evolutionary Computation (CEC 2001)*, Seoul, Korea, May 2001, pp. 94–97.