Bayesian Optimization Models for Particle Swarms

Christopher K. Monson
Brigham Young University
Computer Science Department
3361 TMCB, Provo, UT 84602
c@cs.byu.edu

Kevin D. Seppi
Brigham Young University
Computer Science Department
3330 TMCB, Provo, UT 84602
kseppi@cs.byu.edu

ABSTRACT

We explore the use of information models as a guide for the development of single objective optimization algorithms, giving particular attention to the use of Bayesian models in a PSO context. The use of an explicit information model as the basis for particle motion provides tools for designing successful algorithms. One such algorithm is developed and shown empirically to be effective. Its relationship to other popular PSO algorithms is explored and arguments are presented that those algorithms may be developed from the same model, potentially providing new tools for their analysis and tuning.

Track Category

Ant Colony Optimization and Swarm Intelligence

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: Optimization—*nonlinear programming, unconstrained optimization*

General Terms

Algorithms, Theory

Keywords

Swarm Intelligence, Mathematical Models, Optimization

1. INTRODUCTION

Particle Swarm Optimization (PSO) is a social or evolutionary optimization algorithm that was discovered during experiments with simulated bird flocking [7]. The discovery was valuable, as it has proven to be a good approach to the optimization of a useful class of functions. It has the additional benefit that it is easy to implement and has relatively few tunable parameters.

As the reader is assumed to have some familiarity with PSO, the explanation that follows will be brief. Classical PSO begins by scattering particles in the function domain space, often by means of a uniform distribution. Each particle is a data structure that maintains its current position $\widehat{\mathbf{x}}$ and its current velocity $\widehat{\dot{\mathbf{x}}}$. Additionally, each particle remembers the most fit position it has obtained in the past, denoted $\mathbf{p}$ for "personal best". The most fit $\mathbf{p}$ among all particles is written $\mathbf{g}$ for "global best".

Each particle updates its location over time using

$$\widehat{\dot{\mathbf{x}}}_{t+1} = \widehat{\dot{\mathbf{x}}}_t + \phi_1 \,\mathrm{U}()(\mathbf{p} - \widehat{\mathbf{x}}_t) + \phi_2 \,\mathrm{U}()(\mathbf{g} - \widehat{\mathbf{x}}_t) \qquad (1)$$

$$\widehat{\mathbf{x}}_{t+1} = \widehat{\mathbf{x}}_t + \widehat{\dot{\mathbf{x}}}_{t+1} \qquad (2)$$

where usually $\phi_1 = \phi_2 = 2$, U() is drawn from a standard uniform distribution (either a scalar or a vector of random values to be applied to each element), and velocities are constrained to be smaller than some $V_{\max}$. Two simple and popular improvements to the technique are the use of a *constriction coefficient* χ [3] and an *inertia weight* ω [14], respectively:

$$\widehat{\dot{\mathbf{x}}}_{t+1} = \chi\big(\widehat{\dot{\mathbf{x}}}_t + \phi_1 \,\mathrm{U}()(\mathbf{p} - \widehat{\mathbf{x}}_t) + \phi_2 \,\mathrm{U}()(\mathbf{g} - \widehat{\mathbf{x}}_t)\big) \qquad (3)$$

$$\widehat{\dot{\mathbf{x}}}_{t+1} = \omega\widehat{\dot{\mathbf{x}}}_t + \phi_1 \,\mathrm{U}()(\mathbf{p} - \widehat{\mathbf{x}}_t) + \phi_2 \,\mathrm{U}()(\mathbf{g} - \widehat{\mathbf{x}}_t) \ . \qquad (4)$$

When using (3), definitions of ϕ_1 and ϕ_2 are different from above, and they are used to calculate χ.

Though improvements have been made on nearly all aspects of PSO, the basic structure of the motion equations has remained largely uncontested, limiting most motion improvements to the addition or tuning of equation coefficients. While some have deviated significantly from classical motion with good success [2, 6], few have attempted such a departure. Indeed, it seems that variants on classical motion are hard to beat, but in spite of much analysis of convergence and other motion characteristics [1, 3, 4, 12], as well as some valuable intuition [8, 10], little is known as to why this is true.

In a way, the serendipitous origins of classical PSO have never been fully overcome, leaving room for a more principled and high level perspective for PSO motion. This paper presents and motivates a model-oriented approach to particle motion algorithms, providing tools for the creation of such algorithms that also aid in their analysis. This approach makes explicit the information relationships and optimization assumptions that go into the design of a swarm optimization algorithm.

One class of optimization models based on Bayesian influence networks is presented first. An algorithm is then

GECCO'05, June 25–29, 2005, Washington, DC, USA.

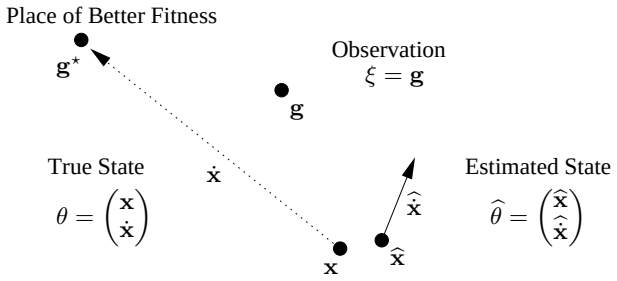


Figure 1: Raw information available to swarm optimization

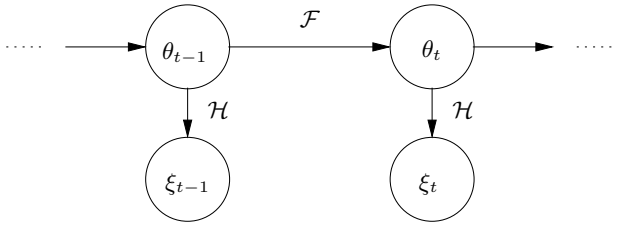


Figure 2: Hidden Markov Model

produced by solving one such model. This algorithm has some deficiencies, which are addressed by making better use of available information and applying approximations to its most complex features. The approximate algorithm is shown to perform well against some recent and successful PSO techniques. Finally, the ramifications of the new algorithm and the process by which it was created are explored.

2. BAYESIAN OPTIMIZATION MODELS

Even though "No Free Lunch" (NFL) theorems dictate that no single algorithm can be used to efficiently optimize every class of functions [15], any optimization problem can be modeled. The purpose of such a model is to sort out all of the information available during the optimization process and to make use of that information in a principled way. That it is always possible to apply a model neither ignores nor negates NFL, but rather indicates that the model must somehow explicitly specify the class of functions that are interesting.

Many such optimization models are possible, but this paper will define and restrict itself to a limited class of these models, hereafter referred to as Bayesian Optimization Models (BOMs). In a BOM, the optimization problem is framed as inference in a Dynamic Bayesian Network (DBN) where information relationships are characterized as conditional probability distributions [13]. While no assertion is made that DBNs are the only appropriate modeling tool for PSO, some class of models must be chosen, and DBNs have some particularly convenient properties.

The use of DBNs instead of a more general class of models represents an approximation that is open to debate. It is a fact, however, that an approximation of some kind must be made, since a fully expressive model of every detail of all possible information relationships would not be tractable for descriptive or computational purposes. Throughout this paper many choices will be made for the sake of approximation, and all of them will be explicit. In the interest of coherent exposition, however, a detailed discussion of those choices will be deferred until Section 6.

DBNs are frequently used to characterize time-sensitive relationships between observable (ξ) and hidden (θ) variables or states. Usually the hidden variables represent desired information and are considered to cause or influence the state of the observable variables. For example, determining the location of an airplane given a blip on a radar screen fits this model: the true location of the plane is unknown or *hidden*, it causes an *observable* blip, and the data is time-sensitive. Provided that a useful *observation model* is available to characterize the noise and other behavior of the radar, the true state can be inferred with some accuracy from a series of observed blips.

In swarm optimization, each particle may observe $\mathbf{g}$ and $\mathbf{p}$. In the context of a BOM, these observations are influenced by some state θ encoding desirable but hidden information, in this case instructions as to what the particle should do to get to a place of even better fitness $\mathbf{g}^\star$.

Ignoring $\mathbf{p}$ for the time being, Figure 1 illustrates the use of hidden states and the raw information available to a swarm optimization algorithm. At each time step, a particle has a current estimate $\widehat{\theta}$ of its hidden state θ. Since the particle is trying to track the trajectory toward $\mathbf{g}^\star$, this estimate is encoded in the actual position and velocity of the particle and is gleaned from data available in the form of $\mathbf{g}$. The hidden state θ represents where the particle ideally *should have been* ($\mathbf{x}$) and the direction in which it *should be going* ($\dot{\mathbf{x}}$) to get directly to a place of even better fitness $\mathbf{g}^\star$.

Just as a radar's observation model is required in order to estimate a plane's true location given a blip, the relationship between θ and ξ must be specified for a particle to effectively estimate a trajectory toward $\mathbf{g}^\star$. Figure 2 illustrates one of the simplest possible models for the various relationships in this system. In this case what is depicted is actually an instance of a Hidden Markov Model (HMM) where the hidden state θ influences the observable state ξ using a known *observation model* $\mathcal{H}$ and changes over time according to some *transition model* $\mathcal{F}$.

In the radar example, $\mathcal{F}$ might be a model of how a plane is expected to move. Large planes do not change speed or direction very quickly and therefore might use a constant velocity transition model, treating deviations from constant velocity as admissible noise. In the context of PSO, the transition model describes the way that $\mathbf{g}$ is expected to move over time. It also describes, however, the way that particles *prefer* to move, since they will make use of $\mathcal{F}$ as they attempt to track the trajectory of $\mathbf{g}$ over time. The $\mathcal{F}$ chosen will therefore depend on the best swarm behavior for the class of functions subject to optimization. In this paper, we will focus on a constant velocity $\mathcal{F}$ for the sake of simplicity and for historical reasons, though other relationships are certainly possible.

The model also indicates that $\mathbf{g}^\star$ influences the observed $\mathbf{g}$ since hidden state influences observations. This influence is inherently noisy because it is unreasonable to believe with absolute certainty that the observation of $\mathbf{g}$ precisely pinpoints $\mathbf{g}^\star$. Noise is thus used as a model of *subjective uncertainty* about the usefulness or accuracy of an observation. In a function with local smoothness, a Gaussian distribution is a good model of uncertainty. For example, a high-variance Gaussian with mean $\mathbf{g}$ indicates that $\mathbf{g}$ is a useful indicator

of $\mathbf{g}^\star$, but that our *belief* or *confidence* in that result is not very strong. Similarly, a low variance would indicate greater confidence that $\mathbf{g}$ is a likely place to look for $\mathbf{g}^\star$.

Regardless of the particular distribution chosen to represent belief, the use of a BOM defines the goal of optimization as the inference of a belief distribution over θ. Ideally, that distribution will converge over time to a delta function (or in the Gaussian case, a distribution with variance approaching zero) centered at the global minimum, but the amount of information available does not often allow for so much precision or certainty. The goal becomes one of finding a distribution that gets as close as possible to the truth and that represents as much confidence as possible in that estimate. This clarifies the role of the network in Figure 2: if the influences shown can be characterized, noise and all, then standard approaches to solving HMMs may be used to estimate a distribution over θ (the best action a particle can take) at each time step.

Every choice made in the creation of this particular BOM, and even the use of a BOM in the first place, represents some assumptions about the class of functions to be optimized. Sometimes the connection is readily evident and other times it is more subtle. The choice of a constant velocity model, for example, indicates that velocity contains information. A consequence of this is that fitness and distance from the global minimum should be correlated; if velocity contains information then in some sense distance must as well. Also, a Gaussian noise model introduces the assumption of some local smoothness in the function, since the belief distribution assigns similar weight to nearby regions.

Whatever the specific details, an optimization model will encompass raw information and intuition as illustrated in Figure 1 as well as explicit information relationships as illustrated in Figure 2. Together, these give an algorithm designer an opportunity to be explicit about not only the available information and what it means, but about the relationships that exist within it. The choice of BOMs in particular allows designers of swarm optimization algorithms to leverage the considerable body of existing knowledge about DBNs and HMMs to generate particle motion.

It should be reiterated that although this paper focuses almost exclusively on HMMs as models of swarm optimization, this restriction is not a requirement. Much richer models, probabilistic or otherwise, may also be applied.

3. A BOM MOTION ALGORITHM

Solution methodologies for HMMs are plentiful, but perhaps none is so easily applied as the Kalman Filter. The Kalman Filter is directly applicable to the solution of an HMM like that in Figure 2, imposing the additional constraints of linear relationships and additive Gaussian noise [5, 9, 13]. This restricted subclass of HMMs is commonly known as Linear Dynamic Systems (LDSs).

This suggests that the Kalman Filter might be applied to PSO as a means of moving particles around, an idea that was introduced earlier in the KSwarm algorithm [11]. Its original presentation was unmotivated, however, making it difficult to determine whether and in what situations it may be sensibly applied. The BOM developed thus far supplies the needed motivation, since the application of a Kalman Filter to the model produces an algorithm very much like KSwarm. To assist in further development of this idea, the basics of the original KSwarm will be presented briefly.

The purpose of the Kalman Filter is to estimate a mean $\bar{\theta}_t$ and covariance $\mathbf{\Sigma}_t$ for the hidden state of an HMM given a series of observations ξ_t, which in this case correspond to measurements of $\mathbf{g}$. The mean may be viewed as the estimate of the hidden state and the covariance as a measure of confidence in that estimate. This estimate can be obtained through recursive application of the following equations, which find themselves at the heart of KSwarm:

$$\mathbf{K}_t = (\mathbf{F}\mathbf{\Sigma}_{t-1}\mathbf{F}^\top + \mathbf{\Sigma}_\theta)\mathbf{H}^\top\big(\mathbf{H}(\mathbf{F}\mathbf{\Sigma}_{t-1}\mathbf{F}^\top + \mathbf{\Sigma}_\theta)\mathbf{H}^\top + \mathbf{\Sigma}_\xi\big)^{-1} \tag{5}$$

$$\bar{\theta}_t = \mathbf{F}\bar{\theta}_{t-1} + \mathbf{K}_t(\xi_t - \mathbf{H}\mathbf{F}\bar{\theta}_{t-1}) \tag{6}$$

$$\mathbf{\Sigma}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{H})(\mathbf{F}\mathbf{\Sigma}_{t-1}\mathbf{F}^\top + \mathbf{\Sigma}_\theta) \ . \tag{7}$$

A single sample from the distribution over a prediction provides the new state of a particle, comprising a position and velocity:

$$\widehat{\theta}_{t+1} = \begin{pmatrix}\widehat{\mathbf{x}}_{t+1}\\ \widehat{\dot{\mathbf{x}}}_{t+1}\end{pmatrix} \sim \text{Normal}(\mathbf{F}\bar{\theta}_t, \mathbf{\Sigma}_t) \ . \tag{8}$$

The value of ξ is obtained from observations, but several other values must be specified before the algorithm can proceed. For example, a transition matrix $\mathbf{F}$ and observation matrix $\mathbf{H}$ must be specified, as well as a prior mean $\bar{\theta}_0$. A constant velocity transition model makes the assumption that the trajectory traced by successive values of $\mathbf{g}$ will tend to take a straight line path toward the global minimum, and a skew-free observation model is used because it is unreasonable to assume that $\mathbf{g}$ is *not* a good estimate of $\mathbf{g}^\star$ in the absence of more information. Additonally, the priors are taken directly from the initial location and velocity of the particle. These specifications are given together here:

$$\mathbf{F} = \begin{pmatrix}\mathbf{I} & \mathbf{I}\\ \mathbf{0} & \mathbf{I}\end{pmatrix} \tag{9}$$

$$\mathbf{H} = \begin{pmatrix}\mathbf{I} & \mathbf{0}\end{pmatrix} \tag{10}$$

$$\bar{\theta}_0 = \begin{pmatrix}\widehat{\mathbf{x}}_0\\ \widehat{\dot{\mathbf{x}}}_0\end{pmatrix} \ . \tag{11}$$

Each of these values carries with it a corresponding covariance matrix, denoted $\mathbf{\Sigma}_\theta$, $\mathbf{\Sigma}_\xi$, and $\mathbf{\Sigma}_0$, respectively. Given a vector $\mathbf{w}$ of side lengths of the "feasible rectangle" (usually provided with the target function) and a small constant ϵ (≈ 0.001), the original KSwarm attempts to simplify the creation of useful covariances by defining diagonal matrices for these values, thus:

$$\mathbf{\Sigma}_\theta = \epsilon\,\text{diag}\begin{pmatrix}\mathbf{w}\\ \mathbf{w}\end{pmatrix} \tag{12}$$

$$\mathbf{\Sigma}_\xi = \epsilon\,\text{diag}(\mathbf{w}) \tag{13}$$

$$\mathbf{\Sigma}_0 = \epsilon\,\text{diag}\begin{pmatrix}\mathbf{w}\\ \mathbf{w}\end{pmatrix} \ . \tag{14}$$

Though KSwarm's published implementation outperforms one version of constricted PSO on a number of common benchmark functions, it has not been shown to be competitive with more recent or well-tuned PSO algorithms.

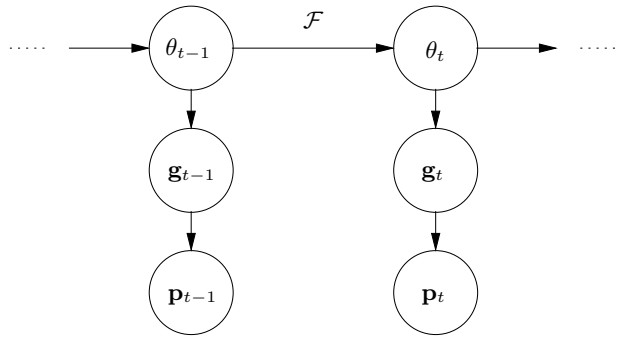


Figure 3: Belief network with p included

4. TAKING THE NEXT STEP

KSwarm was produced from a BOM, making available a large number of existing tools for its analysis. It is, however, not perfect, and several improvements immediately suggest themselves. First, KSwarm ignores the existence of $\mathbf{p}$, which is commonly considered to be an important piece of information in PSO [6, 8]. Second, the computational complexity of KSwarm is $\mathcal{O}(D^3)$ while other popular algorithms are $\mathcal{O}(D)$. Unfortunately, its performance does not warrant this increase in complexity. This section addresses both of these problems.

4.1 Incorporating Personal Best

There are several ways to introduce $\mathbf{p}$ into the HMM previously outlined. Perhaps the simplest is to combine $\mathbf{p}$ and $\mathbf{g}$ into a single observation before applying the Kalman Filter. Thus, $\xi_t = C(\mathbf{g}, \mathbf{p})$, where C is some function that combines the two pieces of information in a useful way.

If $\mathbf{p}$ is considered to be dependent on $\mathbf{g}$, the new network is represented in Figure 3. Combining $\mathbf{g}$ and $\mathbf{p}$ into a single observation is equivalent to letting $\mathbf{p}$ temper the perception of $\mathbf{g}$, which is accomplished by finding the posterior distribution $G'(\mathbf{g}|\mathbf{p})$ by application of Bayes' Law:

$$C(\mathbf{g}, \mathbf{p}) \sim G'(\mathbf{g}|\mathbf{p}) = \frac{P'(\mathbf{p}|\mathbf{g})G(\mathbf{g})}{P(\mathbf{p})} \tag{15}$$

where G and G' are the prior and posterior distributions over $\mathbf{g}$, and P and P' are the marginal and conditional distributions over $\mathbf{p}$. When using multivariate Gaussian distributions, $G'(\mathbf{g}|\mathbf{p})$ is parameterized by mean $\bar{\mathbf{b}}$ and covariance $\mathbf{\Sigma_b}$, the values of which are well known:

$$\mathbf{W} = \mathbf{\Sigma_p}(\mathbf{\Sigma_p} + \mathbf{\Sigma_g})^{-1} \tag{16}$$

$$\bar{\mathbf{b}} = (\mathbf{I} - \mathbf{W})\mathbf{p} + \mathbf{W}\mathbf{g} \tag{17}$$

$$\mathbf{\Sigma_b} = (\mathbf{I} - \mathbf{W})\mathbf{\Sigma_p} \tag{18}$$

where $\mathbf{\Sigma_p}$ and $\mathbf{\Sigma_g}$ represent uncertainty about the utility of $\mathbf{p}$ and $\mathbf{g}$ as estimates of $\mathbf{g}^\star$, respectively.

Substituting $\bar{\mathbf{b}}$ and $\mathbf{\Sigma_b}$ for ξ_t and $\mathbf{\Sigma}_\xi$ in (5), (6), and (7) yields these equations for the "$\mathbf{p}$-augmented KSwarm":

$$\mathbf{K}_t = (\mathbf{F}\mathbf{\Sigma}_{t-1}\mathbf{F}^\top + \mathbf{\Sigma}_\theta)\mathbf{H}^\top\big(\mathbf{H}(\mathbf{F}\mathbf{\Sigma}_{t-1}\mathbf{F}^\top + \mathbf{\Sigma}_\theta)\mathbf{H}^\top + \mathbf{\Sigma_b}\big)^{-1} \tag{19}$$

$$\bar{\theta}_t = \mathbf{F}\bar{\theta}_{t-1} + \mathbf{K}_t\big((\mathbf{I} - \mathbf{W})\mathbf{p} + \mathbf{W}\mathbf{g} - \mathbf{H}\mathbf{F}\bar{\theta}_{t-1}\big) \tag{20}$$

$$\mathbf{\Sigma}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{H})(\mathbf{F}\mathbf{\Sigma}_{t-1}\mathbf{F}^\top + \mathbf{\Sigma}_\theta)\ . \tag{21}$$

The equations do not change the order of polynomial complexity of the original KSwarm, but they do require the specification of yet another covariance matrix.

4.2 The Necessity of Approximations

The use of the Kalman Filter incurs some costs. It was mentioned briefly that although it tests well against a version of constricted PSO, it does not test as well against more recent improvements. This may be due to the difficulty inherent in tuning an algorithm whose parameters are all large matrices. In fact, even though some intuition may be applied to the specification of covariance matrices, the dimensionality of the parameter space far exceeds the dimensionality of the function to be optimized!

In addition, the increased computational complexity must be addressed. Fortunately, it is possible to address both issues simultaneously by crafting an approximation to the $\mathbf{p}$-augmented KSwarm algorithm. Of particular interest is (20) which can be rewritten as follows:

$$\bar{\theta}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{H})\mathbf{F}\bar{\theta}_{t-1} + \mathbf{K}_t(\mathbf{I} - \mathbf{W})\mathbf{p} + \mathbf{K}_t\mathbf{W}\mathbf{g}\ . \tag{22}$$

To simplify things further, it may be assumed that $\mathbf{H} = \mathbf{I}$ and that observation vectors are appropriately augmented with a velocity estimate:

$$\begin{aligned}\begin{pmatrix}\bar{\mathbf{x}}_t\\ \dot{\bar{\mathbf{x}}}_t\end{pmatrix} = (\mathbf{I} - \mathbf{K}_t)\mathbf{F}\begin{pmatrix}\bar{\mathbf{x}}_{t-1}\\ \dot{\bar{\mathbf{x}}}_{t-1}\end{pmatrix} \\ + \mathbf{K}_t(\mathbf{I} - \mathbf{W})\begin{pmatrix}\mathbf{p}\\ \mathbf{p} - \widehat{\mathbf{x}}_{t-1}\end{pmatrix} \\ + \mathbf{K}_t\mathbf{W}\begin{pmatrix}\mathbf{g}\\ \mathbf{g} - \widehat{\mathbf{x}}_{t-1}\end{pmatrix}\ .\end{aligned} \tag{23}$$

Because a new position may be trivially computed given the previous position and a new velocity, (23) may be further simplified by dropping all portions of the equation that directly calculate a position. Recalling from (9) that $\mathbf{F}$ preserves velocity allows it to be dropped entirely from the calculation of the mean filtered velocity $\dot{\bar{\mathbf{x}}}_t$:

$$\begin{aligned}\dot{\bar{\mathbf{x}}}_t = (\mathbf{I} - \mathbf{K}_{t,v})\dot{\bar{\mathbf{x}}}_{t-1} \\ + \mathbf{K}_{t,v}(\mathbf{I} - \mathbf{W}_v)(\mathbf{p} - \widehat{\mathbf{x}}_{t-1}) \\ + \mathbf{K}_{t,v}\mathbf{W}_v(\mathbf{g} - \widehat{\mathbf{x}}_{t-1})\ .\end{aligned} \tag{24}$$

Thus, $\dot{\bar{\mathbf{x}}}_t$ looks like a convex combination of $\dot{\bar{\mathbf{x}}}_{t-1}$, $\mathbf{g}$, and $\mathbf{p}$. Even more simplification is possible if the gains $\mathbf{K}$ and $\mathbf{W}$ are assumed to be the constant scalars a and b instead of dynamic matrices:

$$\dot{\bar{\mathbf{x}}}_t = (1 - a)\dot{\widehat{\mathbf{x}}}_{t-1} + ab(\mathbf{p} - \widehat{\mathbf{x}}_{t-1}) + a(1 - b)(\mathbf{g} - \widehat{\mathbf{x}}_{t-1}) \tag{25}$$

where $\dot{\bar{\mathbf{x}}}_{t-1}$ is approximated by $\dot{\widehat{\mathbf{x}}}_{t-1}$. This bears some resemblance to (4), the equation for inertia-weighted PSO.

Table 1: Common benchmark functions

$$\text{Sphere}(\mathbf{x}) = \sum_{i=1}^{D} x_i^2$$

$$\text{Griewank}(\mathbf{x}) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

$$\text{Rosenbrock}(\mathbf{x}) = \sum_{i=1}^{D-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$$

$$\text{DeJongF4}(\mathbf{x}) = \sum_{i=1}^{D} i x_i^4$$

$$\text{Rastrigin}(\mathbf{x}) = \sum_{i=1}^{D} x_i^2 + 10 - 10\cos(2\pi x_i)$$

The variance vector is all that remains to be determined to make the algorithm concrete. A fairly common trick when adding noise at the end of a PSO calculation is to let the variance be a scalar based on the magnitude of the mean velocity [2,6]. Without further motivation, a similar trick is applied here:

$$\widehat{\mathbf{x}}_t \sim \text{Normal}\left(\bar{\widehat{\mathbf{x}}}_t, \psi\frac{\|\bar{\widehat{\mathbf{x}}}_t\|^2}{D}\mathbf{I}\right) \tag{26}$$

where ψ scales the calculated variance by some fixed amount, usually a small number like 0.05. The dimensional scaling attempts to counter an explosion of the Gaussian support volume as dimensionality increases.

Given the mean as defined in (25) and variance from (26), a new velocity may be created by sampling once from a Gaussian distribution. This approach is the basis of the "**p** Approximate Kalman Swarm" (PAKS) algorithm, which restores $\mathcal{O}(D)$ complexity and substantially reduces the size of the parameter space.

The performance of this algorithm is compared with some popular PSO enhancements on several benchmark functions in Figure 4. The benchmarks are defined in Table 1. The results were obtained for minimization in 30 dimensions, $\alpha = 0.45$, $\beta = 0.5$, $\psi = 0.05$, and a non-reflexive "star" topology (fully connected but without self links) with 20 particles where applicable. TRIBES began with a single particle. In the case of TRIBES, the x-axis represents the best value after every 20 function evaluations to make the results directly comparable. As can be seen, PAKS either outperforms or remains competitive with TRIBES and BareBones. Although it is only an approximation, it always outperforms KSwarm, presumably because it is much easier to tune.

Although PAKS does require more tuning than TRIBES or BareBones, even the most naive parameter settings produced good behavior. Setting $\beta = 0.5$ naively assumes that **g** and **p** are equally reliable sources of information. This was also tried as the value of α, but velocity explosion required it to be lowered slightly. The only parameter that really required any significant attention was ψ, which was reasonably robust once the right range was found.

More significant than the naive and simplistic nature of the tuning is the fact that it could easily have been more principled. The use of a BOM as the basis for PAKS provides a clear and explicit path from a statistical model to a concrete algorithm. That it has roots in a model allows better tuning to occur before the algorithm has ever been executed. The coefficients, for example, are rooted in subjective variance, which was never overtly used. It is conceivable that some experiments could shed light on appropriate variances and thus on appropriate choices of coefficients, something that would not be possible without the use of the model as a starting point. Even without such analysis, it is significant that a simple algorithm using a topology not noted for its exploration capabilities can perform so favorably.

5. APPLICATIONS OF A BOM

BOMs are useful tools for specifying motion algorithms, but they represent just one possible class of models. Perhaps even more interesting than the presented BOM is the accompanying process that was used to generate useful particle swarm motion; the model is the starting point, the solution methodology creates a real algorithm, and the final approximation makes that algorithm tractable. Together these ideas represent a unified framework for function optimization that provides insights into how to tune the new algorithm and why it behaves the way that it does. All of the behavior of the algorithm may be traced back to one or more of the choices made during this process, all of which are explicit, allowing any desired change to be affected by revisiting those choices.

The full impact of this idea becomes evident when working backwards from existing motion methodologies to find appropriate models. In many cases, the very same BOM may be obtained in this manner. Consider, for example, the similarities between PAKS and other popular techniques for computing a new particle velocity:

Constricted [3]:

$$\chi\big(\widehat{\mathbf{x}}_t + \phi_1\,\mathrm{U}()(\mathbf{p} - \widehat{\mathbf{x}}_t) + \phi_2\,\mathrm{U}()(\mathbf{g} - \widehat{\mathbf{x}}_t)\big)$$

Inertia-weighted [14]:

$$\omega\widehat{\mathbf{x}}_t + \phi_1\,\mathrm{U}()(\mathbf{p} - \widehat{\mathbf{x}}_t) + \phi_2\,\mathrm{U}()(\mathbf{g} - \widehat{\mathbf{x}}_t)$$

Noisy Classical [2]:

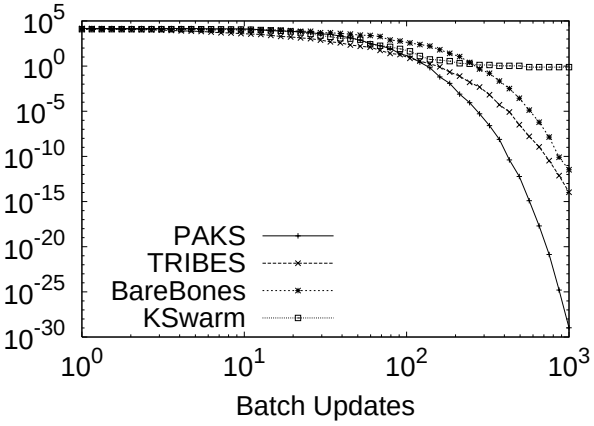
$$\chi\big(\widehat{\mathbf{x}}_t + \mathrm{G}(\mathbf{p} - \widehat{\mathbf{x}}_t, \mathbf{I}\frac{\|\mathbf{p} - \widehat{\mathbf{x}}_t\|^2}{4}) + \mathrm{G}(\mathbf{g} - \widehat{\mathbf{x}}_t, \mathbf{I}\frac{\|\mathbf{g} - \widehat{\mathbf{x}}_t\|^2}{4})\big)$$

PAKS (mean velocity):

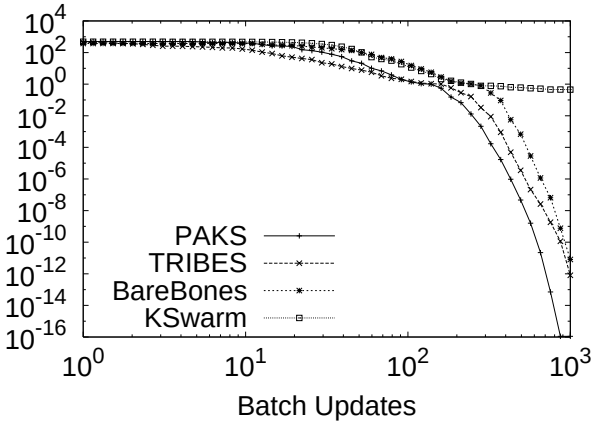
$$(1-a)\widehat{\mathbf{x}}_t + ab(\mathbf{p} - \widehat{\mathbf{x}}_t) + a(1-b)(\mathbf{g} - \widehat{\mathbf{x}}_t)$$

($\mathrm{G}(\cdot,\cdot)$ produces a sample from a Gaussian with the supplied mean vector and covariance matrix).

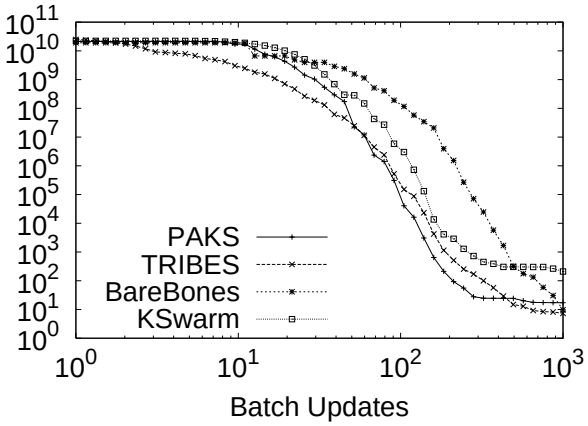
The similarity of the algorithms is striking, as each performs a noisy and linear combination of the same information. In the case of PAKS, it is clear why this is the case: Bayesian reasoning, linearity, and Gaussian noise were elective constraints during the design process. That it is similar to other popular PSO algorithms implies similarity in their underlying model and solution methodology. Consequently, all of these algorithms are describable as approximations of **p**-augmented KSwarm. The differences are limited to tuning or noise insertion and are generally superficial.



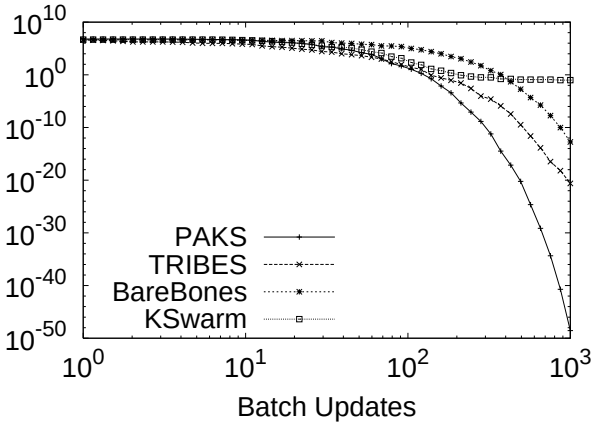
(a) Sphere



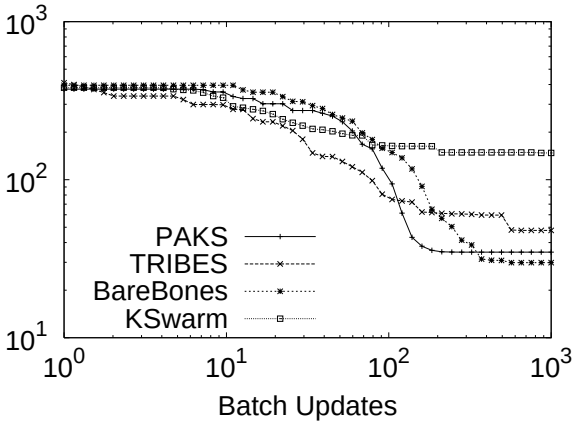
(b) Griewank



(c) Rosenbrock



(d) DeJongF4



(e) Rastrigin

Figure 4: Average fitness of PAKS compared with KSwarm, TRIBES, and BareBones

Knowing the model that can produce these algorithms makes available a wealth of information about them that was not as easily accessible before. Consider, for example, the problem of a varying inertia coefficient (ω) for inertia-weighted PSO. If we view inertia-weighted PSO as an approximation to KSwarm, some interesting things can be said about ω, which corresponds directly to the $\mathbf{I}-\mathbf{K}$ term in the Kalman Filter equations. Because $\mathbf{\Sigma}_t$ converges over time to a fixed point, a large $\mathbf{\Sigma}_0$, corresponding to low confidence in the initial estimate, will cause $\mathbf{\Sigma}_t$ to decrease over time. This decreasing variance corresponds to an *asymptotically increasing* inertia weight, unlike the *linearly decreasing* values so commonly used. While this violates some popular intuition, the superiority of an increasing inertia weight has already been empirically observed [16].

That the noise models are not precisely equivalent is no barrier to forging a connection between existing algorithms and PAKS, especially in consideration of the Central Limit Theorem. Some of them use uniform distributions on the coefficients themselves, but a sum of uniform random variables is Gaussian in the limit, a notion verified empirically in other works [2,6]. Additionally, a sum of Gaussian variables produces a new Gaussian variable, so taking a single sample at the end is not fundamentally different from taking samples on each coefficient. Perhaps the most compelling reason to suggest that all of these algorithms use a Gaussian noise model is the fact that they all combine information linearly, a consequence of the closure of Gaussians under Bayes' Law. Any other noise model would produce a more complex combination of information.

The use of BOMs also applies to motion that does not share as much in common with PAKS as the above algorithms. For example, BareBones, which performs a simple position computation based on $\mathbf{p}$ and $\mathbf{g}$ [6], may be produced from the model shown in Figure 3 by removing the time-sensitive links and dropping velocity from the approximation rather than position. Similar modifications are possible for other algorithms not discussed here.

6. MODELING AND APPROXIMATIONS

While progressing from an optimization model to a concrete PSO algorithm, several modeling decisions and approximations were made along the way. Every one of these decisions was explicit, which is one of the more important benefits of using the design framework outlined here: explicit decisions are easy to analyze and adjust. In spite of the clarity resulting from this process, however, some of the specific decisions and approximations made here have subtle consequences and merit further discussion.

6.1 Bayesian Modeling

Although DBNs do not represent the only class of models that can be used to describe information relationships, it has been made clear that simplicity and tractability were the reasons behind using that class of models. What is perhaps not as clear is the intuitive meaning of the hidden processes in Figures 1 and 2 and the approximations made in ignoring some of the possible information relationships.

Because PSO is a known and concrete algorithm, the only truly hidden process is the target function itself. The HMM presented here approximates all of the hidden characteristics of the target function as a sequence of trajectories. Intuitively, this means that the features of the target function deemed most important by the model are described by paths of improving fitness from each point in space, making a particle's goal one of finding the next step along such a path from its current location. Inherent in this is the assumption that greedy improvements to $\mathbf{g}$ form a noisy trajectory that will eventually lead to the global optimum. This assumption provides a partial specification of the class of functions for which this PSO algorithm is expected to be effective: it will work well on functions that can be described as unimodal with noise, an idea born out in the results for Sphere, DeJongF4, and Griewank.

In addition to approximating a hidden process, the proposed HMM ignores a number of possible information relationships. While adding more information links is certainly possible, limiting them to a select few is common when dealing with Bayesian networks since specifying all possible relationships is not often tractable. The underlying assumption is that noise is an adequate substitute for some of the less consequential relationships.

6.2 Adding New Information

The question naturally arises as to whether the approach to adding new information detailed in Section 4.1 is truly representative of what the information means. Is, for example, $\mathbf{p}$ really dependent upon $\mathbf{g}$? One could certainly argue that it is not, since no such dependency is built into PSO or into optimization in general. In fact, it is much easier to argue that such a relationship would work the other way, since $\mathbf{g}$ is actually the $\mathbf{p}$ of some particle.

This is readily addressed by looking at the model as a description of observed behavior rather than of a known process. While $\mathbf{g}$ does not *cause* the observation of $\mathbf{p}$, it can be *observed* that $\mathbf{p}$ tends to be close to $\mathbf{g}$ during a run of PSO.

The dependency of $\mathbf{p}$ on $\mathbf{g}$ can also be viewed as a *specification* of behavior. This observation of correlation or clustering among $\mathbf{p}$ and $\mathbf{g}$ is, in a sense, a self-fulfilling prophecy: the model says that clustering occurs, the same model drives the motion of particles, and therefore clustering is observed.

This brings up an interesting side point. Many choices were made in this paper not only because they were simple and convenient, but because they led to an algorithm that was similar to existing PSO algorithms. This allowed the framework to be developed in a familiar context and provided a means of rethinking popular motion algorithms in terms of a useful model. Remarkably little foresight was required to do this since some of the approximations made were rather obvious and naive. For these reasons, the seemingly counterintuitive dependency of $\mathbf{p}$ on $\mathbf{g}$ is actually not at all unreasonable; it is descriptive, prescriptive, and effectively models the behavior of existing algorithms.

6.3 Other Approximations

Other notable approximations in Section 4.2 include dropping position instead of velocity and ignoring the Kalman variance calculation in (21).

Position was dropped instead of velocity to better facilitate the connection between PAKS and other algorithms, especially inertia-weighted PSO. It would definitely be possible to drop velocity instead, producing something more like BareBones, as previously discussed.

As for PAKS variance, the corresponding Kalman equations were ignored for simplicity and space. This amounted to replacing useful information with a trick adapted from

the literature, highlighting the fact that significant approximations are often made while building a PSO algorithm. In the case of PAKS, it was clear that the correct variance calculation was ignored, but it was only obvious because the underlying model was already known. Many PSO algorithms, on the other hand, commonly make equally sweeping assumptions without any context for their objective evaluation. Indeed, it is difficult if not impossible to provide such context without the use of a model.

7. CONCLUSIONS AND FUTURE WORK

The single-objective optimization problem is one of using available information to find the global minimum. In order to do this, it is useful to specify in explicit terms what that information is and the relationships between various pieces of the available information. Bayesian Optimization Models are a class of models that make this specification systematic and principled, simultaneously lending valuable intuition to the process.

That this approach can serve as the foundation of a number of PSO motion algorithms, including those developed here, makes it useful as a tool for high level analysis of both new and existing PSO algorithms. It would be interesting to apply in greater detail the model and approximation techniques presented here to existing PSO algorithms. This could provide more insights into their differences in behavior on various functions.

Other DBN models may be used besides the HMM suggested here, and other solution methodologies, such as particle filters, may be applied. More information than $\mathbf{g}$ and $\mathbf{p}$ could also be made available to any chosen model, and methods of combining information should be explored more fully. Additionally, the approximation process outlined for PAKS could be changed to make better use of the model to make coefficient tuning more principled.

In this work, the application of a BOM has only affected each particle individually, but it is possible to create a richer model that includes the notion of sociometry and the information flow between particles in the swarm. Though complex, the study of such influences is at least possible using a model, and its use may provide new topological insights.

The algorithm design framework developed around BOMs actually exists independently of them, suggesting that any model, Bayesian or otherwise, may be used in the framework. The only requirement is that such a model make information relationships explicit and provide a means for inferring desired information from available information. The exploration of alternative models would be an interesting pursuit.

The introduction of a BOM and its success in creating a competitive PSO algorithm highlights the utility of the associated algorithm design framework. The framework is not only valuable as a tool for the synthesis of PSO algorithms, but also for their analysis. This work has presented the framework and model-based approach as a way of *thinking* about optimization and this perspective suggests new ways of approaching the problem.

8. REFERENCES

[1] M. Clerc. The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1999)*, pages 1951–1957, Piscataway, New Jersey, 1999.

[2] M. Clerc. TRIBES - un exemple d'optimisation par essaim particulaire sans paramètres de contrôle. In *Optimisation par Essaim Particulaire (OEP 2003)*, Paris, France, 2003.

[3] M. Clerc and J. Kennedy. The particle swarm: Explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, February 2002.

[4] R. C. Eberhart and Y. Shi. Comparing inertia weights and constriction factors in particle swarm optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2000)*, pages 84–88, San Diego, California, 2000.

[5] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[6] J. Kennedy. Bare bones particle swarms. In *Proceedings of the IEEE Swarm Intelligence Symposium 2003 (SIS 2003)*, pages 80–87, Indianapolis, Indiana, 2003.

[7] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *International Conference on Neural Networks IV*, pages 1942–1948, Piscataway, NJ, 1995. IEEE Service Center.

[8] J. Kennedy and R. C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers, 2001.

[9] C. T. Leondes, editor. *Theory and Applications of Kalman Filtering*. Number 139 in AGARDograph. North Atlantic Treaty Organization, Advisory Group for Aerospace Research and Development, 1970.

[10] R. Mendes, J. Kennedy, and J. Neves. Watch thy neighbor or how the swarm can learn from its environment. In *Proceedings of the IEEE Swarm Intelligence Symposium 2003 (SIS 2003)*, pages 88–94, Indianapolis, Indiana, 2003.

[11] C. K. Monson and K. D. Seppi. The Kalman swarm. In *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 140–150, Seattle, Washington, 2004.

[12] E. Ozcan and C. K. Mohan. Particle swarm optimizaton: Surfing the waves. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1999)*, Washington, D.C., 1999.

[13] S. Russel and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Englewood Cliffs, New Jersey, second edition, 2003.

[14] Y. Shi and R. C. Eberhart. A modified particle swarm optimizer. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1998)*, Piscataway, New Jersey, 1998.

[15] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997.

[16] Y. Zheng, L. Ma, L. Zhang, and J. Qian. Empirical study of particle swarm optimizer with an increasing inertia weight. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2003)*, pages 221–226, Canbella, Australia, 2003.