Stat
The ISI's Journal for the Rapid
Dissemination of Statistics Research
(wileyonlinelibrary.com) DOI: 10.100X/sta.0000

Adaptively-Tuned Particle Swarm Optimization with Application to Spatial Design

Received ; Accepted

Particle swarm optimization (PSO) algorithms are a class of heuristic optimization algorithms that are especially attractive for complex optimizations problems. We propose using PSO in order to solve spatial design problems, e.g. choosing new locations to add to an existing monitoring network. To this end, we introduce a new class of PSO algorithms, called adaptively-tuned PSO, that perform well in a wide variety of circumstances. In order to illustrate these algorithms, we apply them to a common spatial design problem: choosing new locations to add to an existing monitoring network. Specifically, we consider a network in the Houston, TX area for monitoring ambient ozone levels, which have been linked to out-of-hostpital cardiac arrest rates (Ensor et al., 2013).

Keywords: optimization; particle swarm; geostatistics; kriging; optimal design, spatial prediction

1. Introduction

PSO refers to a large class of heuristic optimization algorithms that use an analogy with animal flocking behavior in order to construct more robust optimization algorithms than many alternatives (Clerc & Kennedy, 2002; Blum & Li, 2008; Clerc, 2010). This robustness makes them attractive for more difficult optimization problems, especially when near optimal solutions are tolerable. We introduce a new class of PSO algorithms, called adaptively tuned PSO (AT-PSO), which exploit an analogy with a class of adaptive Markov chain Monte Carlo algorithms in order to tune a crucial parameter of the PSO algorithm adaptively based on the state of the particle swarm. We show that the resulting algorithms tend to be superior to many PSO alternatives. Additionally, we illustrate several PSO algorithms by applying them to choosing a set of new monitoring locations for ozone in Harris County, Texas, where Houston is located. Ambient ozone levels have been linked to cardiac arrest (Ensor et al., 2013), so ozone monitoring is an essential tool for determining when and where populations are at risk.

Section 2 introduces various PSO and AT-PSO algorithms, and briefly discusses the results of an extended simulation study comparing them. Section 3 introduces the generic spatial design problem and the Houson area ozone problem

*Email:

Prepared using staauth.cls [Version: 2012/05/12 v1.00]

as an instance of that problem, and compares several PSO algorithms and some alternatives to solving the problem. Finally, Section 4 discusses our results and concludes.

2. Particle swarm optimization

We briefly describe PSO here; refer to Blum & Li (2008) for a comprehensive introduction, Clerc (2010) for details, and Clerc (2011) for good default versions of the algorithm. Suppose that we wish to minimize some objective function $Q(\boldsymbol{\theta}) : \Re^D \to \Re$. Let $i = 1, 2, \ldots, n$ index a set of particles over time, $k = 1, 2, \ldots, K$, where in every period each particle consists of a location $\boldsymbol{\theta}_i(k) \in \Re^D$, a velocity $\boldsymbol{v}_i(k) \in \Re^D$, a personal best location $\boldsymbol{p}_i(k) \in \Re^D$, and a group best location $\boldsymbol{g}_i(k) \in \Re^D$. Here we mean "best" in the sense of minimizing Q, so $Q(\boldsymbol{p}_i(k)) \geq Q(\boldsymbol{\theta}_i(l))$ for any $k \geq l$. The group best location is defined with respect to some neighborhood $\mathcal{N}_i$ of particle i; that is, $\boldsymbol{g}_i(k) = \arg\min_{\{\boldsymbol{p}_j(k) | j \in \mathcal{N}_i\}} Q(\boldsymbol{p}_j(k))$. In the simplest case where the entire swarm is the neighborhood of each particle, $\boldsymbol{g}_i(k) \equiv \boldsymbol{g}(k) = \arg\min_{\{\boldsymbol{p}_j(k) | j \in 1:n\}} Q(\boldsymbol{p}_j(k))$. The generic PSO algorithm updates each particle i as follows:

$$
\begin{aligned}
\boldsymbol{v}_i(k+1) &= \omega \boldsymbol{v}_i(k) + \phi_1 \boldsymbol{r}_{1i}(k) \circ \{\boldsymbol{p}_i(k) - \boldsymbol{\theta}_i(k)\} + \phi_2 \boldsymbol{r}_{2i}(t) \circ \{\boldsymbol{g}_i(k) - \boldsymbol{\theta}_i(k)\}, \\
\boldsymbol{\theta}_i(k+1) &= \boldsymbol{\theta}_i(k) + \boldsymbol{v}_i(k+1), \\
\boldsymbol{p}_i(k+1) &= \begin{cases} \boldsymbol{p}_i(k) & \text{if } \; Q(\boldsymbol{p}_i(k)) \leq Q(\boldsymbol{\theta}_i(t+1)) \\ \boldsymbol{\theta}_i(k+1) & \text{otherwise,} \end{cases} \\
\boldsymbol{g}_i(k+1) &= \arg \min_{\{\boldsymbol{p}_j(k+1) | j \in \mathcal{N}_i\}} Q(\boldsymbol{p}_j(k+1)),
\end{aligned} \tag{1}
$$

where $\circ$ denotes the Hadamard product (element-wise product), $\boldsymbol{r}_{1i}(k)$ and $\boldsymbol{r}_{2i}(k)$ are each vectors of D random variates independently generated from the $U(0,1)$ distribution, and $\omega > 0$, $\phi_1 > 0$, and $\phi_2 > 0$ are user-defined parameters. The term $\omega \boldsymbol{v}_i(k)$ controls the particle's tendency to keep moving in the direction it is already going, so ω is called the inertia parameter. Similarly $\phi_1 \boldsymbol{r}_{1i}(k) \circ \{\boldsymbol{p}_i(k) - \boldsymbol{\theta}_i(k)\}$ controls the particle's tendency to move towards its personal best location while $\phi_2 \boldsymbol{r}_{2i}(k) \circ \{\boldsymbol{g}_i(k) - \boldsymbol{\theta}_i(k)\}$ controls its tendency to move toward its group best location, so ϕ_1 and ϕ_2 are called the cognitive correction factor and social correction factor, respectively (Blum & Li, 2008). This version of PSO is equivalent to Clerc & Kennedy (2002)'s constriction type I particle swarm, though there are many other variants. One default choice sets $\omega = 0.7298$ and $\phi_1 = \phi_2 = 1.496$ (Clerc & Kennedy, 2002) while another sets $\omega = 1/(2 \ln 2) \approx 0.721$ and $\phi_1 = \phi_2 = 1/2 + \ln 2 \approx 1.193$ (Clerc, 2006).

Any PSO variant can also be combined with various neighborhood topologies that control how the particles communicate to each other. The default global topology allows each particle to see each other particle's previous best location for the social components of their respective velocity updates, but this can cause inadequate exploration and premature convergence. Alternative neighborhood topologies limit how many other particles each particle can communicate with. We use a variant of the stochastic star topology (Miranda et al., 2008). Each particle informs itself and k random particles from the swarm, sampled with replacement during initialization of the algorithm. On average this implies that each particle is informed by k particles, though a small number of particles will often be informed by many of the other particles

Bare bones PSO (BBPSO) is a variant of PSO introduced by Kennedy (2003) that strips away the velocity term. Let $\theta_{ij}(k)$ denote the jth coordinate of the position for the ith particle in period t, and similarly for $p_{ij}(k)$ and $g_{ij}(k)$. Then the BBPSO algorithm obtains a new position coordinate θ_{ij} via

$$
\theta_{ij}(k+1) \sim N\left(\frac{p_{ij}(k) + g_{ij}(k)}{2}, |p_{ij}(k) - g_{ij}(k)|^2 \right). \tag{2}
$$

The updates of $\boldsymbol{p}_i(k)$ and $\boldsymbol{g}_i(k)$ are the same as in (1). We explain the essential ideas of PSO and BBPSO in this section, though several modifications can be made to improve performance. Appendix A details each of the modifactions we use, most of which come from Clerc (2011). In the next two subsections we introduce adaptively-tuned BBPSO and adaptively-tuned PSO respectively. Both of these algorithms can be combined with the various modifications we discuss in Appendix A.

2.1. Adaptively tuned BBPSO

BBPSO adapts the size effective search space of the swarm over time through the variance term, $|p_{ij}(k) - g_{ij}(k)|^2$. As the personal best locations of the swarm move closer together, these variances decrease and the swarm explores locations which are closer to known high value areas in the space. This behavior is desirable, but the adaptation is forced to occur only through personal and group best locations. In order to allow the algorithm to adapt in a more flexible manner, we modify the BBPSO variance to $\sigma^2(k)|p_{ij}(k) - g_{ij}(k)|^2$ and tune $\sigma^2(k)$ in a manner similar to adaptive random walk Metropolis MCMC algorithms (Andrieu & Thoms, 2008). Define the improvement rate of the swarm in period t as $R(k) = \#\{i : Q(\boldsymbol{p}_i(k)) > Q(\boldsymbol{p}_i(k-1))\}/n$ where $\#A$ is the number of members of the set A, and let R^* denote the target improvement rate. Then adaptively tuned BBPSO (AT-BBPSO) updates the swarm's personal best and group best locations as in (1), then updates particle locations as follows:

$$\begin{aligned} \theta_{ij}(k+1) &\sim t_{df}\left(\frac{p_{ij}(k) + g_{ij}(k)}{2}, \sigma^2(k)|p_{ij}(k) - g_{ij}(k)|^2\right), \\ \log\sigma^2(k+1) &= \log\sigma^2(k) + c \times \{R(k+1) - R^*\}, \end{aligned} \tag{3}$$

where df is a user chosen degrees of freedom parameter and c is another user chosen parameter that controls the speed of adaptation. We use a t kernel instead of a Gaussian in order to allow for more flexibility, and we find $df = 1$ appears to combine well with adaptively tuning $\sigma^2(k)$. Setting the target rate from $R^* = 0.3$ to $R^* = 0.5$ tends to yield AT-BBPSO algorithms which work well, which is unsurprising given the connection to adaptive random walk Metropolis (Gelman et al., 1996). The parameter c controls the speed of adaptation so that larger values of c mean the algorithm adapts $\sigma^2(k)$ faster. We find that $c = 0.1$ works well, though anything within an order of maginitude yields similar results. We use $\sigma^2(0) = 1$ to initialize the algorithm at the standard BBPSO algorithm. Optimal selection of the tuning parameters, in particular c, is a subject of future research.

Both using a t kernel and adding a fixed scale parameter have been discussed in the BBPSO literature, but as Kennedy (2003) notes, something about setting $\sigma^2 = 1$ is special that causes the algorithm to work well. A similar BBPSO algorithm from Hsieh & Lee (2010) sets $\sigma^2 < 1$ for an initial set of iterations, then eventually sets $\sigma^2 = 1$. The authors alo suggest dynamically adjusting σ^2 in the early stage of the algorithm before they are set to one, but give no suggestion for how to do this. AT-BBPSO is able to adapt its value on the fly based on local knowledge about the objective function. If too much of the swarm is failing to find new personal best locations, AT-BBPSO proposes new locations closer to known high value areas. If too much of the swarm is improving, AT-BBPSO proposes bolder locations in an effort to make larger improvements. This ability to adapt to local information about the objective function allows AT-BBPSO to more quickly traverse the search space towards the global optimum, though by using local information AT-BBPSO does risk premature convergence to a local optimum. The adaptively tuned component of AT-BBPSO can also be combined with most BBPSO variants, some of which are outlined in Appendix A. In Appendix B we conduct a simulation study on several test functions that compares AT-BBPSO variants to other PSO and BBPSO variants in order to justify the parameter settings discussed above and demonstrate AT-BBPSO variants are attractive PSO algorithms.

2.2. Adaptively-tuned PSO

In AT-BBPSO variants, the parameter $\sigma(k)$ partially controls the effective size of the swarm's search area, and we increase or decrease $\sigma(k)$ and consequently the search area depending on how much of the swarm is finding new personal best locations. In standard PSO the inertia parameter, denoted by ω in (1), is roughly analogous to σ^2 in BBPSO. It controls the effective size of the swarm's search area by controlling how the magnitude of the velocities evolve over time. In AT-BBPSO we use an analogy with tuning a random walk Metropolis-Hastings MCMC algorithm in order to build intuition about how to tune $\sigma(k)$. The analogy is much weaker in this case; nonetheless, the same mechanism works well to tune $\omega(k)$. Formally, AT-PSO updates personal and group best locations as in (1) and updates $\omega(k)$ and particle locations as follows:

$$\begin{aligned}
\boldsymbol{v}_i(k+1) &= \omega(k+1)\boldsymbol{v}_i(k) + \phi_1 \boldsymbol{r}_{1i}(k) \circ \{\boldsymbol{p}_i(k) - \boldsymbol{\theta}_i(k)\} + \phi_2 \boldsymbol{r}_{2i}(k) \circ \{\boldsymbol{g}_i(k) - \boldsymbol{\theta}_i(k)\}, \\
\boldsymbol{\theta}_i(k+1) &= \boldsymbol{\theta}_i(k) + \boldsymbol{v}_i(k+1), \\
\log \omega(k+1) &= \log \omega(k) + c \times \{R(k+1) - R^*\},
\end{aligned} \tag{4}$$

where $R(k)$ is the improvement rate of the swarm in iteration t, R^* is the target improvement rate, and c controls how much $R(k)$ changes on a per iteration basis. Again, we find that target rates from $R^* = 0.3$ to 0.5 work well for AT-PSO. The value of c controls the speed of adaptation. We use $c = 0.1$ as a default value and in simulations (not reported here), we find that the gains from optimizing c appear to be small. However, optimal selection of these tuning parameters is a subject of future research. The adaptive tuning piece of this algorithm can be combined with almost any PSO variant, and Appendix A describes the various modifications to the standard PSO algorithm we employ, both in conjuction with AT and without it.

The idea of time-varying $\omega(k)$ has been in the PSO literature for some time. An early suggestion was to set $\omega(0) = 0.9$ and deterministically decrease it until it reaches $\omega(k) = 0.4$ after the maximum number of iterations allowed (Eberhart & Shi, 2000). In particular, Tuppadung & Kurutach (2011) suggest defining $\omega(k)$ via the parameterized inertia weight function $\omega(k) = \frac{1}{1+\left(\frac{t}{\alpha}\right)^{\beta}}$ where α and β are user-defined parameters. Roughly, α controls how low $\omega(k)$ can go and β controls how fast it gets there, so α and β can be thought of as intercept and slope parameters respectively. The suggestion in Tuppadung & Kurutach (2011) is to set α to a small fraction of the total amount of iterations in which the algorithm is allowed to run (e.g., 10% or 20%), and set β between one and four. We call this type of PSO algorithm deterministic inertia PSO (DI-PSO).

DI-PSO tends to improve on standard PSO if $\omega(k)$'s progression is set appropriately, but this can be difficult and often depends on the problem. A priori it may not be clear exactly which approach is best for any given problem, so an automatic method is desirable. A major strength of AT-PSO relative to DI-PSO and standard PSO is that AT-PSO can increase $\omega(k)$ when information from the swarm suggests there is an unexplored high value region of the space. Just like in AT-BBPSO, this mechanism provides a way for the swarm to adapt its behavior on the fly based on local conditions and speed up convergence by allowing the particles that do improve to make larger improvements, but it can also cause premature convergence to a local optimum. While DI-PSO monotonically decreases $\omega(k)$ toward some minimum value, AT-PSO typically oscillates $\omega(k)$ so that the algorithm alternates between exploring and exploiting more, relative to standard PSO. Appendix B contains an extended simulation study comparing a variety of these PSO and BBPSO algorithms on a suite of test functions that demonstrates some of the behavior detailed above and shows that AT-PSO is an attractive PSO algorithm.

Another similar algorithm comes from Zhang et al. (2003). In their algorithm, ω is constant while ϕ_1 and ϕ_2 not only vary across time, but also across the swarm. Then each particle adapts its values of ϕ_1 and ϕ_2 based on how much it improves on its personal best location. The key difference is that the adaptation is local to each specific particle while

in AT-PSO the adaptation is global. AT-PSO takes into account less information by ignoring local conditions of the particles, but the adaptation scheme is simpler.

3. The Spatial Design Problem

Suppose we are interested predicting some spatially indexed response variable $Y(\boldsymbol{u})$, $\boldsymbol{u} \in \mathcal{D} \subseteq \Re^2$ at a set of target locations $\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_{N_t} \in \mathcal{D}$. Let $\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_{N_s} \in \mathcal{D}$ denote a set of N_s fixed sampling locations within the spatial domain. The design problem is to add N_d new sampling locations in order to optimize the amount we learn about $Y(\boldsymbol{u})$ at the target locations. Let $\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_{N_d} \in \mathcal{D}$ denote a set of candidate design points and suppose that $Y(\boldsymbol{u})$ is a geostatistical process with mean function $\mu(\boldsymbol{u}) = \boldsymbol{x}(\boldsymbol{u})'\boldsymbol{\beta}$ for some covariate $\boldsymbol{x}(\boldsymbol{u})$ known at every point in $\mathcal{D}$ and some covariance function $C(\boldsymbol{u}, \boldsymbol{v})$ for $\boldsymbol{u}, \boldsymbol{v} \in \mathcal{D}$. Not all covariates can be known a priori at every point in the spatial domain; however, covariates that are known functions of the location satisfy this constraint. Once the design points are selected, we observe $Z(\boldsymbol{d}_i)$ for $i = 1, 2, \ldots, N_d$ and $Z(\boldsymbol{s}_i)$ for $i = 1, 2, \ldots, N_s$ where $Z(\boldsymbol{u}) = Y(\boldsymbol{u}) + \varepsilon(\boldsymbol{u})$ and $\varepsilon(\boldsymbol{u})$ is mean zero white noise with variance τ^2, representing measurement error. Typically $\boldsymbol{\beta}$, τ^2, and $C(.,.)$ are unknown and must be estimated, though for now we will treat them as known.

To completely specify the problem we need to define an informative design criterion. Intuitively, the larger the mean square prediction error (MSPE), i.e. the kriging variance, is at each of the target locations, the less information we have about $Y(\boldsymbol{u})$ at those locations. A common design criterion is to optimize is some function of these variances, e.g. to minimize the mean kriging variance or the maximum kriging variance over all target locations. These criteria are somewhat naive since they they ignore parameter uncertainty, and taking that into account will often change the optimal design (Zimmerman, 2006).

3.1. Universal Kriging

In universal kriging, $C(.,.)$ and τ^2 are treated as known while $\boldsymbol{\beta}$ is treated as a parameter that neeeds to be estimated. Let $\boldsymbol{Z}$ be the vector of $Z(\boldsymbol{s}_i)$s and $Z(\boldsymbol{d}_i)$s, $\boldsymbol{X}$ denote the corresponding stacked $\boldsymbol{x}(\boldsymbol{s}_i)'$s and $\boldsymbol{x}(\boldsymbol{d}_i)'$s, $\boldsymbol{C}_Z = \text{cov}(\boldsymbol{Z})$ where $\text{cov}[Z(\boldsymbol{u}), Z(\boldsymbol{v})] = C(\boldsymbol{u}, \boldsymbol{v}) + \sigma^2_\varepsilon 1(\boldsymbol{u} = \boldsymbol{v})$, and $\boldsymbol{c}_Y(\boldsymbol{t}_i) = \text{cov}[Y(\boldsymbol{t}_i), \boldsymbol{Z}]$ where $\text{cov}[Y(\boldsymbol{t}_i), Z(\boldsymbol{u})] = C(\boldsymbol{t}_i, \boldsymbol{u})$. The universal kriging predictor of $Y(\boldsymbol{t}_i)$ is $\widehat{Y}_{uk}(\boldsymbol{t}_i; \boldsymbol{d}) = \boldsymbol{x}(\boldsymbol{t}_i)'\widehat{\boldsymbol{\beta}}_{gls} + \boldsymbol{c}_Y(\boldsymbol{t}_i)'\boldsymbol{C}_Z^{-1}(\boldsymbol{Z} - \boldsymbol{X}\widehat{\boldsymbol{\beta}}_{gls})$ and the MSPE of $\widehat{Y}_{uk}(\boldsymbol{t}_i)$ is

$$\begin{aligned}\sigma^2_{uk}(\boldsymbol{t}_i; \boldsymbol{d}) = C(\boldsymbol{t}_i, \boldsymbol{t}_i) - \boldsymbol{c}_Y(\boldsymbol{t}_i)'\boldsymbol{C}_Z^{-1}\boldsymbol{c}_Y(\boldsymbol{t}_i) \\ + [\boldsymbol{x}(\boldsymbol{t}_i) - \boldsymbol{X}'\boldsymbol{C}_Z^{-1}\boldsymbol{c}_Y(\boldsymbol{t}_i)]'[\boldsymbol{X}'\boldsymbol{C}_Z^{-1}\boldsymbol{X}]^{-1}[\boldsymbol{x}(\boldsymbol{t}_i) - \boldsymbol{X}'\boldsymbol{C}_Z^{-1}\boldsymbol{c}_Y(\boldsymbol{t}_i)].\end{aligned}$$

where $\widehat{\boldsymbol{\beta}}_{gls} = [\boldsymbol{X}'\boldsymbol{C}_Z^{-1}\boldsymbol{X}]^{-1}\boldsymbol{X}'\boldsymbol{C}_Z^{-1}\boldsymbol{Z}$ is the generalized least squares estimate of $\boldsymbol{\beta}$ (Cressie & Wikle, 2011, Section 4.1.2). To avoid clutter we drop the explicit dependence on $\boldsymbol{d}$ in these equations, but $\boldsymbol{c}_Y(\boldsymbol{t}_i)$, $\boldsymbol{C}_Z$, $\boldsymbol{Z}$, $\boldsymbol{X}$, and $\widehat{\boldsymbol{\beta}}_{gls}$ all depend on $\boldsymbol{d}$. The mean universal kriging variance is given by $\overline{Q}_{uk}(\boldsymbol{d}) = \frac{1}{N_t}\sum_i^{N_t} \sigma^2_{uk}(\boldsymbol{t}_i; \boldsymbol{d})$ while the maximum universal kriging variance is given by $Q^*_{uk}(\boldsymbol{d}) = \max_{i=1,2\ldots,N_t} \sigma^2_{uk}(\boldsymbol{t}_i; \boldsymbol{d})$. Zimmerman (2006) finds that the optimal design under both criteria is highly dependent on the class of mean function of the geostatistical process. In practice we are often interested in predicting at the entire spatial domain rather than a finite set of target locations. This changes the mean and maximum kriging variances to $\overline{Q}_{uk}(\boldsymbol{d}) = \frac{1}{|\mathcal{D}|}\int_{\mathcal{D}} \sigma^2_{uk}(\boldsymbol{u}; \boldsymbol{d})d\boldsymbol{u}$ and $Q^*_{uk}(\boldsymbol{d}) = \max_{\boldsymbol{u}\in\mathcal{D}} \sigma^2_{uk}(\boldsymbol{u}; \boldsymbol{d})$ respectively. We can approximate both of these with a large but finite sample of target locations from $\mathcal{D}$ or a large fixed grid in $\mathcal{D}$.

3.2. Parameter Uncertainty Kriging

Assuming that all covariance function parameters are known, the MSPE from kriging at an arbitrary location $\boldsymbol{u}$ is $\sigma^2_{uk}(\boldsymbol{u})$. This underestimates the MSPE when those parameters must be estimated. An approximation of the correct MSPE, which we call the parameter uncertainty kriging variance (PUK variance), is given by (Zimmerman & Cressie, 1992; Abt, 1999)

$$\mathrm{E}[Y(\boldsymbol{u}) - \widehat{Y}_{uk}(\boldsymbol{u})]^2 \approx \sigma^2_{puk}(\boldsymbol{u}; \boldsymbol{d}, \widehat{\boldsymbol{\theta}}) = \sigma^2_{uk}(\boldsymbol{u}; \boldsymbol{d}, \widehat{\boldsymbol{\theta}}) + \mathrm{tr}[\boldsymbol{A}(\boldsymbol{u}; \boldsymbol{d}, \widehat{\boldsymbol{\theta}})\boldsymbol{I}^{-1}(\boldsymbol{d}, \widehat{\boldsymbol{\theta}})]$$

where $\widehat{\boldsymbol{\theta}}$ is the maximum likelihood estimate of $\boldsymbol{\theta}$ from previously observed data, $\boldsymbol{I}^{-1}$ is the inverse Fisher information (FI) matrix, and $\boldsymbol{A} = \mathrm{var}[\partial \widehat{Y}_{uk}/\partial \boldsymbol{\theta}]$. The ijth element of the FI matrix can be derived as $\mathrm{tr}\left(\boldsymbol{C}_Z^{-1}\frac{\partial \boldsymbol{C}_Z}{\partial \theta_i}\boldsymbol{C}_Z^{-1}\frac{\partial \boldsymbol{C}_Z}{\partial \theta_j}\right)$ while $\boldsymbol{A}$ can be derived using elementary matrix calculus. From these we define the mean and maximum PUK variances as $\overline{Q}_{puk}(\boldsymbol{d}) = \frac{1}{N_t}\sum_i^{N_t}\sigma^2_{puk}(\boldsymbol{t}_i; \boldsymbol{d})$ and $Q^*_{puk}(\boldsymbol{d}) = \max_{i=1,2\ldots,N_t}\sigma^2_{puk}(\boldsymbol{t}_i; \boldsymbol{d})$ respectively.

3.3. Houston Ozone Monitoring

The Texas Commission on Environmental Quality (TCEQ) publishes a variety of environmental data for Texas, including many environmental indicators that directly relate to public health. We focus on ozone in the Houston-Galveston-Brazoria area. The TCEQ measures ozone at a network of monitoring locations in this area and publishes daily maximum eight-hour ozone concentrations (DM8s) in parts per billion (ppb) for each monitoring location. DM8s are computed as follows. First, the TCEQ creates publishes an hourly average (in ppb) for each monitoring location. Then an eight hour average is constructed at that location for each contiguous eight-hour period where all eight measurements were present for a given day. The maximum of these eight-hour averages for a given day is the published DM8. Days with less than 18 valid eight-hour averages have no published DM8. A 20 ppb increase in DM8 has been associated with an increased risk of out-of-hospital cardiact arrest, with a relative risk estimate of 1.039 (95% CI (1.005, 1.073); Ensor et al., 2013).

In August 2016 there were 44 active monitoring locations in the Houston-Galveston-Brazoria area. For each location $\boldsymbol{u}$ we compute the monthly average DM8, which we denote by $Z(\boldsymbol{u})$. At one location, MRM-3 Haden Road, there are two DM8 observations of 0 ppb in the month of August. We assume that these were data errors and omit them for the purposes of computing $Z(\boldsymbol{u})$ at that location. Of the 44 locations, one has 15 valid DM8 measurements of 31 possible valid measurements, another has 24 valid measurements, and the rest of the locations have at least 27 valid measurements.

The hypothetical design problem we consider is the addition of 100 new ozone monitoring locations to the Houston-Galveston-Brazoria monitoring network in Harris County, where Houston is located, with the goal of predicting ozone concentrations within Harris County. Of the 44 existing locations, 33 are already in Harris County, though the locations outside of the county are still useful for spatial prediction within Harris County.

Let $Z(\boldsymbol{u})$ denote the measured average DM8 at location $\boldsymbol{u}$ and let $Y(\boldsymbol{u})$ denote the true DM8. We assume that $Z(\boldsymbol{u})$ is a noisy Gaussian measurement of $Y(\boldsymbol{u})$; i.e., the data model is $Z(\boldsymbol{u}) \sim N[Y(\boldsymbol{u}), \tau^2]$. The parameter τ^2 represents variability added due to both measurement error from the instruments and potentially sampling error from measuring DM8 on less than the full 31 days in August. At the process level, we assume that $Y(\boldsymbol{u})$ is a geostatistical process with mean function $\mu(\boldsymbol{u}) = \boldsymbol{x}(\boldsymbol{u})'\boldsymbol{\beta}$ and exponential covariance function $C(\boldsymbol{u}, \boldsymbol{v}) = \sigma^2\exp(-||\boldsymbol{u} - \boldsymbol{v}||/\psi)$. We assume that any fine scale variability is measurement error and captured in the data model. We considered several possible mean functions: constant in $\boldsymbol{u}$, linear in $\boldsymbol{u}$, and quadratic in $\boldsymbol{u}$. We fit each model using maximum likelihood and found that quadratic terms were unnecessary, but linear terms did significantly help explain variation in $Y(\boldsymbol{u})$.

We use both PUK design criteria in order to choose the five new locations in Harris County: $\overline{Q}_{puk}(\boldsymbol{d})$ and $Q^*_{puk}(\boldsymbol{d})$, both defined in Section 3. We assume that the goal is the predict average DM8 in all of Harris County, so we approximate the continuous versions of $\overline{Q}_{puk}(\boldsymbol{d})$ and $Q^*_{puk}(\boldsymbol{d})$ with the finite sample versions using a grid of 1229 points, obtained by gridding up the smallest rectangle containing Harris County and throwing away all points outside of the county. We try a variety of PSO algorithms in order to select the new locations. Since the design space only allows new monitoring locations within Harris County, we modify each of the PSO algorithms we use so that any particle outside of the design space is forced to moved to the nearest point on the edge of the design space using as described in Appendix A. Existing sampling locations outside of Harris County are still used in the estimation of the model and in the construction of the PUK variances.

Table 1 contains the results of applying optimization algorithm to choosing the 100 new monitoring locations. In the table, PSO refers to the standard PSO algorithm with the usual velocity update and all of the other modifications listed in Appendix A.1. The CF modifier indicates that the velocity update is coordinate-free – see Appendix A.1 for details. The AT modifier indicates that $\omega(k)$ is adaptively tuned as described in Section 2.2. AT1 uses $R^* = 0.3$ and AT2 uses $R^* = 0.5$, while both use $c = 0.1$ and $\omega(0) = 1.2$. We use two sets of parameter values for all of the PSO algorithms: $\phi_1^{(1)} = \phi_2^{(1)} = 1.496$ (PSO1) and $\phi_1^{(2)} = \phi_2^{(2)} = \ln(2) + 1/2$ (PSO2), and when applicable the corresponding inertias are $\omega^{(1)} = 0.7298$ and $\omega^{(2)} = 1/(2\ln 2)$. We do not list any DI algorithms since the performed extremely poorly on this problem. All of the BBPSO algorithms we consider are AT, with AT1-BBPSO and AT2-BBPSO using the same parameter values of R^*, and c as AT1-PSO and AT2-PSO. All BBPSO algorithms also use $df = 1$, and each of the modications detailed in Appendix A.2. Further, the modifier CF indicates that the BBPSO algorithm uses the coordinate-free variance update and xp indicates it has a 0.5 probability of moving any given coordinate of a particle to its personal best location on that coordinate, both described in Appendix A.2. All of the PSO and BBPSO algorithms use either the global neighborhood topology, or the variant of the stochastic star neighborhood topology discussed is Section 2 with either three informants. Further, each PSO algorithm has a swarm size of 40 and is run for $K = 2000$ iterations. Additionally, we employ two other classes of alternative algorithms to compare to PSO. First, we use the genetic algorithm described in Hamada et al. (2001), with either one or two batches (B), and with two possible mutation rates (λ) and two possible mutation variances (μ). The genetic algorithms also use a population of 40, though they are only allowed to run for 1000 iterations so that after the initialization, the genetic algorithms use the same number of objective function evaluations as each of the PSO algorithms. Finally, we use an exchange algorithm (Miller & Nguyen, 1994; Nychka & Saltzman, 1998). The exchange algorithm is commonly used in spatial and spatio-temporal design problems (Nychka & Saltzman, 1998; Wikle & Royle, 1999, 2005), and uses a discrete grid as the search space and operates by considering the neighbors of the current point. We use the target grid, $\boldsymbol{t}$, as the search space and consider fifth-order and tenth-order neighbors. Each algorithm was implemented in the R programming language (R Core Team, 2016).

[WAITING ON GENETIC AND EXCHANGE ALGORITHMS TO FINISH TO ADD TO THE TABLE]

From Table 1 we immediately see that the best performing PSO algorithms tend to use the global neighborhood topology and tend not to use the coordinate free velocity/variance update. Further the first set of parameter values tends to work the best in the PSO algorithms while in the BBPSO algorithms the non-xp variants tend to outperform the xp variants, though neither of these is universally true. For both objective functions the PSO1 algorithm with the global neighborhood topology performs the best of the non AT variants and is outperformed by few of the AT variants. The best performing algorithms for $\overline{Q}_{puk}$ are, in order, AT5-PSO2, AT3-PSO1, PSO1, AT5-PSO1, and PSO2, all with the global neighborhood topology. Similarly, the best performing algorithms for Q^*_{puk} are, in order, AT3-PSO1, PSO1, PSO2, and AT5-PSO, again all with the global neighborhood topology. In Appendix B we found that the AT-BBPSO and AT-BBPSO-CF variants were competitive with the other PSO algorithms and in the case of the hardest problems seemed to be the best. The key seems to be that the AT-BBPSO variants are more robust to complicated objective

$\overline{Q}_{puk}$; Nbhd:	Global	SS3
Algorithm	logpost	logpost
PSO1	14.40	15.62
PSO2	14.45	14.87
PSO1-CF	15.53	15.94
PSO2-CF	15.77	15.56
AT3-PSO1	14.38	15.25
AT3-PSO2	14.56	15.88
AT3-PSO1-CF	15.96	15.98
AT3-PSO2-CF	15.60	15.86
AT5-PSO1	14.42	15.70
AT5-PSO2	14.32	15.70
AT5-PSO1-CF	15.85	15.96
AT5-PSO2-CF	15.95	15.87
AT3-BBPSO	14.53	15.72
AT3-BBPSOxp	15.87	15.96
AT3-BBPSO-CF	14.65	14.82
AT3-BBPSOxp-CF	14.84	15.74
AT5-BBPSO	14.65	15.74
AT5-BBPSOxp	15.21	15.32
AT5-BBPSO-CF	14.63	15.32
AT5-BBPSOxp-CF	14.52	15.71

Q^*_{puk}; Nbhd:	Global	SS3
Algorithm	logpost	logpost
PSO1	20.63	22.32
PSO2	21.03	22.63
PSO1-CF	23.54	23.69
PSO2-CF	23.16	23.96
AT3-PSO1	20.57	22.54
AT3-PSO2	23.18	23.14
AT3-PSO1-CF	23.33	23.68
AT3-PSO2-CF	24.02	23.53
AT5-PSO1	21.13	22.63
AT5-PSO2	22.11	23.28
AT5-PSO1-CF	24.00	23.97
AT5-PSO2-CF	23.63	23.58
AT3-BBPSO	22.28	23.31
AT3-BBPSOxp	22.19	23.39
AT3-BBPSO-CF	21.33	23.20
AT3-BBPSOxp-CF	22.34	23.44
AT5-BBPSO	23.49	23.51
AT5-BBPSOxp	23.25	23.15
AT5-BBPSO-CF	21.92	23.33
AT5-BBPSOxp-CF	22.76	23.25

Table 1. Simulation results for $\overline{Q}_{puk}$ and Q^*_{puk}. See text for description.

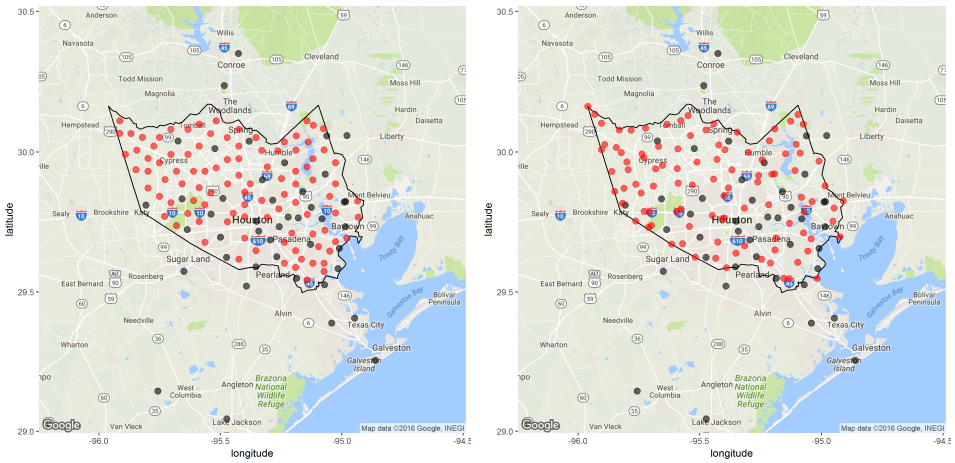

Figure 1. Best designs found according to $\overline{Q}_{puk}$ (left) and Q^*_{puk} (right). Original network points are grey, new points are red. Harris County is outlined in black. Map courtesy of Google Maps via `ggmap` (Kahle & Wickham, 2013).

surfaces with many local minima. When the objective surface is simpler, PSO and AT-PSO variants are more attractive and appear to converge faster. Our spatial design problem is apparently not complex enough for the advantages of AT-BBPSO to kick in. A similar dynamic seems to exist between the standard PSO and AT-PSO variants, namely that for problems that are simple enough PSO outperforms AT-PSO, but for hard enough problems AT-PSO becomes advantageous. For example in simulations not reported here, PSO1 outperforms all other algorithms when adding significantly less monitoring locations to the network, e.g. 20 or 50.

Figure 1 contains the best designs found using both objective functions. When using the mean kriging variance ($\overline{Q}_{puk}$), the best design spreads nodes of the network fairly evenly throughout the spatial domain. With the maximum kriging

variance (Q^*_{puk}), the best design is more erratic. Notably, there are many more nodes directly on or very close to the border of the spatial domain. Further, interior areas have a tendency to be more sparsely populated with nodes. This is unsurprising since Q^*_{puk} is worst case kriging variance over the entire county, which incentivizes putting monitoring locations near the edges of the county for two reasons. First, a point near the center of the county is likely to have monitoring locations all around it so that it is relatively easy to predict DM8 there, while a point near an edge of the county will only have monitoring locations to one side of it. Putting more monitoring locations near the edges of the county mitigates this. Second, the farther away a point is from other monitoring locations, the more the prediction depends on the model's estimate of $\boldsymbol{\beta}$ and the less it depends on the values of nearby observations. So Q^*_{puk} places a premium on a better estimate of $\boldsymbol{\beta}$, and putting more monitoring locations near the edges of the county results in smaller variances for the elements of $\widehat{\boldsymbol{\beta}}_{gls}$.

4. Discussion

[TO BE WRITTEN]

References

Abt, M (1999), 'Estimating the prediction mean squared error in Gaussian stochastic processes with exponential correlation structure,' *Scandinavian Journal of Statistics*, **26**(4), pp. 563–578.

Andrieu, C & Thoms, J (2008), 'A tutorial on adaptive MCMC,' *Statistics and Computing*, **18**(4), pp. 343–373.

Blum, C & Li, X (2008), 'Swarm intelligence in optimization,' in Blum, C & Merkle, D (eds.), *Swarm Intelligence: Introduction and Applications*, Springer.

Clerc, M (2006), 'Stagnation analysis in particle swarm optimisation or what happens when nothing happens,' Tech. rep.

Clerc, M (2010), *Particle swarm optimization*, John Wiley & Sons.

Clerc, M (2011), 'Standard particle swarm optimisation,' Tech. rep.

Clerc, M & Kennedy, J (2002), 'The particle swarm-explosion, stability, and convergence in a multidimensional complex space,' *Evolutionary Computation, IEEE Transactions on*, **6**(1), pp. 58–73.

Cressie, N & Wikle, CK (2011), *Statistics for Spatio-Temporal Data*, John Wiley & Sons.

Eberhart, RC & Shi, Y (2000), 'Comparing inertia weights and constriction factors in particle swarm optimization,' in *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, IEEE, vol. 1, pp. 84–88.

Ensor, KB, Raun, LH & Persse, D (2013), 'A case-crossover analysis of out-of-hospital cardiac arrest and air pollution,' *Circulation*, **127**(11), pp. 1192–1199.

Gelman, A, Roberts, G & Gilks, W (1996), 'Efficient Metropolis jumping rules,' *Bayesian statistics*, **5**(599-608), p. 42.

Hamada, M, Martz, H, Reese, C & Wilson, A (2001), 'Finding near-optimal bayesian experimental designs via genetic algorithms,' *The American Statistician*, **55**(3), pp. 175–181.

Hsieh, HI & Lee, TS (2010), 'A modified algorithm of bare bones particle swarm optimization,' *International Journal of Computer Science Issues*, **7**, p. 11.

Kahle, D & Wickham, H (2013), 'ggmap: Spatial visualization with ggplot2,' *The R Journal*, **5**(1), pp. 144–161.

Kennedy, J (2003), 'Bare bones particle swarms,' in *Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE*, IEEE, pp. 80–87.

Miller, AJ & Nguyen, NK (1994), 'Algorithm as 295: A fedorov exchange algorithm for d-optimal design,' *Journal of the royal statistical society. series c (applied statistics)*, **43**(4), pp. 669–677.

Miranda, V, Keko, H & Duque, AJ (2008), 'Stochastic star communication topology in evolutionary particle swarms (epso),' *International journal of computational intelligence research*, **4**(2), pp. 105–116.

Nychka, D & Saltzman, N (1998), 'Design of air-quality monitoring networks,' in *Case Studies in Environmental Statistics*, Springer, pp. 51–76.

R Core Team (2016), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria.

Tuppadung, Y & Kurutach, W (2011), 'Comparing nonlinear inertia weights and constriction factors in particle swarm optimization,' *International Journal of Knowledge-based and Intelligent Engineering Systems*, **15**(2), pp. 65–70.

Wikle, CK & Royle, JA (1999), 'Space: time dynamic design of environmental monitoring networks,' *Journal of Agricultural, Biological, and Environmental Statistics*, pp. 489–507.

Wikle, CK & Royle, JA (2005), 'Dynamic design of ecological monitoring networks for non-gaussian spatio-temporal data,' *Environmetrics*, **16**(5), pp. 507–522.

Zhang, W, Liu, Y & Clerc, M (2003), 'An adaptive pso algorithm for reactive power optimization,' in *Advances in Power System Control, Operation and Management, 2003. ASDCOM 2003. Sixth International Conference on (Conf. Publ. No. 497)*, IET, vol. 1, pp. 302–307.

Zimmerman, DL (2006), 'Optimal network design for spatial prediction, covariance parameter estimation, and empirical prediction,' *Environmetrics*, **17**(6), pp. 635–652.

Zimmerman, DL & Cressie, N (1992), 'Mean squared prediction error in the spatial linear model with estimated covariance parameters,' *Annals of the Institute of Statistical Mathematics*, **44**(1), pp. 27–43.

Stat
The ISI's Journal for the Rapid
Dissemination of Statistics Research
(wileyonlinelibrary.com) DOI: 10.100X/sta.0000

Supplemental Web Material: Adaptively-Tuned Particle Swarm Optimization with Application to Spatial Design

Received ; Accepted

A. PSO and BBPSO details

In Section 2 we introduced PSO, BBPSO, and our adaptively tuned variants of both. Here we describe in detail several modifications to both PSO and BBPSO.

A.1. PSO

The standard PSO algorithm is given by equation (1). We consider several additions and modifications to this algorithm below. Each of these modifications is combined with adaptively tuned inertia as in equation (4) to create the AT-PSO algorithms we employ.

A.1.1. Initialization: In order to initialize the swarm, the number of particles, their initial locations, and their initial velocities have to be chosen. Clerc (2011) suggests making the swarm size a function of the dimension of the search space, but notes that this is known to be suboptimal. We use their alternative suggest to use a default swarm size of 40. We assume the search space is a D-dimensional hypercube given by $\bigtimes_{j=1}^{D}[min_j, max_j]$. Then each particle's initial location is randomly generated uniformly on the cube, i.e. $\theta_{ij}(0) \overset{ind}{\sim} U(min_j, max_j)$ for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, D$. Each particle's velocity is initialized based on its location via $v_{ij}(0) \overset{ind}{\sim} U(min_j - x_{ij}(0), max_j - x_{ij}(0))$ for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, D$.

In Section 3.3 each design point is constrained to be in Harris County, TX, which is not a rectangle. We initialize the swarm by placing initial design points on the smallest rectangle containing Harris County. Design points outside of Harris County will quickly move back into the county due to the confinement strategy.

*Email:

Prepared using staauth.cls [Version: 2012/05/12 v1.00]

A.1.2. Confinement: Even though the initialization of the swarm is confined to a hypercube, nothing prevents any given particle from leaving the search space. There are a number of things that can be done in order to solve this problem and each has advantages and disadvantages depending on the situation (Helwig & Wanka, 2007). We focus on two approaches. One is to move any particle that leaves the search space to the nearest point still inside the search space and then adjust its velocity, e.g. Clerc (2011) suggests that whenever $x_{ij}(k) > max_j$, it should be set to $x_{ij}(k) = max_j$ and similarly when $x_{ij}(k) < min_j$ it should be set to $x_{ij}(k) = min_j$, and in both cases the velocity along that dimension should be reversed and halved, i.e. $v_{ij}(k) = -0.5v_{ij}(k)$. This causes particles to bounce off the boundary and move back toward the middle of the search space. This is the default strategy we use. This strategy can also be used for more complex search spaces. In Section 3.3 when a particle leaves Harris County, we move it back to the nearest point in the county and reset the particle's velocity as in the previous paragraph. In order to move the particle we use the `gNearestPoint` function from the `rgeos` `R` package (Bivand & Rundel, 2016; R Core Team, 2016).

A.1.3. Redraw Neighborhoods: In Section 2 we briefly described a variant of the stochastic star neighborhood which we use. This neighborhood is stochastic, meaning that each particle's neighbors are randomly drawn when the algorithm is initialized. After an iteration in which the best known value of the objective function is unchanged, each particle's neighbors are randomly redrawn according to the same distribution.

A.1.4. Asynchronous Updates: The way we defined PSO in equation (1) each particle can update simultaneously. This means that the algorithm is parallelizable, which is a major advantage for implementation on modern GPUs. However, asynchronously updating the particles typically results in faster converging algorithms when it is computationally feasible. In an asynchronous update from period k to $k+1$, particle i recognizes that particle $i-1$ has already updated its personal best location to $\boldsymbol{p}_{i-1}(k+1)$ by the time it is i's turn to update. So i computes its group best location taking this into account. This causes particle $i = 1$ to behave differently from particle $i = n$ since particle n always has better information in order to perform its update, so every iteration the particles are randomly reordered. More formally, before every iteration sample $o_1(k+1), o_2(k+1), \ldots, o_n(k+1)$ from $\{1, 2, \ldots, n\}$ without replacement. Then the particles update starting with o_1, o_2, etc., where the group best update becomes $\boldsymbol{g}_{o_i}(k+1) = \arg\min_{\{\boldsymbol{p}_{o_j}(k+1)|j\in\mathcal{N}_i\}} Q(\boldsymbol{p}_j(k_j^*))$ where $k_j^* = k+1$ if $j < i$ and k otherwise. We asynchronously update in all of our algorithms.

A.1.5. Coordinate Free Velocity Updates: The standard velocity update in equation (1) is well known to bias the algorithm toward locations near the coordinate axes and especially the origin (Monson & Seppi, 2005; Spears et al., 2010). In general we may not know if the true optimum is near an axis, so this behavior is undesirable. There are several alternative velocity updates available, e.g. in Monson & Seppi (2005). We use the coordinate free (CF) update suggested by Clerc (2011). First define the center of gravity for particle i to be $\boldsymbol{C}_i(k) = \boldsymbol{\theta}_i(k) + \phi_1\{\boldsymbol{p}_i(k) - \boldsymbol{\theta}_i(k)\}/3 + \phi_2\{\boldsymbol{g}_i(k) - \boldsymbol{\theta}_i(k)\}/3$. Let $\mathcal{H}_i(k)$ denote the hypersphere centered at $\boldsymbol{C}_i(k)$ with radius $||\boldsymbol{C}_i(k) - \boldsymbol{\theta}_i(k)||$ where $||.||$ denotes Euclidean distance. Then a new point $\boldsymbol{\theta}_i'(k)$ is drawn randomly from $\mathcal{H}_i(k)$ by sampling a direction and a radius, each uniformly. This is *not* the same as drawing uniformly over $\mathcal{H}_i(k)$ and in fact favors points near the center. Then the CF velocity update is given by $\boldsymbol{v}_i(k+1) = \omega\boldsymbol{v}_i(k) + \boldsymbol{x}_i'(k)$. We use both the standard and CF velocity updates in our algorithms. The standard PSO algorithm with each feature in this subsection including the CF velocity update is what Clerc (2011) calls SPSO 2011.

A.1.6. When Personal Best = Group Best: When a particle's personal best and group best locations coincide, it is often advantageous to allow the particle to explore more than usual. In the standard velocity update we do this by

removing the social term so that $\boldsymbol{v}_i(k+1) = \omega \boldsymbol{v}_i(k) + \phi_1 \boldsymbol{r}_{1i}(k) \circ \{\boldsymbol{p}_i(k) - \boldsymbol{\theta}_i(k)\}$. In the CF velocity update we change the center of gravity to ignore the social term so that $\boldsymbol{C}_i(k) = \boldsymbol{\theta}_i(k) + \phi_2\{\boldsymbol{p}_i(k) - \boldsymbol{\theta}_i(k)\}/2$.

A.2. BBPSO

The standard BBPSO algorithm was introduced by Kennedy (2003) and updates from t to $t+1$ via equation (2). We use each of the features in Section A.1 in our BBPSO algorithms, though some of them need to be modified for the BBPSO setting. We list them below along with another modification of BBPSO which we employ. Each of these modifications are combined with adaptively tuning a scale parameter as in equation (3) to create our AT-BBPSO algorithms.

A.2.1. BBPSOxp: A commonly used variant of BBPSO also introduced by Kennedy (2003) is called BBPSOxp. In this variant, each coordinate of each particle has a 50% chance of updating according to (2) and a 50% chance of moving directly to that particle's personal best location on that coordinate. In other words

$$\theta_{ij}(k+1) = \begin{cases} N\left(\frac{p_{ij}(k)+g_{ij}(k)}{2}, h_{ij}^2(k)\right) & \text{with probability } 0.5 \\ p_{ij}(k) & \text{otherwise,} \end{cases} \tag{1}$$

where $h_{ij}(k) = |p_{ij}(k) - g_{ij}(k)|$. We use both xp and non-xp versions of our BBPSO algorithms.

A.2.2. CF BBPSO: BBPSO's update also depends on the coordinate system since each coordinate of $\boldsymbol{\theta}$ gets a different standard deviation. We employ BBPSO algorithms using the default standard deviation, but also using a coordinate free standard deviation given by $h_{ij}(k) = ||\boldsymbol{p}_i(k) - \boldsymbol{g}_i(k)||$.

A.2.3. When Personal Best = Group Best in BBPSO: A downside of both BBPSO and BBPSOxp is that any particle whose personal best is currently its group best location does not move due to the definition of the standard deviation term. Several methods have been proposed to overcome this; e.g. Hsieh & Lee (2010) and Zhang et al. (2011). Zhang et al. (2011) propose using mutation and crossover operations for the group best particle. To do this, each group best particle randomly selects three other distinct particles from the entire swarm, i_1, i_2, and i_3, and updates according to

$$\theta_{ij}(k+1) = p_{i_1 j}(k) + 0.5\{p_{i_2 j}(k) - p_{i_3 j}(k)\}. \tag{2}$$

This combines easily with BBPSOxp to update each coordinate of each particle with $h_{ij}(k) = 0$ according to (2) and the rest according to (1).

B. Comparing PSO and BBPSO algorithms

In order to compare AT-BBPSO to other PSO variants, we employ a subset of test functions used in Hsieh & Lee (2010). Each function is listed in Table 1 along with the global minimum and argmin. Further description of many of these functions can be found in Clerc (2010). For each function, we set $D = 20$ so the domain of each function is $\Re^{20}$. Each function was constrained to the hypercube $[-100, 100]^D$.

We use several PSO algorithms in the simulation study. The details of each PSO algorithm are described in Appendix A. All PSO algorithms were run for 1,000 iterations and use one of three neighborhood topologies: the global topology, or

Equation	ArgMin	Minimum
$Q_1(\boldsymbol{\theta}) = \sum_{i=1}^{D} \theta_i^2$	$\boldsymbol{\theta}^* = \mathbf{0}$	$Q_1(\boldsymbol{\theta}^*) = 0$
$Q_2(\boldsymbol{\theta}) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} \theta_j\right)^2$	$\boldsymbol{\theta}^* = \mathbf{0}$	$Q_2(\boldsymbol{\theta}^*) = 0$
$Q_3(\boldsymbol{\theta}) = \sum_{i=1}^{D-1} \left[100\{\theta_{i+1} + 1 - (\theta_i + 1)^2\} + \theta_i^2\right]$	$\boldsymbol{\theta}^* = \mathbf{0}$	$Q_3(\boldsymbol{\theta}^*) = 0$
$Q_4(\boldsymbol{\theta}) = \sum_{i=1}^{D} \{\theta_i^2 - \cos(2\pi\theta_i) + 10\} - 9D$	$\boldsymbol{\theta}^* = \mathbf{0}$	$Q_4(\boldsymbol{\theta}^*) = 0$
$Q_5(\boldsymbol{\theta}) = \frac{1}{4000}\|\|\boldsymbol{\theta}\|\|^2 - \prod_{i=1}^{D} \cos\left(\frac{\theta_i}{\sqrt{i}}\right) + 1$	$\boldsymbol{\theta}^* = \mathbf{0}$	$Q_5(\boldsymbol{\theta}^*) = 0$
$Q_6(\boldsymbol{\theta}) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\|\|\boldsymbol{\theta}\|\|}\right)$ $- \exp\left\{\frac{1}{D}\sum_{i=1}^{D} \cos(2\pi\theta_i)\right\} + 20 + \exp(1)$	$\boldsymbol{\theta}^* = \mathbf{0}$	$Q_6(\boldsymbol{\theta}^*) = 0$

Table 1. Test functions for evaluating PSO algorithms. The dimension of $\boldsymbol{\theta}$ is D and $||.||$ is the Euclidean norm: $||\boldsymbol{\theta}|| = \sqrt{\sum_{i=1}^{D} \theta_i^2}$.

the stochastic star topology with either 1 (SS1) or 3 (SS3) informants. The velocity update for the PSO algorithms can either be standard or coordinate free (CF), and the inertia parameter can either be constant), deterministically adjusted (DI) or adaptively tuned (AT) with $R^* = 0.3$ (AT3) or $R^* = 0.5$ (AT5). The DI algorithms set $\alpha = 0.2 \times 1,000$ and $\beta = 2$ while the AT algorithms set $c = 0.1$ and $\omega(0) = 1.2$. Additionally, each PSO algorithm uses one of two parameter settings: either $\phi_1 = \phi_2 = \ln 2 + 1/2$ with $\omega = 1/(2\ln 2)$ if appropriate (PSO1) or $\phi_1 = \phi_2 = 1.496$ with $\omega = 0.7298$ if appropriate (PSO2). With 3 neighborhoods, 2 velocity updates, 3 inertia updates, and 2 parameter settings that yields 48 PSO algorithms. We consider AT-BBPSO algorithms with the same number of iterations and the same set of neighborhood topologies as the PSO algorithms. Additionally, each AT-BBPSO algorithm can use a CF scale parameter update or not, can use the standard or "xp" kernel, and use one of two parameter settings: $R^* = 0.3$ or $R^* = 0.5$ (AT3 or AT5 respectively), with $df = 1$, $c = 0.1$, and $\sigma(0) = 1$ in all AT-BBPSO algorithms. With 3 neighborhood topologies, 2 scale parameter updates, 2 kernels, and 2 parameter settings we consider 24 AT-BBPSO algorithms. Each algorithm was run for 40 replications of 1,000 iterations for each objective function.

Tables 2-7 contain the simulation results for objective functions 1-6 respectively (OF1, OF2, etc.). We use several measures to quantify how well each algorithm finds the global minimum. First, each table includes the mean and standard deviation of the absolute difference between the true global minimum and the algorithm's estimated global minimum across all 40 replications, denoted by Mean and SD. Second, each table includes a convergence criterion — the proportion of the replications that came within 0.01 of the true global minimum, denoted by $\widehat{P}$. Finally, $\widehat{K}$ denotes the median number of iterations until the algorithm reaches the convergence criterion; $\widehat{K} > 1000$ indicates that the algorithm did not converge in the allotted 1,000 iterations in at least of 50% of replications. Mean, $\widehat{P}$, and $\widehat{K}$ can be thought of how close on average the algorithm gets to the global minimum, what proportion of the time it converges, and long it takes to converge respectively. Values for the Mean and SD that are greater than 10,000 are omitted to keep the size of the tables manageable.

We highlight only some of the features of these tables. The first is that the CF versions of the PSO algorithms do much worse than their corresponding non-CF versions. This is unsurprising since non-CF PSO variants tend to be biased toward finding solutions at the origin and along coordinate axes and each of our objective functions were designed to have a global minimum at the origin. However, for BBPSO variants whether the scale parameter update is CF or not often makes little difference. Often the standard PSO algorithm with either parameter setting, PSO1 or PSO2, and either the Global or SS3 neighborhood performs the best or near the best. Since non-CF PSO algorithms are known to be biased toward the origin and since it is unclear if the DI-PSO and AT-PSO variants are biased to a greater, lesser, or the same degree, PSO1 and PSO2 are not good baselines for comparing to AT variants. Instead we focus on the

OF1; Nbhd:	Global				SS3				SS1			
Algorithm	Mean	SD	$\widehat{P}$	$\widehat{K}$	Mean	SD	$\widehat{P}$	$\widehat{K}$	Mean	SD	$\widehat{P}$	$\widehat{K}$
PSO1	0.00	0.00	1.00	205.5	0.00	0.00	1.00	358.5	384.26	891.68	0.32	> 1000
PSO2	0.00	0.00	1.00	113	0.00	0.00	1.00	200.5	724.66	1160.10	0.20	> 1000
PSO1-CF	173.52	201.37	0.00	> 1000	174.60	139.70	0.00	> 1000	1633.30	975.59	0.00	> 1000
PSO2-CF	164.60	117.21	0.00	> 1000	122.56	97.48	0.00	> 1000	1956.30	1074.90	0.00	> 1000
DI-PSO1	0.00	0.00	1.00	214	0.00	0.00	1.00	265.5	1467.50	1543.60	0.00	> 1000
DI-PSO2	0.00	0.00	1.00	187	0.00	0.00	1.00	233.5	1641.30	1566.30	0.00	> 1000
DI-PSO1-CF	1709.90	897.14	0.00	> 1000	694.98	346.65	0.00	> 1000	2418.30	1140.10	0.00	> 1000
DI-PSO2-CF	1535.30	848.11	0.00	> 1000	790.97	429.16	0.00	> 1000	2435.10	1173.80	0.00	> 1000
AT3-PSO1	0.00	0.00	1.00	186	0.00	0.00	1.00	263	1962.50	1777.20	0.00	> 1000
AT3-PSO2	0.00	0.00	1.00	183	0.00	0.00	1.00	247	932.07	1118.20	0.00	> 1000
AT3-PSO1-CF	329.08	260.13	0.00	> 1000	294.80	166.07	0.00	> 1000	2489.30	1040.10	0.00	> 1000
AT3-PSO2-CF	342.09	293.48	0.00	> 1000	268.43	181.01	0.00	> 1000	2096.10	1131.30	0.00	> 1000
AT5-PSO1	0.00	0.00	1.00	112	0.00	0.00	1.00	154	232.14	205.78	0.00	> 1000
AT5-PSO2	0.00	0.00	1.00	117	0.00	0.00	1.00	150.5	127.58	164.99	0.00	> 1000
AT5-PSO1-CF	165.98	133.79	0.00	> 1000	196.39	130.15	0.00	> 1000	1550.60	696.54	0.00	> 1000
AT5-PSO2-CF	118.22	77.93	0.00	> 1000	160.47	118.92	0.00	> 1000	1453.20	922.27	0.00	> 1000
AT3-BBPSO	0.00	0.00	1.00	740	0.00	0.00	1.00	756	0.00	0.00	1.00	679.5
AT3-BBPSOxp	0.00	0.00	1.00	822.5	0.00	0.00	1.00	831	0.00	0.00	1.00	694
AT3-BBPSO-CF	0.00	0.00	1.00	637	0.00	0.00	1.00	642	0.00	0.00	1.00	572.5
AT3-BBPSOxp-CF	0.00	0.00	1.00	724.5	0.00	0.00	1.00	717	0.00	0.00	1.00	607.5
AT5-BBPSO	0.00	0.00	1.00	445.5	0.00	0.00	1.00	465	0.00	0.00	1.00	451
AT5-BBPSOxp	0.00	0.00	1.00	501	0.00	0.00	1.00	511	0.00	0.00	1.00	479
AT5-BBPSO-CF	0.00	0.00	1.00	386.5	0.00	0.00	1.00	404.5	0.00	0.00	1.00	392
AT5-BBPSOxp-CF	0.00	0.00	1.00	436.5	0.00	0.00	1.00	458	0.00	0.00	1.00	420

Table 2. Simulation results for OF1. See text for description

OF2; Nbhd:	Global				SS3				SS1			
Algorithm	Mean	SD	$\widehat{P}$	$\widehat{K}$	Mean	SD	$\widehat{P}$	$\widehat{K}$	Mean	SD	$\widehat{P}$	$\widehat{K}$
PSO1	0.00	0.00	0.92	882.5	42.08	41.70	0.00	> 1000	3121.10	1900.60	0.00	> 1000
PSO2	0.00	0.00	1.00	455	0.05	0.09	0.28	> 1000	2128.20	2264.80	0.00	> 1000
PSO1-CF	1198.70	727.12	0.00	> 1000	718.76	477.04	0.00	> 1000	3467.90	1973.20	0.00	> 1000
PSO2-CF	892.45	593.77	0.00	> 1000	639.33	342.00	0.00	> 1000	2966.50	1426.90	0.00	> 1000
DI-PSO1	52.59	108.12	0.00	> 1000	225.76	210.47	0.00	> 1000	5494.70	2604.70	0.00	> 1000
DI-PSO2	151.31	169.29	0.00	> 1000	516.50	282.07	0.00	> 1000	4323.90	1872.50	0.00	> 1000
DI-PSO1-CF	3793.40	1717.40	0.00	> 1000	1401.90	678.72	0.00	> 1000	4510.60	2058.90	0.00	> 1000
DI-PSO2-CF	3873.30	1935.70	0.00	> 1000	1566.30	733.85	0.00	> 1000	3637.10	1528.20	0.00	> 1000
AT3-PSO1	0.00	0.00	1.00	478.5	0.08	0.18	0.25	> 1000	7808.60	2929.70	0.00	> 1000
AT3-PSO2	0.00	0.00	1.00	481.5	0.01	0.01	0.85	924	4071.70	1647.50	0.00	> 1000
AT3-PSO1-CF	1424.90	620.02	0.00	> 1000	1021.10	509.66	0.00	> 1000	2910.80	1528.70	0.00	> 1000
AT3-PSO2-CF	1517.60	1070.00	0.00	> 1000	883.77	458.02	0.00	> 1000	2480.30	1419.50	0.00	> 1000
AT5-PSO1	91.83	356.78	0.10	> 1000	0.40	0.85	0.12	> 1000	4323.20	1925.80	0.00	> 1000
AT5-PSO2	5.98	20.86	0.52	979	0.23	1.24	0.60	965.5	2122.30	1402.00	0.00	> 1000
AT5-PSO1-CF	1358.10	766.94	0.00	> 1000	969.11	476.07	0.00	> 1000	2737.70	1082.90	0.00	> 1000
AT5-PSO2-CF	1078.80	640.14	0.00	> 1000	878.00	410.46	0.00	> 1000	2342.40	1427.20	0.00	> 1000
AT3-BBPSO	0.06	0.03	0.00	> 1000	0.01	0.01	0.62	989	0.00	0.00	0.90	916.5
AT3-BBPSOxp	0.67	0.41	0.00	> 1000	0.02	0.01	0.15	> 1000	0.01	0.00	0.85	944
AT3-BBPSO-CF	0.03	0.01	0.05	> 1000	0.00	0.00	0.97	916	0.00	0.00	1.00	819.5
AT3-BBPSOxp-CF	0.35	0.22	0.00	> 1000	0.01	0.01	0.72	972	0.00	0.00	1.00	873.5
AT5-BBPSO	0.00	0.00	1.00	852	0.00	0.00	0.92	846.5	0.00	0.00	1.00	660
AT5-BBPSOxp	0.45	0.63	0.00	> 1000	0.05	0.13	0.50	972	0.00	0.00	1.00	718
AT5-BBPSO-CF	0.00	0.00	1.00	821	0.01	0.02	0.92	825	0.00	0.00	1.00	636.5
AT5-BBPSOxp-CF	0.24	0.20	0.00	> 1000	0.04	0.08	0.52	962	0.00	0.00	1.00	682

Table 3. Simulation results for OF2. See text for description

OF3; Nbhd:	Global				SS3				SS1			
Algorithm	Mean	SD	$\widehat{P}$	$\widehat{K}$	Mean	SD	$\widehat{P}$	$\widehat{K}$	Mean	SD	$\widehat{P}$	$\widehat{K}$
PSO1	51.84	81.96	0.00	> 1000	32.87	35.51	0.00	> 1000			0.00	> 1000
PSO2	24.09	40.64	0.00	> 1000	34.97	56.51	0.00	> 1000			0.00	> 1000
PSO1-CF			0.00	> 1000			0.00	> 1000			0.00	> 1000
PSO2-CF			0.00	> 1000			0.00	> 1000			0.00	> 1000
DI-PSO1	73.91	118.06	0.00	> 1000	138.79	237.03	0.00	> 1000			0.00	> 1000
DI-PSO2	73.15	101.70	0.00	> 1000	285.84	515.43	0.00	> 1000			0.00	> 1000
DI-PSO1-CF			0.00	> 1000			0.00	> 1000			0.00	> 1000
DI-PSO2-CF			0.00	> 1000			0.00	> 1000			0.00	> 1000
AT3-PSO1	42.13	62.63	0.00	> 1000	27.96	32.20	0.00	> 1000			0.00	> 1000
AT3-PSO2	29.68	49.48	0.02	> 1000	31.32	39.42	0.00	> 1000			0.00	> 1000
AT3-PSO1-CF			0.00	> 1000			0.00	> 1000			0.00	> 1000
AT3-PSO2-CF			0.00	> 1000			0.00	> 1000			0.00	> 1000
AT5-PSO1	63.40	150.86	0.02	> 1000	27.31	45.22	0.00	> 1000			0.00	> 1000
AT5-PSO2	45.50	127.02	0.00	> 1000	21.57	33.17	0.00	> 1000			0.00	> 1000
AT5-PSO1-CF			0.00	> 1000			0.00	> 1000			0.00	> 1000
AT5-PSO2-CF			0.00	> 1000			0.00	> 1000			0.00	> 1000
AT3-BBPSO	286.56	733.87	0.00	> 1000	163.98	568.42	0.00	> 1000	55.96	68.75	0.00	> 1000
AT3-BBPSOxp	152.06	495.54	0.00	> 1000	53.66	40.22	0.00	> 1000	38.12	36.67	0.00	> 1000
AT3-BBPSO-CF	121.97	358.24	0.00	> 1000	199.26	573.65	0.00	> 1000	36.63	52.87	0.00	> 1000
AT3-BBPSOxp-CF	390.89	857.50	0.00	> 1000	39.49	42.56	0.00	> 1000	31.37	35.33	0.00	> 1000
AT5-BBPSO	192.88	490.38	0.00	> 1000	63.77	83.78	0.00	> 1000	84.38	132.31	0.00	> 1000
AT5-BBPSOxp	124.92	235.06	0.00	> 1000	41.88	46.31	0.00	> 1000	34.63	36.02	0.00	> 1000
AT5-BBPSO-CF	325.19	767.79	0.00	> 1000	115.76	398.59	0.00	> 1000	71.89	121.28	0.00	> 1000
AT5-BBPSOxp-CF	229.62	646.40	0.00	> 1000	45.22	55.65	0.00	> 1000	37.75	59.87	0.00	> 1000

Table 4. Simulation results for OF3. See text for description

OF4; Nbhd:	Global				SS3				SS1			
Algorithm	Mean	SD	$\widehat{P}$	$\widehat{K}$	Mean	SD	$\widehat{P}$	$\widehat{K}$	Mean	SD	$\widehat{P}$	$\widehat{K}$
PSO1	3.78	1.53	0.00	> 1000	0.90	1.12	0.47	> 1000	411.24	913.72	0.00	> 1000
PSO2	8.82	4.25	0.00	> 1000	2.62	1.62	0.07	> 1000	814.28	1218.90	0.00	> 1000
PSO1-CF	504.88	318.43	0.00	> 1000	337.63	182.82	0.00	> 1000	1951.90	1010.30	0.00	> 1000
PSO2-CF	337.14	160.49	0.00	> 1000	253.11	127.14	0.00	> 1000	1973.70	971.52	0.00	> 1000
DI-PSO1	7.68	2.34	0.00	> 1000	3.02	1.76	0.07	> 1000	1499.60	1649.80	0.00	> 1000
DI-PSO2	11.15	4.86	0.00	> 1000	4.71	2.28	0.00	> 1000	1497.30	1511.10	0.00	> 1000
DI-PSO1-CF	1796.40	910.83	0.00	> 1000	780.47	623.55	0.00	> 1000	2276.80	1045.00	0.00	> 1000
DI-PSO2-CF	1652.70	826.63	0.00	> 1000	717.85	335.94	0.00	> 1000	2295.10	941.03	0.00	> 1000
AT3-PSO1	4.76	2.15	0.00	> 1000	2.16	1.76	0.17	> 1000	1584.60	1101.90	0.00	> 1000
AT3-PSO2	6.85	2.12	0.00	> 1000	2.43	1.49	0.05	> 1000	772.81	909.62	0.00	> 1000
AT3-PSO1-CF	963.98	737.17	0.00	> 1000	470.14	242.39	0.00	> 1000	2032.40	937.66	0.00	> 1000
AT3-PSO2-CF	721.45	403.80	0.00	> 1000	358.45	172.38	0.00	> 1000	1780.40	650.79	0.00	> 1000
AT5-PSO1	7.44	3.39	0.00	> 1000	3.50	1.66	0.00	> 1000	108.91	112.21	0.00	> 1000
AT5-PSO2	10.01	4.71	0.00	> 1000	5.02	2.70	0.00	> 1000	122.32	191.30	0.00	> 1000
AT5-PSO1-CF	495.75	240.51	0.00	> 1000	454.87	259.16	0.00	> 1000	1355.10	663.30	0.00	> 1000
AT5-PSO2-CF	425.66	293.16	0.00	> 1000	363.88	192.89	0.00	> 1000	1209.60	810.10	0.00	> 1000
AT3-BBPSO	3.22	1.39	0.05	> 1000	0.46	0.53	0.30	> 1000	1.15	1.23	0.35	> 1000
AT3-BBPSOxp	0.05	0.15	0.00	> 1000	0.03	0.01	0.00	> 1000	0.14	0.35	0.65	974.5
AT3-BBPSO-CF	3.59	1.63	0.00	> 1000	0.24	0.52	0.80	898.5	1.17	1.15	0.35	> 1000
AT3-BBPSOxp-CF	0.15	0.34	0.75	985	0.01	0.00	0.85	973.5	0.12	0.32	0.88	865
AT5-BBPSO	4.59	1.93	0.00	> 1000	0.62	0.79	0.52	633.5	1.14	1.18	0.40	> 1000
AT5-BBPSOxp	0.17	0.43	0.85	656	0.00	0.00	1.00	672	0.12	0.32	0.88	633.5
AT5-BBPSO-CF	3.54	1.85	0.00	> 1000	0.59	0.74	0.52	583.5	1.57	1.30	0.22	> 1000
AT5-BBPSOxp-CF	0.09	0.36	0.92	591.5	0.00	0.00	1.00	614	0.10	0.29	0.90	598.5

Table 5. Simulation results for OF4. See text for description

OF5; Nbhd:	Global				SS3				SS1			
Algorithm	Mean	SD	$\widehat{P}$	$\widehat{K}$	Mean	SD	$\widehat{P}$	$\widehat{K}$	Mean	SD	$\widehat{P}$	$\widehat{K}$
PSO1	0.03	0.03	0.30	> 1000	0.01	0.01	0.57	441	0.52	0.65	0.07	> 1000
PSO2	0.02	0.02	0.38	> 1000	0.01	0.01	0.60	205	0.56	0.60	0.07	> 1000
PSO1-CF	0.89	0.20	0.00	> 1000	0.96	0.14	0.00	> 1000	1.45	0.28	0.00	> 1000
PSO2-CF	0.94	0.15	0.00	> 1000	0.99	0.09	0.00	> 1000	1.48	0.31	0.00	> 1000
DI-PSO1	0.02	0.02	0.28	> 1000	0.01	0.01	0.65	258.5	1.27	0.45	0.00	> 1000
DI-PSO2	0.02	0.03	0.38	> 1000	0.01	0.02	0.65	239	1.36	0.42	0.00	> 1000
DI-PSO1-CF	1.42	0.26	0.00	> 1000	1.16	0.08	0.00	> 1000	1.60	0.30	0.00	> 1000
DI-PSO2-CF	1.33	0.18	0.00	> 1000	1.15	0.09	0.00	> 1000	1.58	0.28	0.00	> 1000
AT3-PSO1	0.02	0.02	0.42	> 1000	0.01	0.01	0.78	253	1.37	0.28	0.00	> 1000
AT3-PSO2	0.02	0.02	0.32	> 1000	0.01	0.01	0.82	239.5	1.44	0.56	0.00	> 1000
AT3-PSO1-CF	1.09	0.22	0.00	> 1000	1.04	0.10	0.00	> 1000	1.64	0.26	0.00	> 1000
AT3-PSO2-CF	1.03	0.21	0.00	> 1000	1.04	0.11	0.00	> 1000	1.61	0.26	0.00	> 1000
AT5-PSO1	0.03	0.03	0.38	> 1000	0.01	0.01	0.52	167	0.93	0.28	0.00	> 1000
AT5-PSO2	0.03	0.03	0.35	> 1000	0.01	0.01	0.65	132.5	0.85	0.27	0.00	> 1000
AT5-PSO1-CF	0.97	0.25	0.00	> 1000	1.02	0.10	0.00	> 1000	1.42	0.21	0.00	> 1000
AT5-PSO2-CF	0.91	0.19	0.00	> 1000	0.97	0.10	0.00	> 1000	1.38	0.21	0.00	> 1000
AT3-BBPSO	0.01	0.01	0.55	875.5	0.00	0.01	0.85	555.5	0.00	0.01	0.82	535.5
AT3-BBPSOxp	0.01	0.01	0.72	631	0.00	0.00	1.00	623.5	0.00	0.00	0.95	566.5
AT3-BBPSO-CF	0.02	0.02	0.35	> 1000	0.00	0.01	0.85	475	0.01	0.01	0.68	472
AT3-BBPSOxp-CF	0.01	0.01	0.78	518.5	0.00	0.00	1.00	549	0.00	0.00	0.97	469.5
AT5-BBPSO	0.02	0.02	0.40	> 1000	0.00	0.01	0.92	366.5	0.01	0.01	0.72	406.5
AT5-BBPSOxp	0.01	0.01	0.78	387	0.00	0.00	0.97	400.5	0.00	0.01	0.90	380.5
AT5-BBPSO-CF	0.01	0.02	0.52	478.5	0.00	0.01	0.82	308.5	0.00	0.01	0.78	307
AT5-BBPSOxp-CF	0.01	0.01	0.72	370.5	0.00	0.00	1.00	342.5	0.00	0.01	0.80	349

Table 6. Simulation results for OF5. See text for description

OF6; Nbhd:	Global				SS3				SS1			
Algorithm	Mean	SD	$\widehat{P}$	$\widehat{K}$	Mean	SD	$\widehat{P}$	$\widehat{K}$	Mean	SD	$\widehat{P}$	$\widehat{K}$
PSO1	20.01	0.02	0.00	> 1000	20.13	0.07	0.00	> 1000	20.26	0.10	0.00	> 1000
PSO2	20.00	0.01	0.00	> 1000	20.21	0.11	0.00	> 1000	20.37	0.13	0.00	> 1000
PSO1-CF	20.67	0.14	0.00	> 1000	20.59	0.11	0.00	> 1000	20.56	0.13	0.00	> 1000
PSO2-CF	20.68	0.19	0.00	> 1000	20.62	0.11	0.00	> 1000	20.62	0.13	0.00	> 1000
DI-PSO1	20.02	0.03	0.00	> 1000	20.06	0.04	0.00	> 1000	20.11	0.05	0.00	> 1000
DI-PSO2	20.13	0.14	0.00	> 1000	20.18	0.11	0.00	> 1000	20.23	0.09	0.00	> 1000
DI-PSO1-CF	20.43	0.11	0.00	> 1000	20.33	0.10	0.00	> 1000	20.29	0.11	0.00	> 1000
DI-PSO2-CF	20.48	0.13	0.00	> 1000	20.38	0.11	0.00	> 1000	20.32	0.11	0.00	> 1000
AT3-PSO1	20.00	0.00	0.00	> 1000	20.01	0.01	0.00	> 1000	20.00	0.00	0.00	> 1000
AT3-PSO2	20.02	0.04	0.00	> 1000	20.08	0.06	0.00	> 1000	20.00	0.00	0.00	> 1000
AT3-PSO1-CF	20.06	0.06	0.00	> 1000	20.14	0.07	0.00	> 1000	20.00	0.00	0.00	> 1000
AT3-PSO2-CF	20.23	0.09	0.00	> 1000	20.16	0.09	0.00	> 1000	19.68	2.06	0.00	> 1000
AT5-PSO1	20.01	0.02	0.00	> 1000	20.07	0.05	0.00	> 1000	20.22	0.09	0.00	> 1000
AT5-PSO2	20.12	0.10	0.00	> 1000	20.27	0.08	0.00	> 1000	20.33	0.11	0.00	> 1000
AT5-PSO1-CF	20.33	0.10	0.00	> 1000	20.33	0.10	0.00	> 1000	20.32	0.11	0.00	> 1000
AT5-PSO2-CF	20.38	0.12	0.00	> 1000	20.32	0.13	0.00	> 1000	20.32	0.11	0.00	> 1000
AT3-BBPSO	17.22	8.02	0.00	> 1000	5.23	9.13	0.00	> 1000	18.71	6.28	0.00	> 1000
AT3-BBPSOxp	18.34	6.18	0.00	> 1000	17.58	6.79	0.00	> 1000	20.42	0.21	0.00	> 1000
AT3-BBPSO-CF	15.63	9.14	0.07	> 1000	0.53	3.30	0.42	> 1000	13.68	9.62	0.22	> 1000
AT3-BBPSOxp-CF	14.75	9.19	0.00	> 1000	10.35	9.46	0.00	> 1000	20.40	0.33	0.00	> 1000
AT5-BBPSO	17.76	7.56	0.15	> 1000	5.95	8.70	0.65	684.5	19.04	5.54	0.07	> 1000
AT5-BBPSOxp	18.22	6.18	0.10	> 1000	17.65	5.79	0.05	> 1000	20.17	1.84	0.00	> 1000
AT5-BBPSO-CF	18.33	6.76	0.07	> 1000	2.06	6.25	0.90	628	14.47	9.01	0.22	> 1000
AT5-BBPSOxp-CF	14.49	9.15	0.28	> 1000	10.03	8.80	0.32	> 1000	18.77	4.41	0.00	> 1000

Table 7. Simulation results for OF6. See text for description

CF versions of all algorithms. The difference between the CF and non-CF version of each algorithm does serve as a possible measure of the origin-seeking bias of the non-CF version of the algorithm, however.

Narrowing in on the CF algorithms, the AT variants of the PSO-CF algorithms are a mixed bag relative to their pure PSO counterparts. Sometimes the AT version performs better, sometimes not, and often they are essentially indistinguishable. Both classes of algorithms tend to perform better than their DI-PSO-CF counterparts, however, and it is clear for all PSO algorithms that the Global and SS3 neighborhoods both tend to be significantly superior than the SS1 neighborhood. The real story, however, is the performance of the AT-BBPSO-CF variants. Typically they are the best performing CF variant, and often they are even competitive with the best non-CF PSO variants. *Which* AT-BBPSO-CF algorithm performs best depends on the situation. For relatively easy problems such as for OF1 and OF2, AT5 tends to be better than AT3, BBPSO tends to be better than BBPSOxp, and the Global or SS1 neighborhoods tend to be the best across all three measures, so AT5-BBPSO-CF with the Global or SS1 neighborhoods appear attractive. For more complex objective functions such as OF4-OF6, any of the AT-BBPSO-CF variants could perform the best, and whether e.g. AT3 or AT5 is better often interacts with whether the BBPSO or BBPSOxp kernel is used. There does not appear to be a hard an fast rule to abide by, so in more complex optimization problems we recommend trying several AT-BBPSO-CF variants, and perhaps even non-CF variants. Our default recommendation is to try AT5-BBPSO-CF with the global neighborhood. In the case where trying multiple algorithms is worthwhile, e.g. for particularly difficult optimization problems, we recommend experimenting with the neighborhood before experimenting with any of the other knobs on these algorithms. Changing the neighborhood often yields the largest differences between algorithms in our simulations, though sometimes changing the kernel (BBPSO vs. BBPSOxp) or R^* (e.g. AT3 vs. AT5) can also sometimes yield large gains.

The DI-PSO and AT-PSO algorithms are similar conceptually, but often yield very different results. DI-PSO deterministically reduces the inertia parameter over time in the same manner given a fixed set of parameter values (α and β), while AT-PSO dynamically adjusts the inertia parameter to hit a target improvement rate. Figure 1 plots the inertia over time for the DI-PSO algorithm with $\alpha = 200$ and $\beta = 1$, and observed inertia over time for one replication of the AT5-PSO2-CF algorithm with the SS3 neighborhood for OF1 and one replication for OF6. DI-PSO smoothly decreases its inertia with a slowly decreasing rate, while AT5-PSO2-CF behaves very differently depending on the objective function. For OF1 it drops the inertia to about to about 0.7 then bounces back up to about 0.8 and fluctuates around that point. This is pretty typical behavior for the inertia parameter of AT-PSO — it tends to bounce around a level which is approximately the average over time of the DI-PSO's inertia, though lower values of R^* will result in higher inertias. In this way, AT-PSO alternates periods of exploration (relatively high inertia) and periods of exploitation (relatively low inertia). The main exception to this pattern is when AT-PSO converges around a local minimum. In this case, inertia plummets to zero as the particles settle down. This is precisely what happens for OF6 in Figure 1, though in this case the minimum is not global — Table 7 indicates the algorithm never converged to the global min. In optimization problems with multiple local optima, both AT-PSO can exhibit this behavior and prematurely converge to a local minima, so they may not be advantageous for those problems. In theory we expect this behavior for AT-BBPSO as well, but as Table 7 indicates, some variants of the AT-BBPSO algorithms were able to get significantly closer to the global minimum.

Figure 2 displays the scale parameter over time for one replication of the AT5-BBPSO-CF algorithm for each of the objective functions we considered with the SS3 neighborhood. Notably, they all result in similar scale parameter dynamics. This holds up remarkably well across replications, BBPSO vs. BBPSOxp, and CF vs. not, such that it may be possible to pick a one-size-fits-all deterministic progression of the scale parameter that matches the algorithm to an AT algorithm with a specific target improvement rate, R^*. One key source of variation that is sometimes more pronounced than in Figure 2 is that for some objective functions the AT algorithms first increase the scale parameter before following the exponentially decreasing progression. This flexibility to adapt to the objective function may not be

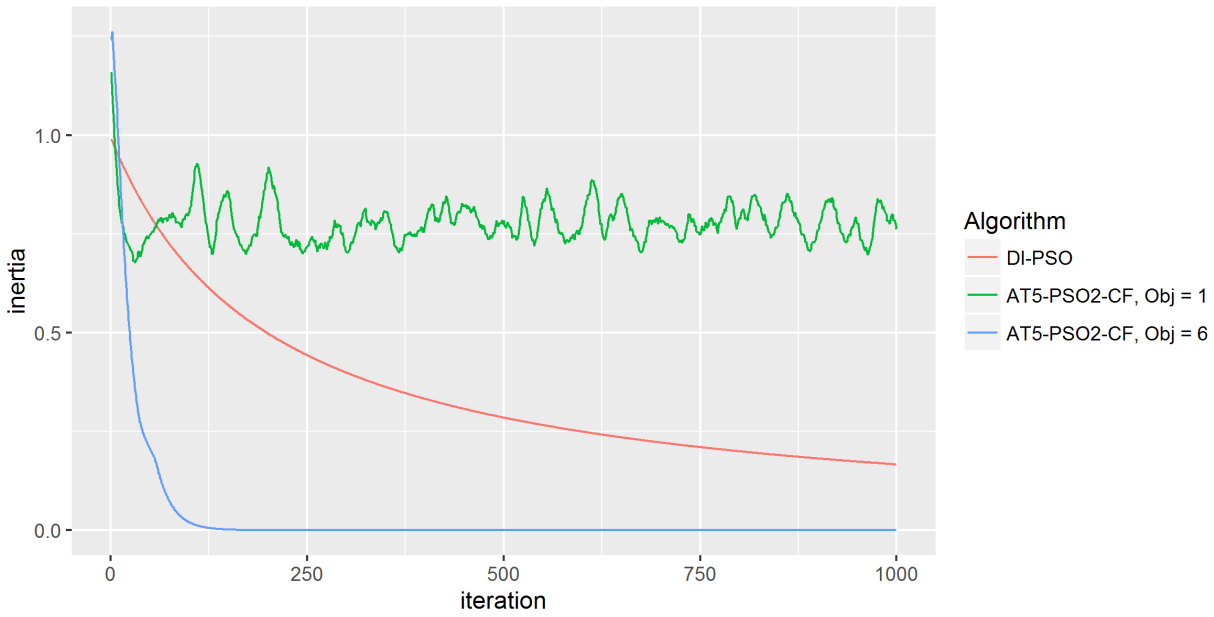

Figure 1. Inertia over time for the DI-PSO algorithm with $\alpha = 200$ and $\beta = 1$, and for one replication of the AT-PSO-0.5 algorithm with the SS3 neighborhood for each of OFs 1 and 6.

worth sacrificing for the one-size-fits all approach. Rather, we highlight this possibility as a possible avenue for further understanding the AT-BBPSO algorithms.

References

Bivand, R & Rundel, C (2016), *rgeos: Interface to Geometry Engine - Open Source (GEOS)*, r package version 0.3-21.

Clerc, M (2010), *Particle swarm optimization*, John Wiley & Sons.

Clerc, M (2011), 'Standard particle swarm optimisation,' Tech. rep.

Helwig, S & Wanka, R (2007), 'Particle swarm optimization in high-dimensional bounded search spaces,' in *2007 IEEE Swarm Intelligence Symposium*, IEEE, pp. 198–205.

Hsieh, HI & Lee, TS (2010), 'A modified algorithm of bare bones particle swarm optimization,' *International Journal of Computer Science Issues*, **7**, p. 11.

Kennedy, J (2003), 'Bare bones particle swarms,' in *Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE*, IEEE, pp. 80–87.

Monson, CK & Seppi, KD (2005), 'Exposing origin-seeking bias in pso,' in *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, ACM, pp. 241–248.

R Core Team (2016), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria.

Spears, WM, Green, DT & Spears, DF (2010), 'Biases in particle swarm optimization,' *International Journal of Swarm Intelligence Research*, **1**(2), pp. 34–57.

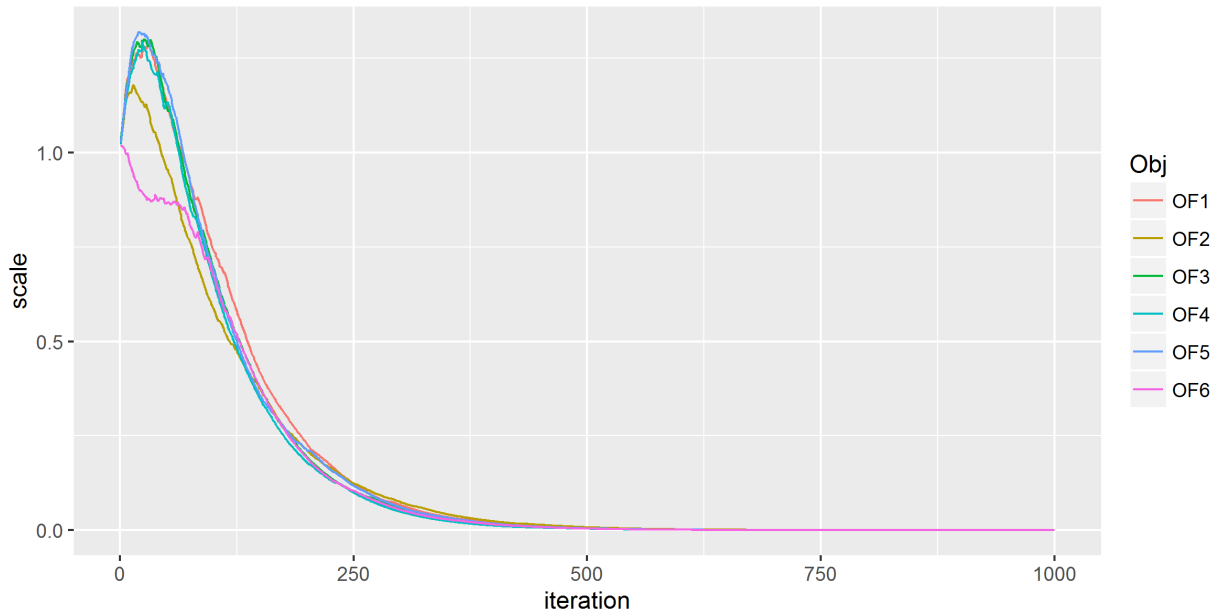

Figure 2. Scale parameter over time for one replication of the the AT5-BBPSO-CF algorithm with the SS3 neighborhood for each of OF1–OF6.

Zhang, H, Kennedy, DD, Rangaiah, GP & Bonilla-Petriciolet, A (2011), 'Novel bare-bones particle swarm optimization and its performance for modeling vapor–liquid equilibrium data,' *Fluid Phase Equilibria*, **301**(1), pp. 33–45.