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 attractive for complex optimization problems. We propose using PSO to solve spatial design problems, e.g. choosing new locations to add to an existing monitoring network. Additionally, we introduce two new classes of PSO algorithms that perform well in a wide variety of circumstances, called adaptively-tuned PSO and adaptively-tuned bare bones PSO. 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-hospital cardiac arrest rates. Copyright © 0000 John Wiley & Sons, Ltd.

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

1. Introduction

A common step in many statistical problems is the optimization of some objective function. Gradient-based algorithms solve many of these problems efficiently, but some problems are too difficult. In spatial design problems, the objective function consists of some design criterion typically related to the variability of predictions, which is a complex function of the design points often with multiple local and global optima, and this function may be defined over a complex domain. Gradient-based methods are typically inefficient for these problems and often fail. We propose using particle swarm optimization (PSO) as a more robust alternative for design problems and more generally for other statistical optimization problems where gradient-based methods fail. We illustrate the usefulness of PSO as a tool for solving design problems by applying several PSO algorithms 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), and ozone monitoring is an essential tool for determining when and where populations are at risk. For a fixed network size or cost, optimized network design improves the quality of information available to policy makers using the network.

Particle swarm optimization (PSO) refers to a large class of heuristic optimization algorithms that that rely on an analogy with animal flocking behavior and are typically more robust than many alternatives; Clerc & Kennedy (2002);

*Email:

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

Blum & Li (2008); Clerc (2010). This robustness makes them attractive for more difficult optimization problems such as the design problems we discuss, especially when near optimal solutions are tolerable. We introduce two new classes of PSO algorithms, called adaptively-tuned PSO (AT-PSO), and adaptively-tuned bare bones PSO (AT-BBPSO), which exploit an analogy with a class of adaptive Markov chain Monte Carlo algorithms in order to tune a crucial parameter of the algorithm adaptively based on the state of the particle swarm. We show that the resulting algorithms are competitive with and in some cases significantly better than many PSO alternatives.

The remainder of the paper is as follows. Section 2 introduces various PSO, AT-PSO, and AT-BBPSO algorithms, whereas Section 3 briefly discusses the results of a simulation study comparing them. Section 4 introduces a generic spatial design problem and the Houston area ozone problem as an instance of that problem, and compares several PSO algorithms and some alternatives to solving the problem. Finally, Section 5 discusses our results and concludes.

2. Particle swarm optimization

We briefly describe PSO here; see Blum & Li (2008) for a comprehensive introduction, Clerc (2010) for additional details, and Clerc (2012) 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 $Q(\boldsymbol{p}_i(k)) \leq Q(\boldsymbol{\theta}_i(l))$ for any $k \geq l$. The group best location is defined with respect to some neighborhood $\mathcal{N}_i$ of particles for 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 (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 with 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 force each particle to communicate with fewer of the other particles. In addition to the global topology, we use a variant of the stochastic star topology from 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). Several modifications can be made to PSO and BBPSO to improve performance, and Section S1 of the supporting materials details which ones we use. In the next two subsections we introduce two novel algorithms: adaptively-tuned BBPSO and adaptively-tuned PSO respectively.

2.1. Adaptively-tuned BBPSO

BBPSO adapts the effective size of the swarm's search space 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 that 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 how the proposal standard deviation is tuned in 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 $\#A$ denotes the cardinality of the set A. Adaptively-tuned BBPSO (AT-BBPSO) compares $R(k)$ to a target improvement rate R^* and adjusts $\sigma^2(k)$ accordingly. Specifically 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 a user specified parameter that controls the speed of adaptation. We use a t kernel parameterized in terms of location and scale parameters with $df = 1$ instead of a Gaussian because this works 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 that work well, which is unsurprising given that random walk Metropolis MCMC algorithms work best with R^* in a similar range; Gelman et al. (1996). The parameter c controls the speed of adaptation so that larger values of c cause the algorithm to adapt $\sigma^2(k)$ faster. We find that $c = 0.1$ works well with our other parameter settings, though anything within an order of magnitude yields similar results and in our experience with these algorithms, there does not appear to be much to gain from optimizing the choice of this parameter. We use $\sigma^2(0) = 1$ to initialize the algorithm at the standard BBPSO algorithm.

Both using a t kernel and adding a fixed scale parameter have been discussed in the BBPSO literature, but as Kennedy (2003) notes, in the standard BBPSO setting $\sigma^2 = 1$ yields the best algorithms. Hsieh & Lee (2010) propose a variant that deterministically changes σ^2, starting it below one and gradually increasing it until a final set of iterations with $\sigma^2 = 1$. They also suggest tuning σ^2 dynamically, 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 locations farther away from these areas 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 Section S1 of the supporting materials. In Section 3 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. In particular, AT-BBPSO typically drastically improves on BBPSO, and for the most difficult objective functions, the best algorithms are AT-BBPSO variants.

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 tune this parameter based 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 magnitudes of the velocities evolve over time. We can use the mechanism we used to tune $\sigma^2(k)$ in AT-PSO to tune $\omega(k)$. Formally, AT-PSO is the same as PSO in (1), but at the end of an iteration it adds a step to update $\omega(k)$ via

$$\log \omega(k+1) = \log \omega(k) + c \times \{R(k+1) - R^*\}, \tag{4}$$

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. This step can be combined with almost any PSO variant, and we combine it with each of the modifications listed in Section S1 of the supporting materials.

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 $\omega(k) = 1/\left\{1 + (t/\alpha)^\beta\right\}$ where α and β are user-defined parameters. They suggest setting 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 setting 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, which suggests that 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 based on local conditions and thus it speeds up convergence by allowing the particles that do improve to make larger improvements; note, however, 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 periods of exploration and exploitation, e.g. see Figure 1 in Section 3

Zhang et al. (2003) introduce a similar algorithm where ω is constant while ϕ_1 and ϕ_2 vary across both time and 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 from our algorithm is that each particle adapts differently based on its own state while in AT-PSO each particle adapts in the same way based on the state of the entire swarm. The next section describes the

results of a simulation study including several PSO, BBPSO, AT-PSO, and AT-BBPSO algorithms and demonstrates some of the behavior detailed above.

3. 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 $D = 20$ and constrain it to the hypercube $[-100, 100]^D$.

All PSO algorithms we use in the simulation study were run for 1,000 iterations and use one of three neighborhood topologies: the global topology, or the stochastic star topology with either 1 (SS1) or 3 (SS3) informants. The standard PSO velocity update is known to be biased in favor of solutions at the origin, Monson & Seppi (2005) and Spears et al. (2010), so we use both the normal update and a coordinate free (CF) update described in Section S1.1 of the supporting materials in order to make a fair comparison. Further, the inertia parameter can either be constant, deterministically adjusted (DI) or adaptively-tuned (AT) with $R^* = 0.3$ (AT1) or $R^* = 0.5$ (AT2). 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, though we do not report the results of any BBPSO algorithms because they perform significantly worse than any of the algorithms we consider. Each AT-BBPSO algorithm can use a CF scale parameter update described in Section S1.2 of the supporting materials or the usual update in (2), can use the standard or "xp" kernel described in Section S1.2, and use one of two parameter settings: $R^* = 0.3$ or $R^* = 0.5$ (AT1 or AT2 respectively). We set $df = 1$, $c = 0.1$, 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. See Section S1 of the supporting materials for the full details of each of the algorithms we employ, including explanation of the variations discussed in the preceding paragraph.

Tables S2.1-S2.6 in the supporting materials contain the simulation results for objective functions 1-6 respectively (OF1, OF2, etc.). We highlight only some of the results of those tables here. The first is that the CF version of a given PSO algorithm tends to be much worse than the non-CF version. This is not surprising given the origin-seeking bias of the standard PSO velocity update discussed in the previous paragraph since our test functions all obtain their minimum at the origin. A second major feature is that the global and SS3 neighborhood topologies both outperform the SS1 topology in most cases, though which of the global and SS3 topologies is better depends on the objective function and PSO algorithm. Turning to our AT variants, PSO and AT-PSO appear to be very comparable. Sometimes the AT-PSO version of an algorithm is better and sometimes the PSO version is better, and similarly sometimes the best AT-PSO algorithm beats the best PSO algorithm and vice versa. The only consistent pattern is that both PSO and AT-PSO both tend to outperform DI-PSO, though the DI-PSO algorithm is occasionally competitive. The difficulty of choosing a good progression for DI-PSO is a key factor here — the DI-PSO algorithm we use is the only one that performed reasonably well on any of the problems we tried, and it still is often significantly worse than both PSO and AT-PSO.

The DI-PSO and AT-PSO algorithms are similar conceptually, but often yield 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 AT2-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 AT2-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 between 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 S2.6 indicates the algorithm never converged to the global minimum. In optimization problems with multiple local optima, AT-PSO can exhibit this behavior and prematurely converge to a local minima, so it may not be advantageous for those problems. In theory we expect this behavior for AT-BBPSO as well, but it turns out that some variants of the AT-BBPSO algorithms were able to get significantly closer to the global minimum for OF6, which has many local optima.

The AT-BBPSO algorithms are fairly competitive for most of the objective functions we consider, but AT-BBPSO strongly outperforms the alternatives in a few cases where the objective function is particularly difficult to optimize. This is clearest in Table S2.6 for OF6, but in Tables S2.4 and S2.5 for OF5 and OF6, respectively, the best performing AT-BBPSO variants are significantly better than the best performing PSO, AT-PSO, and DI-PSO variants. AT-BBPSO algorithms also buck a couple of trends. First, the difference between the CF and non-CF variants is much less stark and is often reversed — the best algorithms for OF6, for example, are AT-BBPSO-CF variants. Second, the SS1 neighborhood topology often works well for AT-BBPSO variants. There does not appear to be a hard and fast rule to abide by for deciding *which* AT-BBPSO variant is best, but one of AT2-BBPSO-CF or AT2-BBPSOxp-CF with either the global or SS3 neighborhood tends to be the best AT-BBPSO variant for the problems we consider in this section.

Figure 2 displays the scale parameter over time for one replication of the AT2-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, CF vs. not, and neighborhood topologies 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 exponentially decaying. This flexibility to adapt to the objective function may not be worth sacrificing for the one-size-fits all approach. Rather, we highlight this possibility as a potential avenue for further understanding the AT-BBPSO algorithms.

AT-PSO and AT-BBPSO both improve on PSO in at least some contexts based on the results of this section. Notably for more difficult problems with, e.g., many local optima, AT-BBPSO variants are capable of vastly outperforming other PSO variants. In the next section, we turn to applying these algorithms to the practical problem of spatial network design.

4. The Spatial Design Problem

Suppose we are interested in predicting a 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 of interest here is to add N_d new sampling locations in order to optimize the amount we learn about $Y(\boldsymbol{u})$ at the target locations. See Müller (2007), Mateu & Müller (2012), and Le & Zidek (2006), Section 11, for general discussions of spatial design problems. 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 $\boldsymbol{u}, \boldsymbol{v} \in \mathcal{D}$. Typically not all covariates are 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 and/or nugget effects. Typically $\boldsymbol{\beta}$, τ^2, and $C(\cdot, \cdot)$ are unknown and must be estimated.

To completely specify the problem we need to choose 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 some function of these variances, e.g., to minimize the mean kriging variance or the maximum kriging variance over all target locations.

4.1. Universal Kriging

In universal kriging, $\boldsymbol{\beta}$, τ^2, and the parameters of $C(\cdot, \cdot)$ are estimated, but only $\boldsymbol{\beta}$ is treated as uncertain. 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

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

(Cressie & Wikle, 2011, Section 4.1.2) 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 $\boldsymbol{\beta}$. For ease of notation 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 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 $\mathcal{D}$.

4.2. 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

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

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 $\boldsymbol{A} = \text{var}[\partial\widehat{Y}_{uk}/\partial\boldsymbol{\theta}]$; Zimmerman & Cressie (1992); Abt (1999). The ijth element of the FI matrix can be derived as $\text{tr}\left(\boldsymbol{C}_Z^{-1}\frac{\partial\boldsymbol{C}_Z}{\partial\theta_i}\boldsymbol{C}_Z^{-1}\frac{\partial\boldsymbol{C}_Z}{\partial\theta_j}\right)$ 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.

4.3. Houston Ozone Monitoring

A 20 parts per billion (ppb) increase in daily maximum eight-hour ozone concentration (DM8) has been associated with an increased risk of out-of-hospital cardiac arrest, with a relative risk estimate of 1.039; (95% CI [1.005, 1.073], Ensor et al., 2013). The Texas Commission on Environmental Quality (TCEQ) has monitoring stations throughout Texas recording Ozone levels, along with a variety of other environmental indicators. The TCEQ measures ozone at a network of monitoring locations and publishes DM8s in ppb for each monitoring location. The DM8s are computed as follows. First, the TCEQ creates an hourly mean in ppb for each monitoring location. Then they construct an eight hour mean 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 means for a given day is the published DM8. Days with less than 18 valid eight-hour means have no published DM8.

In August 2016 there were 44 active monitoring locations in the Houston-Galveston-Brazoria area, which we focus on. For each location $\boldsymbol{u}$ we compute the monthly mean DM8, which we denote by $Z(\boldsymbol{u})$. At one location, MRM-3 Haden Road, there are two DM8 observations of zero 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 the county. Harris County contains 33 of the 44 existing locations, though the locations outside of the county are still useful for spatial prediction within the county.

Let $Z(\boldsymbol{u})$ denote the measured mean DM8 at location $\boldsymbol{u}$ and let $Y(\boldsymbol{u})$ denote the true DM8. We assume that $Z(\boldsymbol{u})$ is a noisy measurement of $Y(\boldsymbol{u})$ with the data model $Z(\boldsymbol{u}) \sim N[Y(\boldsymbol{u}), \tau^2]$. The parameter τ^2 represents variability added due to measurement error from the instruments, small-scale effects, 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 Gaussian 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 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 defined in Section 4 in order to choose the 100 new monitoring locations in Harris County, namely $\overline{Q}_{puk}(\boldsymbol{d})$ and $Q^*_{puk}(\boldsymbol{d})$. We assume that the goal is to predict mean 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 the `gNearestPoint` function from the `rgeos` `R` package (Bivand & Rundel, 2016; R Core Team, 2016). 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 2 contains the results of applying each optimization algorithm to choosing the 100 new monitoring locations once with each algorithm. In the table, PSO refers to the standard PSO algorithm with the usual velocity update and all of the other modifications listed in Section S1.1 of the supporting materials. The CF modifier indicates that the velocity update is coordinate-free – see Section S1.1 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 they 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 respectively. All BBPSO algorithms also use $df = 1$, and each of the modifications detailed in Section S1.2 of the supporting materials. Further, the CF modifier indicates that the BBPSO algorithm uses the coordinate-free variance update and the xp modifier 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 Section S1.2. All of the PSO and BBPSO algorithms use the global neighborhood topology. In simulations not reported here, we found that the stochastic star topology resulted in worse PSO algorithms for the spatial design problem, so we limit our attention to the global topology. Further, each PSO algorithm has a swarm size of 40, which is the standard value proposed by Clerc (2012). We run each PSO algorithm for $K = 2000$ iterations, which is roughly when most of them seemed to settle down and stop improving.

Additionally, we employ a class of genetic algorithms (GAs) described in Hamada et al. (2001) to compare to PSO, with one batch, and with two possible mutation rates ($\lambda_1 = 0.01$ or $\lambda_2 = 0.1$) and two possible mutation variances ($\mu_1 = 1$ or $\mu_2 = 2$). The GAs also use a population of 40, though they are only allowed to run for 1000 iterations so that after the initialization, the GAs use the same number of objective function evaluations as each of the PSO algorithms. In order to handle the bounded search space, we use the same method for the GAs as for the PSO algorithms described in Section S1 of the supporting materials: any point the GA suggests that is outside of Harris County is moved to the nearest point on the border of the county. Each algorithm was implemented in the `R` programming language (R Core Team, 2016). In Table 2, GAs are referred to by "GA-*ab*" where *a* denotes which value of λ is used and *b* denotes which value of μ is used. As a point of comparison for both classes of algorithms, we randomly selected the 100 new monitoring locations uniformly from Harris county 10,000 times independently and computed the values of $\overline{Q}_{puk}$ and Q^*_{puk}. The means of these values are labeled as "Uniform" in Table 2.

From Table 2 we immediately see that the first set of parameter values tends to work the best in the PSO algorithms, while in the AT-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 standard PSO variants and is outperformed by several of the AT variants. The best GAs are competitive with the best PSO algorithms, though they appear to be slightly worse. The five best performing algorithms for $\overline{Q}_{puk}$ are, in order, AT2-PSO2, AT1-PSO1, GA-11, PSO1-Global, and AT2-PSO1 (bold in Table 2). Similarly, the five best performing algorithms for Q^*_{puk} are, in order, AT1-PSO1, PSO1, GA-12, PSO2, and AT2-PSO1 (bold in Table 2). In Section 3 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 surfaces with many local minima. When the objective surface is simpler, PSO and AT-PSO variants are more attractive and appear to converge faster. Similarly, 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. But in this case, with 100 new locations, AT1-PSO1 appears to be superior for both objective functions, though the difference is small.

Figures 3 and 4 contain the best designs found using $\overline{Q}_{puk}$ and Q^*_{puk} respectively. 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 while 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 obtaining a

better estimate of $\boldsymbol{\beta}$. The further 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, and putting more monitoring locations near the edges of the county results in smaller variances for the elements of $\widehat{\boldsymbol{\beta}}_{gls}$.

5. Discussion

In this paper we reviewed PSO algorithms, introduced AT-PSO and AT-BBPSO algorithms, and demonstrated that PSO, AT-PSO, and AT-BBPSO algorithms are useful for spatial design problems. In the simulation study in Section 3 we found that AT-BBPSO variants were especially attractive for optimization problems with complicated objective surfaces, potentially with many local optima. However, in Section 4.3 we found little difference between the standard PSO algorithm and the best AT algorithms. The upshot is that at least for spatial design problems similar to or easier than the one we considered here, using a standard PSO algorithm with standard parameter settings and the typical modifications we list in Section S1.1 is a good default, especially since this algorithm is already implemented in several widely available packages such as the `pso` package in `R` (Bendtsen, 2012). Furthermore, in similar spatial design problems which are much lower dimensional than the one discussed in this paper we have found that standard PSO algorithms are often the best.

However, our results in Section 3 suggest that with more complex objective functions there may be significant benefits to using AT-BBPSO. In the hardest problems, AT-BBPSO variants were significantly better than the alternatives. In our spatial design problem in Section 4.3, the best performing algorithms were AT-PSO variants, though the difference was small. This is consistent with our results in Section 3 where occasionally an AT-PSO variant was the best for one of the simpler objective functions, but usually not by much. Compared to the design problem we considered, additional objective function complexity may come from a more complex spatial model with a more complex mean or covariance function, or a more complex design criterion such as an entropy based criterion. As long as the user has a reasonable expectation that the objective surface is not too complex, whichever algorithm is easier for them to implement and use is likely the best, but for more complex problems there is much to be gained from trying several different algorithms and in particular from the AT-BBPSO variants we introduced.

The usefulness of PSO and AT variants for statistics is not limited to spatial design. Other design problems are amenable to PSO; for example, the problem of which blocks to select in the context of address canvassing, e.g. see Young et al. (2016), after determination of coverage errors. In general, any optimization problem which is difficult to solve with gradient based algorithms is a good candidate, especially when near-optimal solutions are tolerable. In the spatial design literature, exchange algorithms are commonly employed instead of PSO or GAs, though these require limiting the search space to a discrete grid; Nychka & Saltzman (1998); Wikle & Royle (1999, 2005). PSO (and GA) allow the entire search space to be used without restriction and are attractive for this reason, though in cases where there is a known discrete set of possible monitoring locations the exchange algorithm is more suited to the task.

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, doi:10.1111/1467-9469.00168.

Andrieu, C & Thoms, J (2008), 'A tutorial on adaptive MCMC,' *Statistics and Computing*, **18**(4), pp. 343–373, doi:10.1007/s11222-008-9110-y.

Bendtsen, C (2012), *pso: Particle Swarm Optimization*, R package version 1.0.3.

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

Blum, C & Li, X (2008), 'Swarm intelligence in optimization,' in Blum, C & Merkle, D (eds.), *Swarm Intelligence: Introduction and Applications*, Springer-Verlag, Berlin, pp. 43–85, doi:10.1007/978-3-540-74089-6_2.

Clerc, M (2006), 'Stagnation analysis in particle swarm optimisation or what happens when nothing happens,' 17 pages. https://hal.archives-ouvertes.fr/hal-00122031.

Clerc, M (2010), *Particle swarm optimization*, John Wiley & Sons, doi:10.1002/9780470612163.

Clerc, M (2012), 'Standard particle swarm optimisation,' 15 pages. https://hal.archives-ouvertes.fr/hal-00764996.

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, doi:10.1109/4235.985692.

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

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, doi:10.1109/CEC.2000.870279.

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, doi:10.1161/CIRCULATIONAHA.113.000027.

Gelman, A, Roberts, G & Gilks, W (1996), 'Efficient Metropolis jumping rules,' in Bernardo, J, Berger, J, Dawid, A & Smith, A (eds.), *Bayesian statistics 5*, Oxford University Press, Oxford, pp. 599–608.

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, doi:10.1198/000313001317098121.

Hsieh, HI & Lee, TS (2010), 'A modified algorithm of bare bones particle swarm optimization,' *International Journal of Computer Science Issues*, **7**, pp. 12–17.

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, doi:10.1109/SIS.2003.1202251.

Le, ND & Zidek, JV (2006), *Statistical Analysis of Environmental Space-Time Processes*, Springer-Verlag, New York, doi:10.1007/0-387-35429-8.

Mateu, J & Müller, WG (2012), *Spatio-Temporal Design: Advances in Efficient Data Acquisition*, John Wiley & Sons, Ltd, West Sussex, United Kingdom, doi:10.1002/9781118441862.

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.

Monson, CK & Seppi, KD (2005), 'Exposing origin-seeking bias in PSO,' in *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, ACM, New York, pp. 241–248, doi:10.1145/1068009.1068045.

Müller, WG (2007), *Collecting Spatial Data: Optimum Design of Experiments for Random Fields*, Springer-Verlag, Berlin, doi:10.1007/978-3-540-31175-1.

Nychka, D & Saltzman, N (1998), 'Design of air-quality monitoring networks,' in Nychka, D, Piegorsch, WW & Cox, LH (eds.), *Case Studies in Environmental Statistics*, Springer, New York, pp. 51–76, doi:10.1007/978-1-4612-2226-2_4.

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, doi:doi:10.4018/jsir.2010040103.

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, doi:10.3233/KES-2010-0211.

Wikle, CK & Royle, JA (1999), 'Space-time dynamic design of environmental monitoring networks,' *Journal of Agricultural, Biological, and Environmental Statistics*, **4**(4), pp. 489–507, doi:10.2307/1400504.

Wikle, CK & Royle, JA (2005), 'Dynamic design of ecological monitoring networks for non-Gaussian spatio-temporal data,' *Environmetrics*, **16**(5), pp. 507–522, doi:10.1002/env.718.

Young, DS, Raim, AM & Johnson, NR (2016), 'Zero-inflated modelling for characterizing coverage errors of extracts from the US Census Bureau's Master Address File,' *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, **180**(1), pp. 73–97, doi:10.1111/rssa.12183.

Zhang, W, Liu, Y & Clerc, M (2003), 'An adaptive PSO algorithm for reactive power optimization,' in *IET Conference Proceedings*, Institution of Engineering and Technology, pp. 302–307, doi:10.1049/cp:20030603.

Zimmerman, DL (2006), 'Optimal network design for spatial prediction, covariance parameter estimation, and empirical prediction,' *Environmetrics*, **17**(6), pp. 635–652, doi:10.1002/env.769.

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, doi:10.1007/BF00048668.

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 $||\cdot||$ is the Euclidean norm: $||\boldsymbol{\theta}|| = \sqrt{\sum_{i=1}^{D} \theta_i^2}$.

Figure 1. Inertia over time for the DI-PSO algorithm with $\alpha = 200$ and $\beta = 1$, and for one replication of the AT2-PSO-CF algorithm with the SS3 neighborhood for each of OFs 1 and 6.

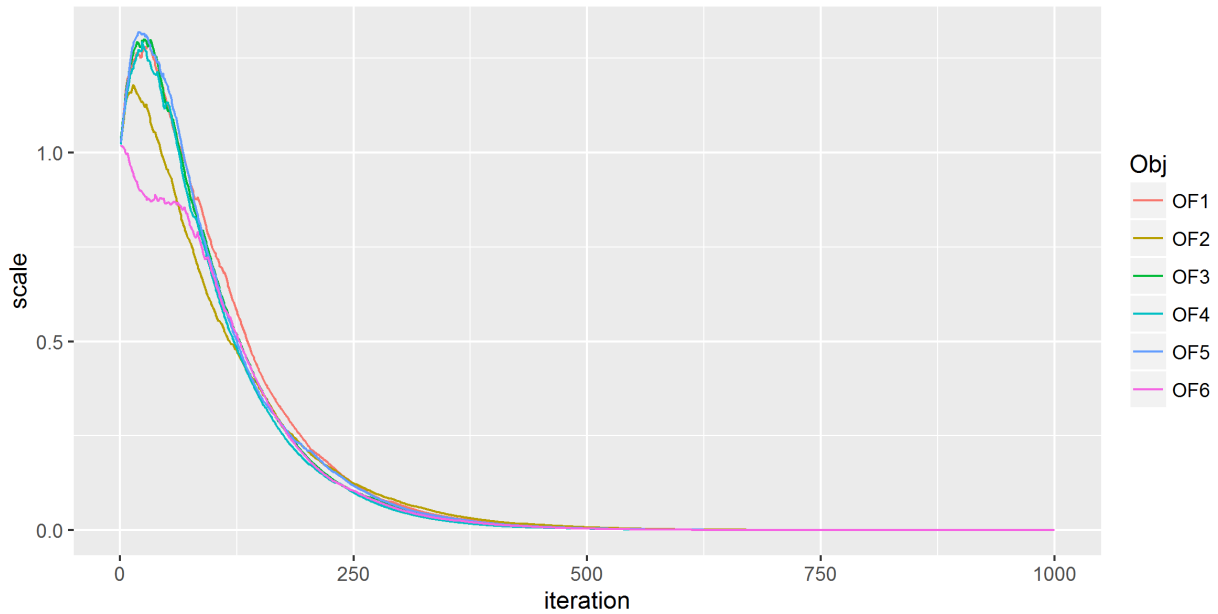

Figure 2. Scale parameter over time for one replication of the the AT2-BBPSO-CF algorithm with the SS3 neighborhood for each of OF1–OF6.

Algorithm	$\overline{Q}_{puk}$	Q^*_{puk}
Uniform	16.40	26.80
PSO1	**14.40**	**20.63**
PSO2	14.45	**21.03**
PSO1-CF	15.53	23.54
PSO2-CF	15.77	23.16
AT1-PSO1	**14.38**	**20.57**
AT1-PSO2	14.56	23.18
AT1-PSO1-CF	15.96	23.33
AT1-PSO2-CF	15.60	24.02
AT2-PSO1	**14.42**	**21.13**
AT2-PSO2	**14.32**	22.11
AT2-PSO1-CF	15.85	24.00
AT2-PSO2-CF	15.95	23.63
AT1-BBPSO	14.53	22.28
AT1-BBPSOxp	15.87	22.19
AT1-BBPSO-CF	14.65	21.33
AT1-BBPSOxp-CF	14.84	22.34
AT2-BBPSO	14.65	23.49
AT2-BBPSOxp	15.21	23.25
AT2-BBPSO-CF	14.63	21.92
AT2-BBPSOxp-CF	14.52	22.76
GA-11	**14.40**	21.19
GA-21	15.20	23.21
GA-12	14.45	**20.84**
GA-22	15.26	22.61

Table 2. Simulation results for each objective function, $\overline{Q}_{puk}$ and Q^*_{puk}. Bold values are the five best values for that objective function. AT-PSO and AT-BBPSO variants are adaptively-tuned, with AT1 and AT2 corresponding to two different parameter settings for the tuned portion of the algorithm. CF indicates that the velocity or scale parameter update is coordinate free. PSO1 and PSO2 use two different parameter settings for any non-adaptive parameters, and similarly for BBPSO and BBPSOxp. GAs have two possible values for a rate parameter and also for a variance parameter, indicated by the appended numbers. See text for further description of all of these parameter settings.

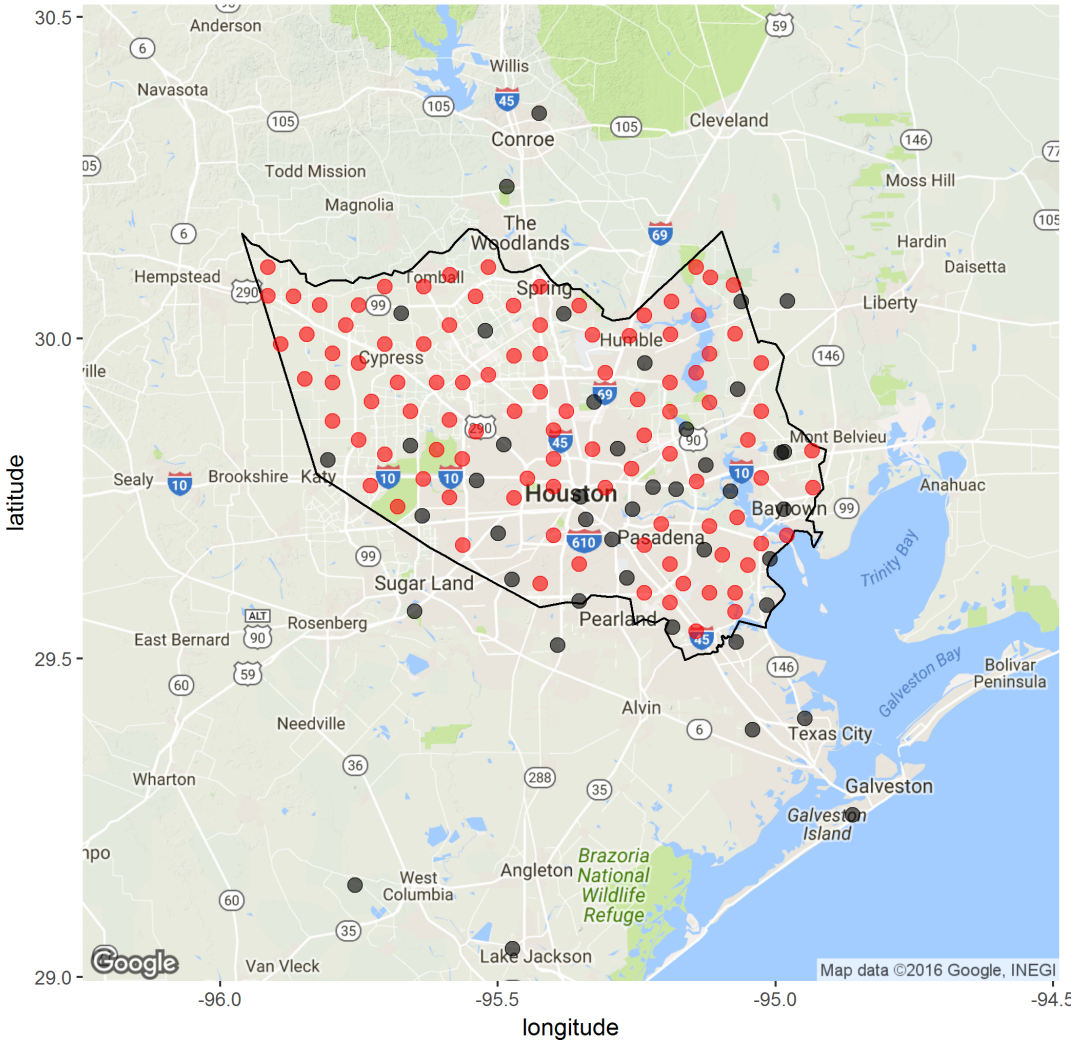

Figure 3. Best designs found according to $\overline{Q}_{puk}$. Original network points are gray, new points are red. Harris County is outlined in black. The background map is from Google Maps via `ggmap`; Kahle & Wickham (2013).

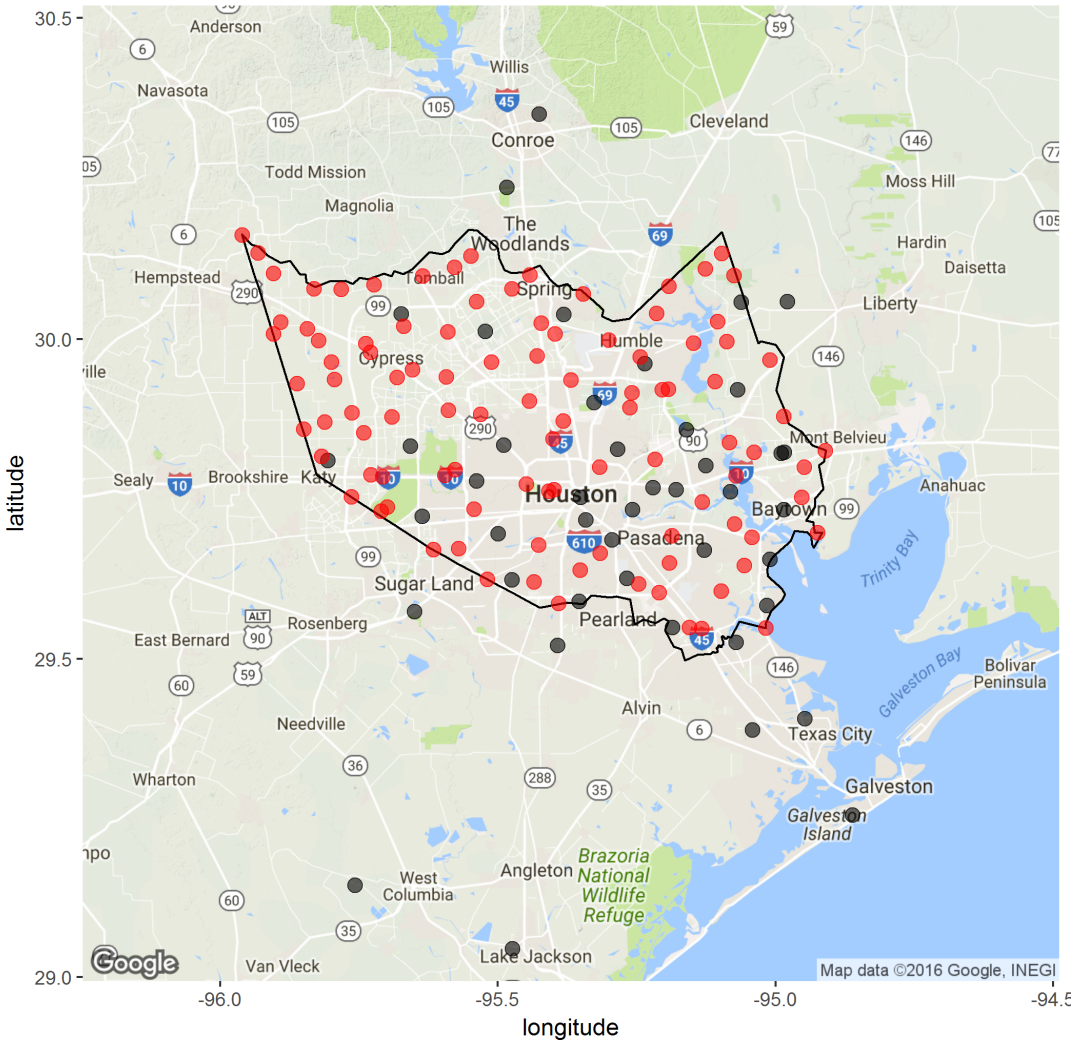

Figure 4. Best designs found according to Q^*_{puk}. Original network points are Gray, new points are red. Harris County is outlined in black. The background map is from Google Maps via ggmap; Kahle & Wickham (2013).

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

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

Received ; Accepted

S1. PSO and BBPSO details

In Section 2 we reviewed PSO and BBPSO, and introduced our adaptively tuned variants of both. Here we describe in detail several modifications to both PSO and BBPSO. Most of these are so-called standard modifications from Clerc (2012).

S1.1. PSO

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

S1.1.1. Initialization: In order to initialize the swarm, the number of particles, their initial locations, and their initial velocities have to be chosen. Clerc (2012) suggests making the swarm size a function of the dimension of the search space, but notes that this is known to be hard to do well and suggests a default swarm size of 40. We use this latter suggestion. 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 4.3 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 discussed next.

*Email:

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

S1.1.2. 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 address this problem and each has advantages and disadvantages depending on the situation. 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 (2012) 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, and in Section 4.3 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. To move the particle we use the `gNearestPoint` function from the `rgeos` `R` package (Bivand & Rundel, 2016; R Core Team, 2016).

S1.1.3. 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.

S1.1.4. Asynchronous Updates: The way we defined PSO in (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 iteration $k+1$ 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 $o_1(k+1)$, $o_2(k+1)$, etc., so that $o_j(k+1)$ is the jth particle to be updated in period $k+1$. Then 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.

S1.1.5. Coordinate Free Velocity Updates: The standard velocity update in (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., see Monson & Seppi (2005). We use the coordinate free (CF) update suggested by Clerc (2012). 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 $||\cdot||$ 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 (2012) calls SPSO 2011.

S1.1.6. When Personal Best Equals the 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 $\boldsymbol{C}_i(k) = \boldsymbol{\theta}_i(k) + \phi_2\{\boldsymbol{p}_i(k) - \boldsymbol{\theta}_i(k)\}/2$.

S1.2. BBPSO

The standard BBPSO algorithm was introduced by Kennedy (2003) and updates from t to $t+1$ via (2). We use each of the features in Section S1.1 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 (3) to create our AT-BBPSO algorithms.

S1.2.1. 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

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

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

S1.2.2. 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 $h_{ij}(k) = ||\boldsymbol{p}_i(k) - \boldsymbol{g}_i(k)||$.

S1.2.3. When Personal Best Equals The 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

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

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

S2. Tables for Comparing PSO and BBPSO algorithms

Tables S2.1–S2.6 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 as 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 for the sake of readable tables. See Section 3 for discussion of these tables.

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
AT1-PSO1	0.00	0.00	1.00	186	0.00	0.00	1.00	263	1962.50	1777.20	0.00	> 1000
AT1-PSO2	0.00	0.00	1.00	183	0.00	0.00	1.00	247	932.07	1118.20	0.00	> 1000
AT1-PSO1-CF	329.08	260.13	0.00	> 1000	294.80	166.07	0.00	> 1000	2489.30	1040.10	0.00	> 1000
AT1-PSO2-CF	342.09	293.48	0.00	> 1000	268.43	181.01	0.00	> 1000	2096.10	1131.30	0.00	> 1000
AT2-PSO1	0.00	0.00	1.00	112	0.00	0.00	1.00	154	232.14	205.78	0.00	> 1000
AT2-PSO2	0.00	0.00	1.00	117	0.00	0.00	1.00	150.5	127.58	164.99	0.00	> 1000
AT2-PSO1-CF	165.98	133.79	0.00	> 1000	196.39	130.15	0.00	> 1000	1550.60	696.54	0.00	> 1000
AT2-PSO2-CF	118.22	77.93	0.00	> 1000	160.47	118.92	0.00	> 1000	1453.20	922.27	0.00	> 1000
AT1-BBPSO	0.00	0.00	1.00	740	0.00	0.00	1.00	756	0.00	0.00	1.00	679.5
AT1-BBPSOxp	0.00	0.00	1.00	822.5	0.00	0.00	1.00	831	0.00	0.00	1.00	694
AT1-BBPSO-CF	0.00	0.00	1.00	637	0.00	0.00	1.00	642	0.00	0.00	1.00	572.5
AT1-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
AT2-BBPSO	0.00	0.00	1.00	445.5	0.00	0.00	1.00	465	0.00	0.00	1.00	451
AT2-BBPSOxp	0.00	0.00	1.00	501	0.00	0.00	1.00	511	0.00	0.00	1.00	479
AT2-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
AT2-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 S2.1. Simulation results for OF1. Each algorithm was run for 1,000 iterations over 40 replications. Mean and SD denote the mean and standard deviation of minimum objective function values found over all replications, while $\widehat{P}$ denotes the proportion of all replications that came within 0.01 of the true global minimum (equal to zero), and $\widehat{K}$ denotes the median number of iterations until the algorithm came within 0.01 of the global minimum.

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
AT1-PSO1	0.00	0.00	1.00	478.5	0.08	0.18	0.25	> 1000	7808.60	2929.70	0.00	> 1000
AT1-PSO2	0.00	0.00	1.00	481.5	0.01	0.01	0.85	924	4071.70	1647.50	0.00	> 1000
AT1-PSO1-CF	1424.90	620.02	0.00	> 1000	1021.10	509.66	0.00	> 1000	2910.80	1528.70	0.00	> 1000
AT1-PSO2-CF	1517.60	1070.00	0.00	> 1000	883.77	458.02	0.00	> 1000	2480.30	1419.50	0.00	> 1000
AT2-PSO1	91.83	356.78	0.10	> 1000	0.40	0.85	0.12	> 1000	4323.20	1925.80	0.00	> 1000
AT2-PSO2	5.98	20.86	0.52	979	0.23	1.24	0.60	965.5	2122.30	1402.00	0.00	> 1000
AT2-PSO1-CF	1358.10	766.94	0.00	> 1000	969.11	476.07	0.00	> 1000	2737.70	1082.90	0.00	> 1000
AT2-PSO2-CF	1078.80	640.14	0.00	> 1000	878.00	410.46	0.00	> 1000	2342.40	1427.20	0.00	> 1000
AT1-BBPSO	0.06	0.03	0.00	> 1000	0.01	0.01	0.62	989	0.00	0.00	0.90	916.5
AT1-BBPSOxp	0.67	0.41	0.00	> 1000	0.02	0.01	0.15	> 1000	0.01	0.00	0.85	944
AT1-BBPSO-CF	0.03	0.01	0.05	> 1000	0.00	0.00	0.97	916	0.00	0.00	1.00	819.5
AT1-BBPSOxp-CF	0.35	0.22	0.00	> 1000	0.01	0.01	0.72	972	0.00	0.00	1.00	873.5
AT2-BBPSO	0.00	0.00	1.00	852	0.00	0.00	0.92	846.5	0.00	0.00	1.00	660
AT2-BBPSOxp	0.45	0.63	0.00	> 1000	0.05	0.13	0.50	972	0.00	0.00	1.00	718
AT2-BBPSO-CF	0.00	0.00	1.00	821	0.01	0.02	0.92	825	0.00	0.00	1.00	636.5
AT2-BBPSOxp-CF	0.24	0.20	0.00	> 1000	0.04	0.08	0.52	962	0.00	0.00	1.00	682

Table S2.2. Simulation results for OF2. Each algorithm was run for 1,000 iterations over 40 replications. Mean and SD denote the mean and standard deviation of minimum objective function values found over all replications, while $\widehat{P}$ denotes the proportion of all replications that came within 0.01 of the true global minimum (equal to zero), and $\widehat{K}$ denotes the median number of iterations until the algorithm came within 0.01 of the global minimum.

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
AT1-PSO1	42.13	62.63	0.00	> 1000	27.96	32.20	0.00	> 1000			0.00	> 1000
AT1-PSO2	29.68	49.48	0.02	> 1000	31.32	39.42	0.00	> 1000			0.00	> 1000
AT1-PSO1-CF			0.00	> 1000			0.00	> 1000			0.00	> 1000
AT1-PSO2-CF			0.00	> 1000			0.00	> 1000			0.00	> 1000
AT2-PSO1	63.40	150.86	0.02	> 1000	27.31	45.22	0.00	> 1000			0.00	> 1000
AT2-PSO2	45.50	127.02	0.00	> 1000	21.57	33.17	0.00	> 1000			0.00	> 1000
AT2-PSO1-CF			0.00	> 1000			0.00	> 1000			0.00	> 1000
AT2-PSO2-CF			0.00	> 1000			0.00	> 1000			0.00	> 1000
AT1-BBPSO	286.56	733.87	0.00	> 1000	163.98	568.42	0.00	> 1000	55.96	68.75	0.00	> 1000
AT1-BBPSOxp	152.06	495.54	0.00	> 1000	53.66	40.22	0.00	> 1000	38.12	36.67	0.00	> 1000
AT1-BBPSO-CF	121.97	358.24	0.00	> 1000	199.26	573.65	0.00	> 1000	36.63	52.87	0.00	> 1000
AT1-BBPSOxp-CF	390.89	857.50	0.00	> 1000	39.49	42.56	0.00	> 1000	31.37	35.33	0.00	> 1000
AT2-BBPSO	192.88	490.38	0.00	> 1000	63.77	83.78	0.00	> 1000	84.38	132.31	0.00	> 1000
AT2-BBPSOxp	124.92	235.06	0.00	> 1000	41.88	46.31	0.00	> 1000	34.63	36.02	0.00	> 1000
AT2-BBPSO-CF	325.19	767.79	0.00	> 1000	115.76	398.59	0.00	> 1000	71.89	121.28	0.00	> 1000
AT2-BBPSOxp-CF	229.62	646.40	0.00	> 1000	45.22	55.65	0.00	> 1000	37.75	59.87	0.00	> 1000

Table S2.3. Simulation results for OF3. Each algorithm was run for 1,000 iterations over 40 replications. Mean and SD denote the mean and standard deviation of minimum objective function values found over all replications, while $\widehat{P}$ denotes the proportion of all replications that came within 0.01 of the true global minimum (equal to zero), and $\widehat{K}$ denotes the median number of iterations until the algorithm came within 0.01 of the global minimum.

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
AT1-PSO1	4.76	2.15	0.00	> 1000	2.16	1.76	0.17	> 1000	1584.60	1101.90	0.00	> 1000
AT1-PSO2	6.85	2.12	0.00	> 1000	2.43	1.49	0.05	> 1000	772.81	909.62	0.00	> 1000
AT1-PSO1-CF	963.98	737.17	0.00	> 1000	470.14	242.39	0.00	> 1000	2032.40	937.66	0.00	> 1000
AT1-PSO2-CF	721.45	403.80	0.00	> 1000	358.45	172.38	0.00	> 1000	1780.40	650.79	0.00	> 1000
AT2-PSO1	7.44	3.39	0.00	> 1000	3.50	1.66	0.00	> 1000	108.91	112.21	0.00	> 1000
AT2-PSO2	10.01	4.71	0.00	> 1000	5.02	2.70	0.00	> 1000	122.32	191.30	0.00	> 1000
AT2-PSO1-CF	495.75	240.51	0.00	> 1000	454.87	259.16	0.00	> 1000	1355.10	663.30	0.00	> 1000
AT2-PSO2-CF	425.66	293.16	0.00	> 1000	363.88	192.89	0.00	> 1000	1209.60	810.10	0.00	> 1000
AT1-BBPSO	3.22	1.39	0.05	> 1000	0.46	0.53	0.30	> 1000	1.15	1.23	0.35	> 1000
AT1-BBPSOxp	0.05	0.15	0.00	> 1000	0.03	0.01	0.00	> 1000	0.14	0.35	0.65	974.5
AT1-BBPSO-CF	3.59	1.63	0.00	> 1000	0.24	0.52	0.80	898.5	1.17	1.15	0.35	> 1000
AT1-BBPSOxp-CF	0.15	0.34	0.75	985	0.01	0.00	0.85	973.5	0.12	0.32	0.88	865
AT2-BBPSO	4.59	1.93	0.00	> 1000	0.62	0.79	0.52	633.5	1.14	1.18	0.40	> 1000
AT2-BBPSOxp	0.17	0.43	0.85	656	0.00	0.00	1.00	672	0.12	0.32	0.88	633.5
AT2-BBPSO-CF	3.54	1.85	0.00	> 1000	0.59	0.74	0.52	583.5	1.57	1.30	0.22	> 1000
AT2-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 S2.4. Simulation results for OF4. Each algorithm was run for 1,000 iterations over 40 replications. Mean and SD denote the mean and standard deviation of minimum objective function values found over all replications, while $\widehat{P}$ denotes the proportion of all replications that came within 0.01 of the true global minimum (equal to zero), and $\widehat{K}$ denotes the median number of iterations until the algorithm came within 0.01 of the global minimum.

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
AT1-PSO1	0.02	0.02	0.42	> 1000	0.01	0.01	0.78	253	1.37	0.28	0.00	> 1000
AT1-PSO2	0.02	0.02	0.32	> 1000	0.01	0.01	0.82	239.5	1.44	0.56	0.00	> 1000
AT1-PSO1-CF	1.09	0.22	0.00	> 1000	1.04	0.10	0.00	> 1000	1.64	0.26	0.00	> 1000
AT1-PSO2-CF	1.03	0.21	0.00	> 1000	1.04	0.11	0.00	> 1000	1.61	0.26	0.00	> 1000
AT2-PSO1	0.03	0.03	0.38	> 1000	0.01	0.01	0.52	167	0.93	0.28	0.00	> 1000
AT2-PSO2	0.03	0.03	0.35	> 1000	0.01	0.01	0.65	132.5	0.85	0.27	0.00	> 1000
AT2-PSO1-CF	0.97	0.25	0.00	> 1000	1.02	0.10	0.00	> 1000	1.42	0.21	0.00	> 1000
AT2-PSO2-CF	0.91	0.19	0.00	> 1000	0.97	0.10	0.00	> 1000	1.38	0.21	0.00	> 1000
AT1-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
AT1-BBPSOxp	0.01	0.01	0.72	631	0.00	0.00	1.00	623.5	0.00	0.00	0.95	566.5
AT1-BBPSO-CF	0.02	0.02	0.35	> 1000	0.00	0.01	0.85	475	0.01	0.01	0.68	472
AT1-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
AT2-BBPSO	0.02	0.02	0.40	> 1000	0.00	0.01	0.92	366.5	0.01	0.01	0.72	406.5
AT2-BBPSOxp	0.01	0.01	0.78	387	0.00	0.00	0.97	400.5	0.00	0.01	0.90	380.5
AT2-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
AT2-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 S2.5. Simulation results for OF5. Each algorithm was run for 1,000 iterations over 40 replications. Mean and SD denote the mean and standard deviation of minimum objective function values found over all replications, while $\widehat{P}$ denotes the proportion of all replications that came within 0.01 of the true global minimum (equal to zero), and $\widehat{K}$ denotes the median number of iterations until the algorithm came within 0.01 of the global minimum.

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
AT1-PSO1	20.00	0.00	0.00	> 1000	20.01	0.01	0.00	> 1000	20.00	0.00	0.00	> 1000
AT1-PSO2	20.02	0.04	0.00	> 1000	20.08	0.06	0.00	> 1000	20.00	0.00	0.00	> 1000
AT1-PSO1-CF	20.06	0.06	0.00	> 1000	20.14	0.07	0.00	> 1000	20.00	0.00	0.00	> 1000
AT1-PSO2-CF	20.23	0.09	0.00	> 1000	20.16	0.09	0.00	> 1000	19.68	2.06	0.00	> 1000
AT2-PSO1	20.01	0.02	0.00	> 1000	20.07	0.05	0.00	> 1000	20.22	0.09	0.00	> 1000
AT2-PSO2	20.12	0.10	0.00	> 1000	20.27	0.08	0.00	> 1000	20.33	0.11	0.00	> 1000
AT2-PSO1-CF	20.33	0.10	0.00	> 1000	20.33	0.10	0.00	> 1000	20.32	0.11	0.00	> 1000
AT2-PSO2-CF	20.38	0.12	0.00	> 1000	20.32	0.13	0.00	> 1000	20.32	0.11	0.00	> 1000
AT1-BBPSO	17.22	8.02	0.00	> 1000	5.23	9.13	0.00	> 1000	18.71	6.28	0.00	> 1000
AT1-BBPSOxp	18.34	6.18	0.00	> 1000	17.58	6.79	0.00	> 1000	20.42	0.21	0.00	> 1000
AT1-BBPSO-CF	15.63	9.14	0.07	> 1000	0.53	3.30	0.42	> 1000	13.68	9.62	0.22	> 1000
AT1-BBPSOxp-CF	14.75	9.19	0.00	> 1000	10.35	9.46	0.00	> 1000	20.40	0.33	0.00	> 1000
AT2-BBPSO	17.76	7.56	0.15	> 1000	5.95	8.70	0.65	684.5	19.04	5.54	0.07	> 1000
AT2-BBPSOxp	18.22	6.18	0.10	> 1000	17.65	5.79	0.05	> 1000	20.17	1.84	0.00	> 1000
AT2-BBPSO-CF	18.33	6.76	0.07	> 1000	2.06	6.25	0.90	628	14.47	9.01	0.22	> 1000
AT2-BBPSOxp-CF	14.49	9.15	0.28	> 1000	10.03	8.80	0.32	> 1000	18.77	4.41	0.00	> 1000

Table S2.6. Simulation results for OF6. Each algorithm was run for 1,000 iterations over 40 replications. Mean and SD denote the mean and standard deviation of minimum objective function values found over all replications, while $\widehat{P}$ denotes the proportion of all replications that came within 0.01 of the true global minimum (equal to zero), and $\widehat{K}$ denotes the median number of iterations until the algorithm came within 0.01 of the global minimum.