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 optimization problems. We propose using PSO in order to solve spatial design problems, e.g. choosing new locations to add to an existing monitoring network. Additionally 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.

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 many, such as design problems, are more difficult. In spatial design problems, the objective function is some design criterion typically related to the variability of predictions, which is a complex, multimodal function of the design points, and this function often lives in a complex domain. Graident-based methods are typically inefficient 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 also 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 tend to be superior to many PSO alternatives as well as to a class of genetic algorithms.

The remainder of the paper is as follows. Section 2 introduces various PSO, AT-PSO, and AT-BBPSO algorithms, and Section 3 briefly discusses the results of a simulation study comparing them. Section 4 introduces a generic spatial design problem and the Houson 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 (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 $Q(\boldsymbol{p}_i(k)) \leq Q(\boldsymbol{\theta}_i(l))$ for any $k \geq l$. The group best location is defined with respect to some neighborhood $\mathcal{N}_i$ of 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 limit how many other particles each particle can communicate with. 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 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 number of members 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 specificied parameter that controls the speed of adaptation. We use a t kernel 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 the connection to adaptive random walk Metropolis algorithms. 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, though anything within an order of maginitude yields similar results and there do no appear to be large gains to optimizing 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 standard BBPSO setting $\sigma^2 = 1$ yields the best algorithms. Hsieh & Lee (2010) propose a variant that keeps deterministically sets $\sigma^2 < 1$ for an initial set of iterations before setting $\sigma^2 = 1$ and even 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 AT-BBPSO yields the best 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 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 magnitude of the velocities evolve over time. We can also use the same mechanism 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. Tuppadung & Kurutach (2011) 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.

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 is in their algorithm each particle adapts differently while in AT-PSO each particle adapts in the same way. 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 $D = 20$ so the domain of each function is $\Re^{20}$. Each function was constrained 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 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 consider BBPSO algorithms because they do much worse than any of the algorithms we consider. 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 $R^* = 0.5$ (AT1 or AT2 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. 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 features of these 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 progession 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 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 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 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 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 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, 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 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.

AT-PSO and AT-BBPSO clearly 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 bastly 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 $\text{cov}[Z(\boldsymbol{u}), Z(\boldsymbol{v})] = C(\boldsymbol{u}, \boldsymbol{v}) + \sigma_\varepsilon^2 1(\boldsymbol{u} = \boldsymbol{v})$, 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

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

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 $\overline{Q}_{uk}(\boldsymbol{d}) = \frac{1}{N_t}\sum_i^{N_t}\sigma_{uk}^2(\boldsymbol{t}_i; \boldsymbol{d})$ while the maximum universal kriging variance is given by $Q_{uk}^*(\boldsymbol{d}) = \max_{i=1,2\ldots,N_t}\sigma_{uk}^2(\boldsymbol{t}_i; \boldsymbol{d})$. 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 $\overline{Q}_{uk}(\boldsymbol{d}) = \frac{1}{|\mathcal{D}|}\int_{\mathcal{D}}\sigma_{uk}^2(\boldsymbol{u}; \boldsymbol{d})d\boldsymbol{u}$ and $Q_{uk}^*(\boldsymbol{d}) = \max_{\boldsymbol{u}\in\mathcal{D}}\sigma_{uk}^2(\boldsymbol{u}; \boldsymbol{d})$ 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 $\sigma_{uk}^2(\boldsymbol{u})$. 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

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

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 $\overline{Q}_{puk}(\boldsymbol{d}) = \frac{1}{N_t}\sum_i^{N_t}\sigma_{puk}^2(\boldsymbol{t}_i; \boldsymbol{d})$ and $Q_{puk}^*(\boldsymbol{d}) = \max_{i=1,2\ldots,N_t}\sigma_{puk}^2(\boldsymbol{t}_i; \boldsymbol{d})$, 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 cardiact arrest, with a relative risk estimate of 1.039; Ensor et al. (2013), 95% CI (1.005, 1.073). The Texas Commission on Environmental Quality (TCEQ) has monitoring stations throughout Texas recording a variety of environmental indicators, including Ozone. 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 an eight hour mean 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 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 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 and small-scale effects. 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 in order to choose the 100 new monitoring locations in Harris County: $\overline{Q}_{puk}(\boldsymbol{d})$ and $Q^*_{puk}(\boldsymbol{d})$, both defined in Section 4. We assume that the goal is the 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 modications 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 ourself to the global topology. Further, each PSO algorithm has a swarm size of 40, which is the standard value proposed by Clerc (2011). 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 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 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 (bolded in Table 2). Similarly, the five best performing algorithms for Q^*_{puk} are, in order, AT1-PSO1, PSO1, GA-12, PSO2, and AT2-PSO1 (bolded 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.

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. 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 because 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 $\widehat{\boldsymbol{\beta}}_{gls}$.

5. Discussion

In this paper we reviewed PSO algorithms, introduced AT-PSO and AT-BBPSO algorithms, and demonstrated that AT-PSO and PSO 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 Sectoin S1.1 of the supporting materials 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). Similarly, standard GAs are a good default option, and packages for implementing them are also widely available. 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 such as the problem of which blocks to select in the context of address canvassing, e.g. in 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.

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

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.

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.

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.

Le, ND & Zidek, JV (2006), *Statistical Analysis of Environmental Space-Time Processes*, Springer Science & Business Media.

Mateu, J & Müller, WG (2012), *Spatio-Temporal Design: Advances in Efficient Data Acquisition*, John Wiley & Sons.

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, pp. 241–248.

Müller, WG (2007), *Collecting Spatial Data: Optimum Design of Experiments for Random Fields*, Springer Science & Business Media.

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.

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

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.

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)*.

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.

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: $||\boldsymbol{\theta}|| = \sqrt{\sum_{i=1}^{D} \theta_i^2}$.

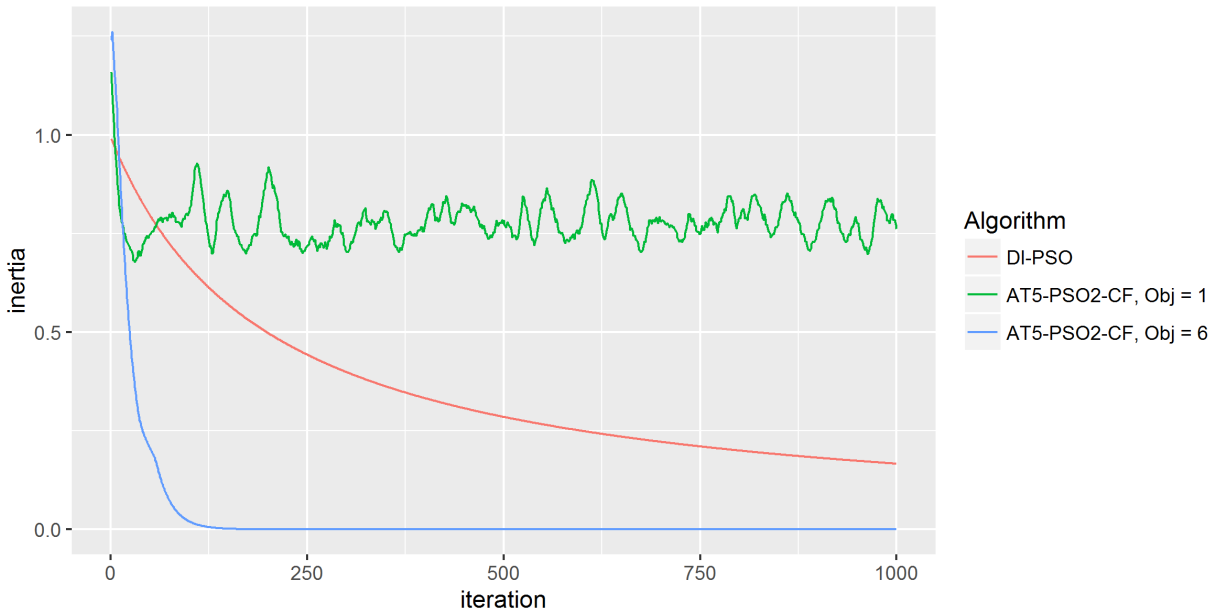

Figure 1. Inertia over time for the DI-PSO algorithm with $\alpha = 200$ and $\beta = 1$, and for one replication of the AT-PSO-0.5 algorithm with the SS3 neighborhood for each of OFs 1 and 6.

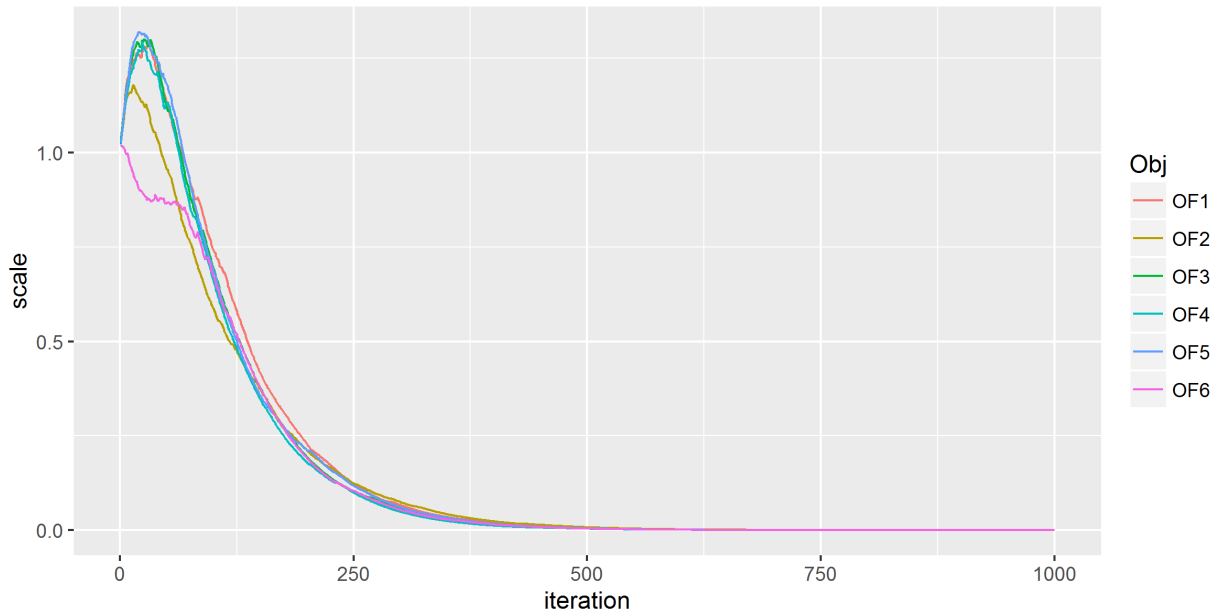

Figure 2. Scale parameter over time for one replication of the the AT2-BBPSO-CF algorithm with the SS3 neighborhood for each of OF1–OF6.

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}. Bolded 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.

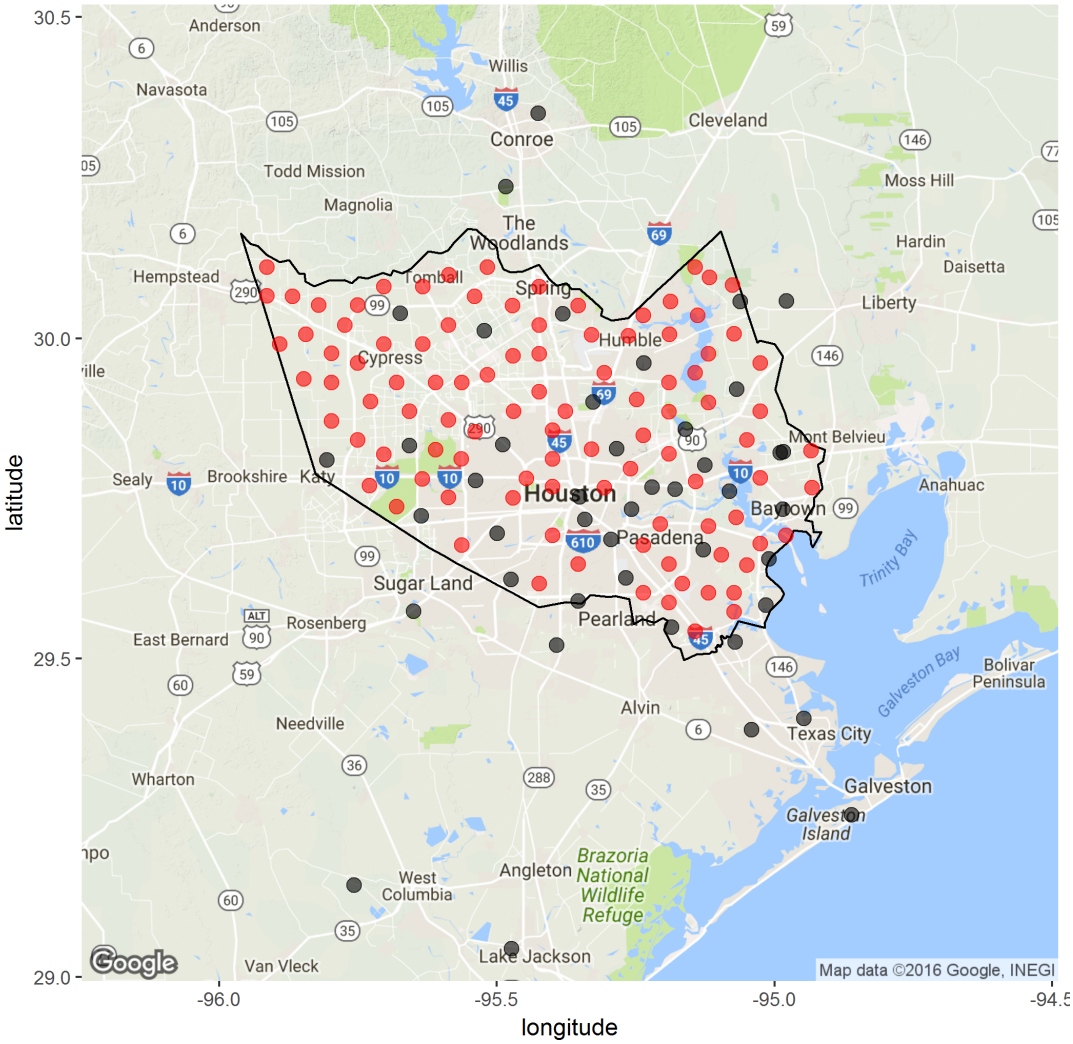

Figure 3. Best designs found according to $\overline{Q}_{puk}$. Original network points are grey, new points are red. Harris County is outlined in black. The background map is from Google Maps via `ggmap`; Kahle & Wickham (2013).

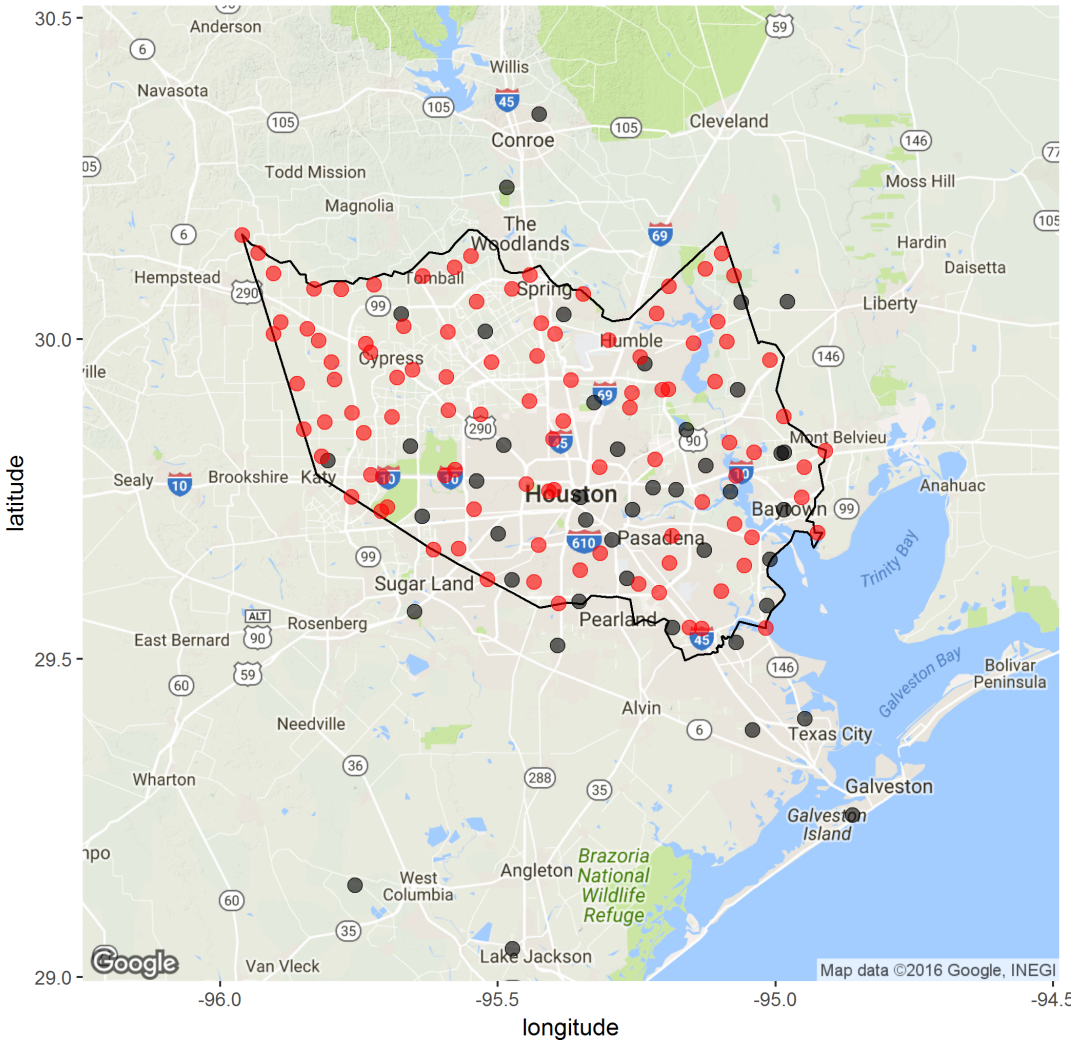

Figure 4. Best designs found according to Q^*_{puk}. Original network points are grey, new points are red. Harris County is outlined in black. The background map is from Google Maps via ggmap; Kahle & Wickham (2013).