Framework for Particle Swarm Optimization with Surrogate Functions

Technical Report TUD-CS-2009-0139

August 26, 2009

Matthew D. Parno[*], **Kathleen R. Fowler**[*], **and Thomas Hemker**[†] [‡]
[*] Department of Mathematics, Clarkson University, Potsdam, NY 13699-5815, United States (parnomd@clarkson.edu,kfowler@clarkson.edu)
[†] Simulation, Systems Optimization and Robotics, Department of Computer Science, Technische Universität Darmstadt, Hochschulstraße 10, 64289 Darmstadt, Germany (hemker@sim.tu-darmstadt.de)
[‡] Corresponding author

TECHNISCHE UNIVERSITÄT DARMSTADT

Abstract

Particle swarm optimization (PSO) is a population-based, heuristic minimization technique that is based on social behavior. The method has been shown to perform well on a variety of problems including those with nonconvex, nonsmooth objective functions with multiple local minima. However, the method can be computationally expensive in that a large number of function calls is required to advance the swarm at each optimization iteration. This is a significant drawback when function evaluations depend on output from an off-the-shelf simulation program, which is often the case in engineering applications. To this end, we propose an algorithm which incorporates surrogate functions, which serve as a stand-in for the expensive objective function, within the PSO framework. We present numerical results to show that this hybrid approach can improve algorithmic efficiency.

1 Introduction

We propose an algorithmic technique to improve the computational efficiency of a general particle swarm optimization (PSO) algorithm [1]. The motivation for this work is that population-based, heuristic methods are often applied in engineering disciplines on simulation-based design problems. The PSO has been successfully applied to a variety of such problems including [2, 3, 4, 5]. The challenges in these types of problems are inherent in the objective function and constraints, which rely on output from a black-box simulator. A wide range of applicable derivative-free methods exist for these types of problems. Population-based heuristics are often used because on initial stages the elements are widely spread out in the design space to identify promising regions. Even if prior knowledge can be seeded into initial populations, they do not rely on initial guesses for starting points which can bias many derivative-free single search approaches. However, a major flaw is that a large number of function calls is usually required.

Surrogate optimization in the context considered in this work means to solve surrogate problems generated with "stand-in" functions by using true objective function values to build an approximation to the original optimization problem. Thus, surrogate function approaches can help here to reduce the number of expensive black-box calls. The key feature of surrogate optimization is that the approach avoids running directly in a loop with the real system that generates the original optimization problem, but calls a surrogate function which has a lower computational cost to determine new promising points to test with the real system. One well known method to construct such surrogate functions is by using Design and Analysis of Computer Experiements (DACE), as described by Sacks [6]. Moreover, there are various other approximation techniques; many of which have been applied for optimization purposes, a short overview is given in e.g., [7, 8]. Surrogate optimization offers flexibility in both the choice of approximation method and choice of optimizer. The optimization method can be chosen appropriate to the properties of the generated surrogate functions so that almost no restrictions are given to use a specific method on the arising surrogate problem. For this work, we incorporate a surrogate function to serve as an oracle in the search phase of the PSO algorithm.

We provide background on the PSO algorithm in Section 2 and on functional approximations by DACE in Section 3, as well as details on the implementation in Section 4. In Section 5 we provide promising numerical results on a set of standard optimization test problems and a groundwater management application proposed in [9] and studied in [10, 11, 12, 13] as a test problem for derivative-free optimization methods. We end with some concluding remarks and plans for future work.

2 Particle Swarm Optimization

We pose our optimization problem as

$$\min_{\boldsymbol{x}\in\Omega} f(\boldsymbol{x}) \tag{1}$$

where the objective function f is defined on some feasible set Ω. The PSO algorithm was first proposed by Kennedy and Eberhart in 1995 [1] and is based on social communication within a group. A general PSO algorithm involves a set of k particles (design points), used to search the design space for a function's global minimum. Each particle moves through the space by updating its position with a velocity that accounts for the best position that particle has found and the best position found by any particle. Mathematically, for $i = 1, ..., k$, we have

$$v_i(t+1) = \omega v_i(t) + c_1 r_1(\boldsymbol{p}_i - \boldsymbol{x}_i) + c_2 r_2(\boldsymbol{g} - \boldsymbol{x}_k) \tag{2}$$

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + v_i(t+1) \tag{3}$$

where t is discrete iteration counter, $v(t)$ is the current velocity, $v(t+1)$ is the updated velocity, ω is an inertia weighting term, c_1, c_2 are real constants, r_1, r_2 are random numbers in (0,1], $\boldsymbol{x}_k$ is the current position of the i^{th} particle, $\boldsymbol{p}_k$ is the best position found by the i^{th} particle, and $\boldsymbol{g}$ is the best position found by any particle [1]. The random

numbers r_1, r_2 in Eqs. (2) and (3) make this implementation of PSO stochastic. Deterministic implementations exist when $r_1 = r_2 = 1$ [14].

Convergence and efficiency are of great concern in the application of PSO thus much analysis of the algorithm has been done. See [14, 15, 16], [17] for example. Much of the focus of previous work, specifically [14, 15], has been on altering Eq. (2) to speed up convergence of the PSO algorithm. One alteration proposed by Clerc and Kennedy in [14] removes the inertia weighting term and uses a constriction factor instead. The velocity equation then becomes

$$v_i(t+1) = \chi \left[v_i(t) + c_1 r_1 (\boldsymbol{p}_i - \boldsymbol{x}_i) + c_2 r_2 (\boldsymbol{g} - \boldsymbol{x}_i) \right] \tag{4}$$

where χ is the constriction factor given by

$$\chi := \frac{2\kappa}{\varphi - 2 + \sqrt{\varphi^2 - 4\varphi}} \tag{5}$$

and κ governs rate of convergence $\kappa \in (0, 2)$ and $\varphi = c_1 + c_2$.

For critically damped convergence, $\varphi = 4$ and for overdamped convergence $\varphi > 4$ [14]. For this work, we use $\varphi = 4.1$.

We further speed up the convergence of the PSO by making use of a surrogate function within the algorithmic framework. The surrogate function is used in addition to the objective function to exploit the use of intermediate data points found by the particles and better predict the objective function behavior. The minimum of the surrogate at each iteration is used to guide the choice of $\boldsymbol{g}$ for use in Eq. (4). This process is described in more detail in subsequent sections.

3 Surrogate functions by DACE

The optimization method we apply consists of a statistical approximation method that calculates a smooth surrogate function of the original objective function during the first iteration, and afterwards every l iterations.

As a standard approach designed for deterministic black-box optimization problems, design and analysis of computer experiments (DACE) described by Sacks et al. [6] is widely used, which has its origins in the geostatistical Kriging method [18]. A DACE model $\hat{f}$ is, in its general form, a two component model,

$$\hat{f}(\boldsymbol{x}) = \sum_{j=1}^{m} \beta_j \mu_j(\boldsymbol{x}) + Z(\boldsymbol{x}).$$

The first and more global part approximates the global trend of the unknown function f by a linear combination of a parameter vector $\beta \in \mathbb{R}^m$ and m given functions μ_j. Different approaches can be considered for this first part [19]. Universal Kriging is predicting the mean of the underlying function as a realization of a stochastic process by a linear combination of m known basis functions on the whole domain, as well as detrended Kriging applies a classical linear regression model. In our application we follow the idea of ordinary Kriging where a constant mean is assumed on the whole domain with $m = 1$ and one real valued constant μ.

The second part, $Z(\boldsymbol{x})$, guarantees that the DACE model $\hat{f}$ fits to f for a number of sets of points $\boldsymbol{x}_i$, $i = 1, ..., n$, out of the variable space,

$$f(\boldsymbol{x}_i) = \hat{f}(\boldsymbol{x}_i),\ i = 1, ..., n.$$

This part is assumed to be a realization of a stationary Gaussian random function with a mean of zero, $E[Z(x)] = 0$, and a covariance

$$Cov[Z(\boldsymbol{x}_i), Z(\boldsymbol{x}_j)] = \sigma^2 R(\boldsymbol{x}_i, \boldsymbol{x}_j),$$

with $i \neq j$, and $i, j \in \{1, ..., n\}$. The smoothness and properties like differentiability are controlled by the chosen correlation function R, in our case just the product of one dimensional exponential correlation functions R_j,

$$R(\boldsymbol{x}_1, \boldsymbol{x}_2) = \prod_{j=1}^{dim(x)} R_j(|x_{1,j} - x_{2,j}|),$$

as it is proposed by Sacks et al. in [6].

By maximum likelihood estimation the process variance σ^2, the regression parameter β, and the correlation parameter of R are estimated most consistent to the present objective function values to which $\hat{f}$ has to fit. For a more detailed description on the theory behind these kind of models we refer the reader to [20, 21, 22].

Algorithm 1 sPSO algorithm for NLP problems

Require: function to be minimized $f(\boldsymbol{x})$, maximum number of iterations t_{max}, guess at minimum fitness f_{min}, number of particles k, box constraints $(\boldsymbol{x}_{lb}, \boldsymbol{x}_{ub})$, maximum velocity v_{max}, and the number of iterations between surrogate function refinement l.

Ensure: During all iterations all pairs $(\boldsymbol{x}, f(\boldsymbol{x}))$ are saved for use with DACE, finally point $\boldsymbol{x}^*$ with the lowest obtained objective function value is provided.

$t = 0$
for $(i = 1 : k)$ **do**
  if (t=0) **then**
    $v_i(t+1) = rand(-v_{max}, v_{max})$,
    $\boldsymbol{x}_i = rand(\boldsymbol{x}_{lb}, \boldsymbol{x}_{ub})$,
    $\boldsymbol{p}_i = \boldsymbol{x}_i$
  else
    $v_i(t+1) = \chi[v_i(t) + c_1 R_1(\boldsymbol{p}_i - \boldsymbol{x}_i) + c_2 R_2(\boldsymbol{x}^* - \boldsymbol{x}_i)]$,
    $\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + v_i(t+1)$,
    $\boldsymbol{p}_i(t+1) = \min(\boldsymbol{p}_i(t), \boldsymbol{x}_i(t))$
  end if
  $y_i = f(\boldsymbol{x}_i)$
end for
$\boldsymbol{x}^* =$ argmin $\{f(\boldsymbol{x}) | \boldsymbol{x} \in \{\boldsymbol{x}_j\}_{j=1,\ldots,k}\}$
while $(t \leq t_{max}) \vee (f(\boldsymbol{x}^*) > f_{min})$ **do**
  if $(t = 0) \vee (t \bmod l = 0)$ **then**
    Build surrogate $\hat{f}$ with DACE
    Find $\boldsymbol{g} =$ argmin $\{\hat{f}\boldsymbol{x}) | \boldsymbol{x}_{lb} \leq \boldsymbol{x} \leq \boldsymbol{x}_{ub}\}$ by using standard PSO on $\hat{f}$
    if $(f(\hat{\boldsymbol{x}}^*) < f(\boldsymbol{g}))$ **then**
      $\boldsymbol{x}^* = \hat{\boldsymbol{x}}^*$
    else
      $\boldsymbol{x}^* = \boldsymbol{g}$
    end if
  end if
  $t = t + 1$
end while
return $\boldsymbol{x}^*$

4 Algorithmic Framework

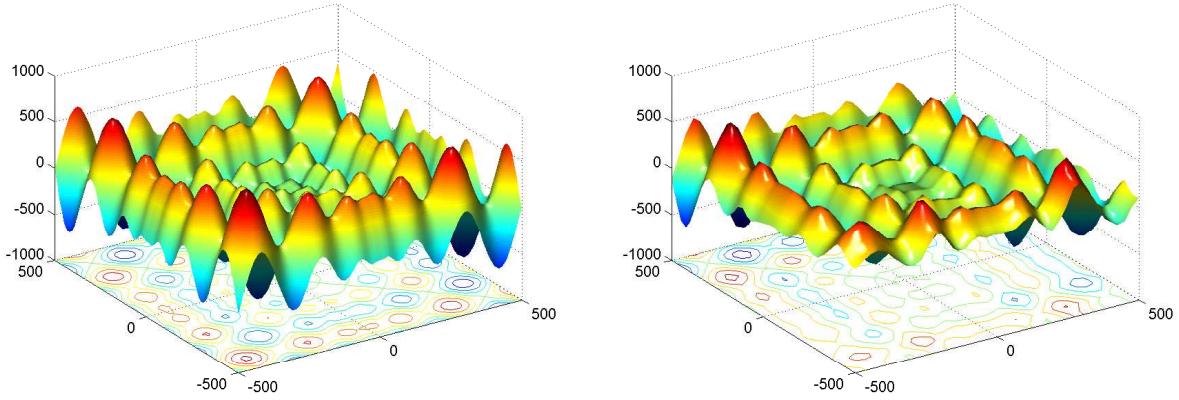
In the context of simulation-based optimization, it is desirable for an algorithm to be as frugal as possible in the number of objective function evaluations performed. For the PSO, the swarm is most often between 20 and 30 particles. After only a few iterations there will already have been a large number of function calls, which implies a reasonable surrogate function should be possible.

4.1 Surrogate Implementation in PSO Algorithm

In this implementation, a surrogate function $\hat{f}(\boldsymbol{x})$ of the objective function, $f(\boldsymbol{x})$, is first created after the initial function values for the swarm are computed. Similarities exist to earlier approaches as described in [23]. However, here the surrogate function is treated as a black box and thus better approximations can easily be made in terms of specific fitness functions. For example, if simplified physics models are available.

The surrogate function is then updated every l iterations during the loop, where l is a user specified parameter that will depend on the computational expense for each evaluation of the problem; a larger value of l may be useful on less difficult problems where the added computational effort of maintaining the surrogate outweighs the benefit of having the surrogate. Let $\hat{\boldsymbol{x}}^*$ be the minimum position found on the surrogate function (that is $\hat{f}(\hat{\boldsymbol{x}}^*)$ is the minimum value when optimizing the surrogate). To incorporate the surrogate into the PSO algorithm, we replace $\boldsymbol{g}$ in Eq. (4) with $\boldsymbol{x}^*$. Here $\boldsymbol{x}^*$ is defined by

$$\boldsymbol{x}^* = \begin{cases} \hat{\boldsymbol{x}}^*, & \text{if} \quad f(\hat{\boldsymbol{x}}^*) < f(\boldsymbol{g}) \\ \boldsymbol{g}, & \text{else} \end{cases} \tag{6}$$



(a) Original Schwefel Function (b) Surrogate Model

Figure 1: Surrogate Model of the Schwefel Function after 10 iterations of sPSO

The new equation for updating velocity becomes

$$v_k(t+1) = \chi \left[v_k(t) + c_1 r_1 (\boldsymbol{p}_k - \boldsymbol{x}_k) + c_2 r_2 (\boldsymbol{x}^* - \boldsymbol{x}_k) \right]. \tag{7}$$

We outline the algorithm for the PSO with the surrogate oracle, referred to as sPSO, in Algorithm 1.

4.2 Social Interaction within sPSO

In a basic PSO, the best position found by any of the particles is shared among all the particles. This is a very limited exchange of information. Real societal interactions generally involve a much larger combined knowledge. For example, if a group of people is attempting to find the least expensive gas station in a city, although they would ask one another the best gas prices found by each individual, a more successful group may also go to a website that keeps track of all the gas prices over time. The surrogate model works in a similar way to the gas price website, it provides a larger database of information where more accurate predictions of the global minimum can be made.

5 Numerical Results

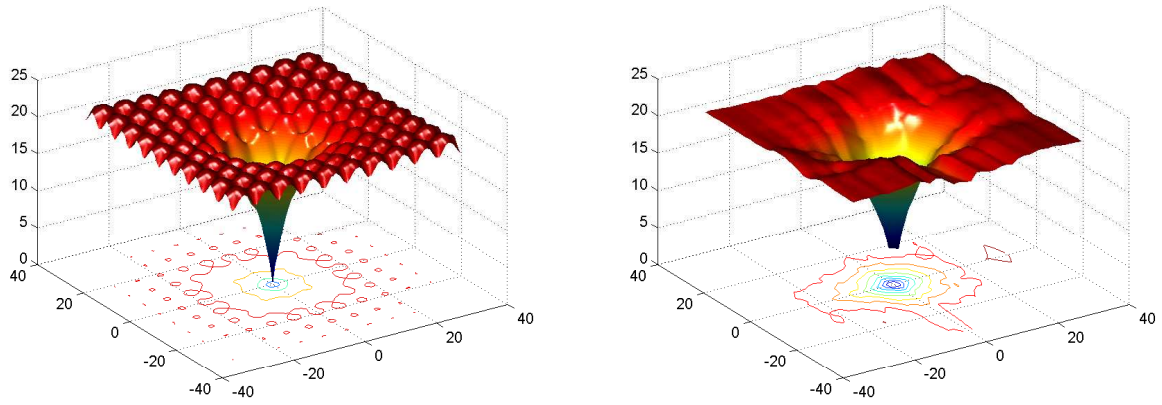
For this work, the sPSO algorithm is implemented in MATLAB, although the algorithm is platform independent. To build the surrogate function we use the Kriging capabilities of the DACE toolbox [21, 22]. It should be noted, however, that any technique can be used to create a surrogate function in the sPSO algorithm. For example, if prior knowledge about the underlying fitness function is available this should be used for the initial setting of the functional approximation methods, as any additional information can improve the quality of used surrogates functions. We begin our analysis of sPSO performance by comparing sPSO performance to the performance of an identical PSO algorithm, omitting the surrogate function.

5.1 Standard Test Problems

Our initial test suite includes the Ackley, Griewank, Michalewicz, Rastrigin, and Schwefel functions [24,25,26,27,28,29]. These test problems each contain two decision variables to ease surrogate visualization; however, the groundwater problem considered on Section 5.2 is a six variable problem. This initial set of problems are challenging for the sPSO algorithm in a variety of ways ranging from the presence of high frequency noise in the Griewank function to deep valleys and plateaus in the Michalewicz function. The functions are shown in Figures 2-1.

Optimization was performed on each function 30 times with both the sPSO and PSO algorithms. Repeating the minimization is necessary because of the random coefficients that both methods use to update each particle's position. For algorithmic parameters, we use a swarm size of 30 particles, $c_1 = 2.8$, $c_2 = 1.3$, $K = 0.7$, $\varphi = 4.1$, and $n = 1$ (meaning we update the surrogate at each iteration). For our first simple test problem set, global minima are known. Thus, the stopping criteria is defined as the point when the algorithm reached a minimum within ϵ of the known global minimum. The value of ϵ will depend on the scale of the problem and desired accuracy of the result; however, in our test problems $\epsilon < .001$ in all cases.

A summary of the results is in Table 1. For all of the test functions except the Griewank function 3, the global minima was reached with every optimization run. Although in a few instances the PSO and sPSO algorithm did not converge to



(a) Original Ackley Function (b) Surrogate Model

Figure 2: Surrogate Model of the Ackley Function after 10 iterations of sPSO

Table 1: Summary of Results for comparison of sPSO and PSO algorithms

	Ackley		Griewank		Michalewicz		Rastrigin		Schwefel	
	PSO	sPSO	PSO	sPSO	PSO	sPSO	PSO	sPSO	PSO	sPSO
$\bar{N}$	44.7	26.73	149.6	148.7	16.23	10.23	33.93	20.63	60.5	28.4
s	4.55	1.95	94.9	89.4	3.34	1.61	7.88	3.91	69.8	34
$\bar{N}_s$	0	7.03	0	3.07	0	7.43	0	11.6	0	9.4

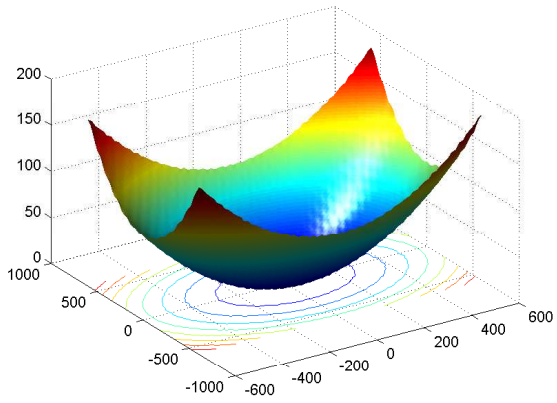
the global minima of the Griewank function within 200 iterations, when run longer (up to 400 iterations), the algorithms always converged to the global minima. In Table 1, $\bar{N}$ is the total number of iterations in an optimization run, $\bar{N}_s$ is the number of times the surrogate minimum was used instead of the best particle, s is the sample standard deviation of $\bar{N}$.

The results imply that the use of a surrogate function can improve the efficiency of the algorithm by enhancing the search phase. Statistically, the two optimization techniques perform the same on the Griewank function, implying that the use of a surrogate function in the PSO algorithm is not beneficial on functions containing high frequency noise. Conceptually, this makes sense; the higher frequency components of a function cannot be modeled using the relatively limited sampling ability of the swarm.

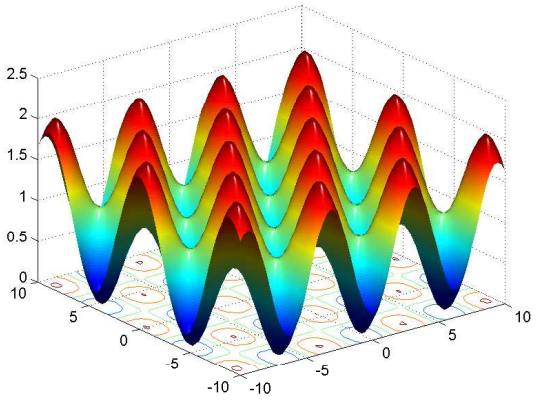
The surrogate functions generated by the sPSO can be seen in Figures 2-1. Notice the surrogate functions are more accurate near global minimums of the surrogate functions. As the sPSO algorithm progresses, the swarm moves closer to the surrogate minimum. When the surrogate function is updated, the larger number of known values causes the surrogate function to be more accurate near the surrogate minimum. In the above test functions this does not effect sPSO performance; however, the uneven refinement could trap the swarm at local minima far from the global minimum. If the surrogate function does not capture the area surrounding the global minimum of the function, the swarm has a lower chance of finding the global minimum. To help avoid this, Latin Hypercube Sampling (LHS) is used to generate the initial positions of the particles. Users can also decrease the risk of not capturing the global minimum by increasing the swarm size, thus creating a finer initial surrogate function.

5.2 Groundwater Management Application

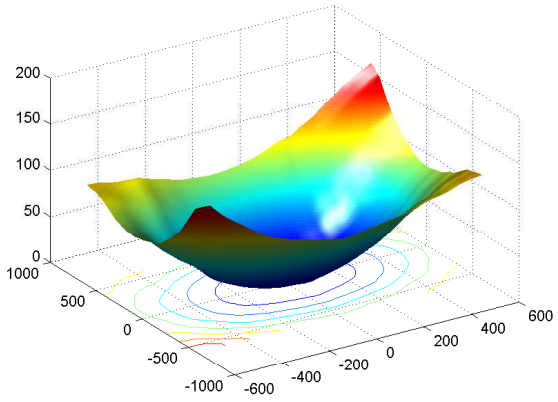
The purpose of this section is to demonstrate the performance of the sPSO algorithm on a simulation-based engineering application. The problem studied here is part of a suite of simulation-based particle swarm optimization benchmarking problems from hydrology proposed in [9]. The problem has been shown to be challenging due to the presence of local minima, a disconnected feasible region, nonconvexity, and a discontinuous objective function [10, 11, 12, 13]. Moreover, in [30] the authors show a strong dependence on initial candidates for single-search optimization methods. The objective is to control the migration of a contaminant plume by using extraction wells to alter the direction of groundwater flow. For this work the decision variables are the well locations, $\{(x_i, y_i)\}_{i=1}^2$, and the pumping rates $\{Q_i\}_{i=1}^2$ of two wells, with the possibility that the optimal design may only contain one well (that is $Q_i = 0$ m^3/s for $i = 1$ or 2).



(a) Original Griewank Function



(b) High Frequency Components of Griewank Function



(c) Surrogate Model

Figure 3: Surrogate Model of the Griewank Function after 9 iterations of sPSO

The objective function, J, is the sum of a capital installation cost J^c and an operational cost J^o given by

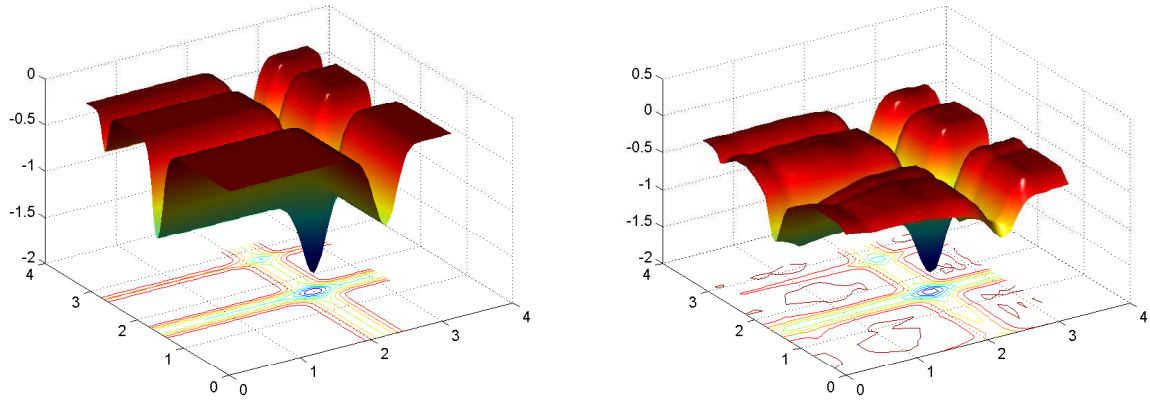
$$J = \underbrace{\sum_i \left(\nu_0 d_i^{b_0} + \nu_1 |Q_i^m|^{b_1} (z_{gs} - h^{min})^{b_2} \right)}_{J^c} + \underbrace{\int_0^{t_f} \left(\sum_i \nu_2 Q_i (h_i - z_{gs}) \right) dt}_{J^o}. \tag{8}$$

In J^c, the first term accounts for drilling and installing each well and the second term represents the additional cost for pumps for extraction wells. The operational cost, J^o, includes the lift cost associated with raising the water to the surface. More specifically, in Eq. (8) ν_j and b_j are cost coefficients and exponents, respectively, d_i is the depth of well i, which is set to the ground surface elevation z_{gs}, Q_i^m is the design pumping rate, and h^{min} is the minimum allowable hydraulic head. Including the installation term, J^c means that removing a well from the design leads to a large decrease in cost but also yields a discontinuous objective function. If, in the course of the optimization, a well rate satisfies the inequality $|Q_i| \leq 10^{-6}$ m^3/s, that well is removed from the design space and excluded from all other calculations.

The hydraulic heads, h_i for well i, also vary with the decision variables and obtaining their values for each design point requires a solution to equations that model saturated flow. The model is given by

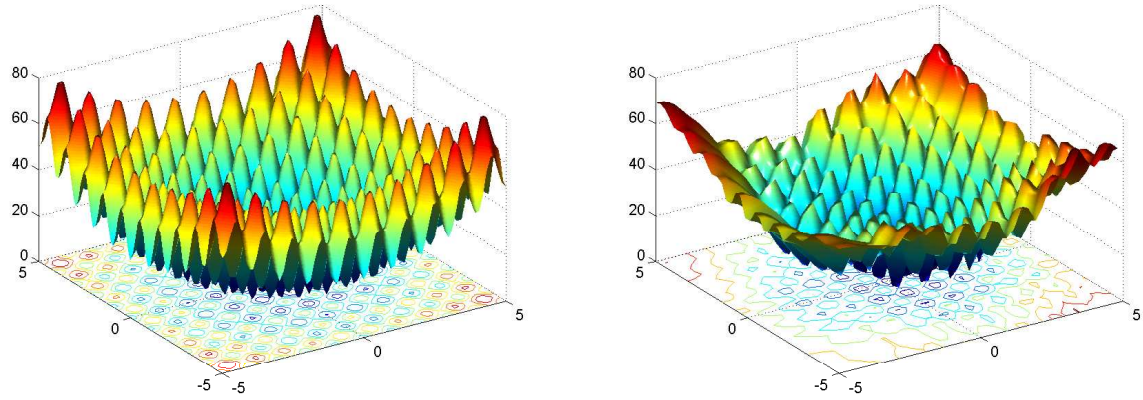
$$S_s \frac{\partial h}{\partial t} = \nabla \cdot (\mathbf{K} \cdot \nabla h) + \bar{S}, \tag{9}$$

where S_s is the specific storage coefficient, h is the hydraulic head, $\mathbf{K}$ is the hydraulic conductivity tensor, and $\bar{S}$ is a fluid source term and is where the decision variables enter into the state equation. The simulator MODFLOW2000 [31] is used to find a solution to Eg. (9).



(a) Original Michalewicz Function (b) Surrogate Model

Figure 4: Surrogate Model of the Michalewicz Function after 11 iterations of sPSO



[t]

(a) Original Rastrigin Function (b) Surrogate Model

Figure 5: Surrogate Model of the Rastrigin Function after 12 iterations of sPSO

Constraints include bounds on the well capacities and the hydraulic head at each well location. We also constrain the net pumping rate using the inequality

$$Q_T = \sum_{i=1}^{n} Q_i \geq Q_T^{max}, \tag{10}$$

where Q_T^{max} is the maximum allowable total extraction rate.

A direct approach for plume containment is to impose constraints on the concentration at specified locations. This constraint can be expressed as

$$C_j \leq C_j^{max} \tag{11}$$

where C_j is the concentration in kg/m^3 at some observation node j and C_j^{max} is the maximum allowable concentration. Evaluation of this constraint requires a solution to the contaminant transport equation

$$\frac{\partial(\theta^{\alpha} C^{\iota})}{\partial t} = \nabla \cdot (\theta^{\alpha} \mathbf{D}^{\iota} \cdot \nabla C^{\iota}) - \nabla \cdot (\mathbf{q} C^{\iota}), \tag{12}$$

where C^{ι} kg/m^3 is the concentration of species ι in the aqueous phase, θ^{α} is the volume fraction of the aqueous phase, $\mathbf{D}$ is a hydrodynamic dispersion tensor, $\mathbf{v}$ is the mean pore velocity, and $\mathbf{q}$ is the Darcy velocity. The simulator known as MT3DMS [32] is used to obtain a solution to the transport equation.

The physical domain is a $1000 \times 1000 \times 30$ unconfined aquifer implying that the bound constraints on the hydraulic head depends nonlinearly on the pumping rates. The soil parameters, boundary and initial conditions, and cost information are all specified in [9].

Table 2: sPSO and PSO performance on Groundwater Application

	PSO	sPSO
$\bar{N}$	26.25	18.05
s	3.32	3.41

5.3 sPSO Performance on Groundwater Application

For this application we considered two extraction wells allowing for six decision variables; the x and y positions of the wells along with the extraction rates. Bound constraints on the extraction wells are given by $-0.0064 \leq Q_i \leq 0$ m^3/s. The remaining linear and nonlinear constraints were implemented via weighted penalty term based on the relative constraint violations. We used the same optimization parameters as in the previous test suite, but since one groundwater and transport simulation can take up to 45 seconds, we ran the optimization only 10 times with both the sPSO and PSO algorithms. For the convergence test on this problem, an approximate global minimum is known, so a similar procedure as before was used simply for comparison of the PSO and sPSO algorithms. Table 2 summarizes the results of the optimization.

The sPSO algorithm used fewer function evaluations than the PSO algorithm, increasing the appeal of the surrogate function to improve efficiency on the background of computational expensive fitness function evaluations (here 45 seconds per run). Further speed up in term of less fitness function evaluations may be obtained by taking surrogate function in each iteration as a dynamic optimization problem as described e.g. in [33].

6 Conclusions and Discussion

Previous studies have shown that particle swarm optimization is a suitable technique when little is known about the behavior of the objective function. One strength of PSO is that the method requires only a few initial parameters. Initial iterates obtained by any kind of expert knowledge are not required to start PSO.

Surrogate functions were incorporated into the PSO algorithm to improve efficiency on engineering problems. Since the global PSO is used on a variety of engineering problems, [34, 35, 4, 5], we also start with the global PSO variant as a building block for the sPSO. However, many enhanced variations of the PSO algorithm exist, each excelling on different types of problems. This encourages the development of alternative sPSO algorithms based on other PSO variants, such as the local PSO, adaptive PSO, or discrete PSO, [36]. For example, on high dimensional problems, the local PSO may have more reliable convergence properties than the global PSO. In practice, we have found that in general, the local PSO begins to outperform the global PSO on problems with more than 10 dimensions. Thus, applications with a large number of decision variables may consider using the ideas presented in this work to create a sPSO variant from a local PSO topology.

We provide a new framework for including surrogate functions to the PSO framework. The surrogate is built using intermediate function values from the optimization and the minimum of the surrogate helps guide the search phase. No additional objective function evaluations are required to build these surrogates. We have shown that the proposed sPSO can result in significantly fewer objective function evaluations than the PSO. This is of particular interest for simulation based problems where the simulation typically is much more computationally expensive than the cost to maintain the surrogate function.

References

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization," *IEEE Conference on Neural Networks*, vol. 4, pp. 1942–1948, 1995.

[2] C.-J. Liao, C.-T. Tseng, and P. Luarn, "A discrete version of particle swarm optimization for flowshop scheduling problems," *Computers & Operations Research*, vol. 34, pp. 3099–3111, 2007.

[3] J. Robinson and Y. Rahmat-Samii, "Particle swarm optimization in electromagnetics," *IEEE Transactions on Antennas and Propagation*, vol. 52, no. 2, pp. 397–407, 2004.

[4] H. Yoshida, K. Kawata, Y. Fukuyama, S. Takayama, and Y. Nakanishi, "A particle swarm optimization for reactive power and voltage control considering voltage security assessment," *IEEE Transactions on Power Systems*, vol. 15, no. 4, pp. 1232–1239, 2000.

[5] L. S. Matott, S. L. Bartelt-Hunt, A. J. Rabideau, and K. R. Fowler, "Application of heuristic optimization techniques and algorithm tuning to multilayed sorptive barrier design," *Environmental Science and Technology*, vol. 40, pp. 4354–6360, 2006.

[6] J. Sacks, S. B. Schiller, and W. J. Welch, "Design for computer experiments," *Technometrics*, vol. 31, pp. 41–47, 1989.

[7] R. Jin, X. Du, and W. Chen, "The use of metamodeling techniques for optimization under uncertainty," ASME Design Automation Conference, 2001.

[8] R. Jin, W. Chen, and T. W. Simpson, "Comparative studies of metamodelling techniques under multiple modelling criteria," *Structural and Multidisciplinary Optimization*, vol. 23, pp. 1–13, 2001.

[9] A. S. Mayer, C. T. Kelley, and C. T. Miller, "Optimal design for problems involving flow and transport phenomena in saturated subsurface systems," *Advances in Water Resources*, vol. 12, pp. 1233–1256, 2002.

[10] L. S. Mattot, A. J. Rabideau, and J. R. Craig, "Pump-and-treat optimization using analytic element method flow models," *Advances in Water Resources*, vol. 29, pp. 760–775, 2006.

[11] T. Hemker, K. R. Fowler, and O. von Stryk, "Derivative-free optimization methods for handling fixed costs in optimal groundwater remediation design," in *Proc. of the CMWR XVI - Computational Methods in Water Resources*, 19-22 June 2006.

[12] G. A. Gray and K. R. Fowler, "Approaching the groundwater remediation problem using multifidelity optimization," in *Proc. of the CMWR XVI - Computational Methods in Water Resources*, 19-22 June 2006.

[13] T. Hemker, K. R. Fowler, M. W. Farthing, and O. von Stryk, "A mixed-integer simulation-based optimization approach with surrogate functions in water resources management," *Optimization and Engineering*, p. to appear, 2008.

[14] M. Clerc and J. Kennedy, "The particle swarm explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 58–73, 2002.

[15] H. Fan, "A modification to particle swarm optimization algorithm," *Engineering Computations*, vol. 19, no. 9, pp. 970–989, 2002.

[16] I. C. Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection," *Information Processing Letters*, vol. 85, pp. 317–325, 2003.

[17] M. Jiang, Y. P. Luo, and S. Y. Yang, "Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm," *Information Processing Letters*, vol. 102, pp. 8–16, 2007.

[18] G. Matheron, "Principles of geostatistics," *Econom. Geol.*, vol. 58, pp. 1246 – 1266, 1963.

[19] J. Martin and T. Simpson, "A study on the use of kriging models to approximate deterministic computer models," in *Proceedings of DETC'03*, 2003.

[20] J. Koehler and A. Owen, "Computer experiments," *Handbook of Statistics*, vol. 13, pp. 261–308, 1996.

[21] S. N. Lophaven, H. B. Nielson, and J. Søndergaard, *DACE: A MATLAB Kriging Toolbox*, Technical University of Denmark.

[22] S. N. Lophaven, H. B. Nielsen, and S. Søndergaard, "Aspects of the Matlab toolbox DACE," IMM, Tech. Rep. IMM-TR-2002-13, August 2002.

[23] M. Iqbal and M. A. Montes de Oca, "An estimation of distribution particle swarm optimization algorithm," in *Ant Colony Optimization and Swarm Intelligence. 5th International Workshop, ANTS 2006, Brussels*, ser. LNCS, D. M. et. al., Ed., vol. 4150/2006. Berlin, Germany: Springer Verlag, August 2006, pp. 72–83. [Online]. Available: http://www.cs.kent.ac.uk/pubs/2006/2601

[24] J. J. More, B. S. Garbow, and K. E. Hillstrom, "Testing unconstrained optimization," *ACM Transactions on Mathematical Software*, vol. 7, no. 1, pp. 17–41, 1981.

[25] D. H. Ackley, *A connectionist machine for genetic hillclimbing*. Norwell, MA, USA: Kluwer Academic Publishers, 1987.

[26] H.-P. Schwefel, *Numerical Optimization of Computer Models*. New York, NY, USA: John Wiley & Sons, Inc., 1981.

[27] L. A. Rastrigin, "Extremal control systems," *Theoretical Foundations of Engineering Cybernetics Series (in Russian)*.

[28] T. Back, D. B. Fogel, and Z. Michalewicz, *Handbook of Evolutionary Computation*. CRC Press, 1997.

[29] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin, Heidelberg, New York: Springer-Verlag, 1992.

[30] K. R. Fowler, J. P. Reese, C. E. Kees, J. E. Dennis, Jr., C. T. Kelley, C. T. Miller, C. Audet, A. J. Booker, G. Couture, R. W. Darwin, M. W. Farthing, D. E. Finkel, J. M. Gablonsky, G. Gray, and T. G. Kolda, "A comparison of derivative-free optimization methods for water supply and hydraulic capture community problems," *Accepted to Advances in Water Resources*, 2008.

[31] C. Zheng, M. C. Hill, and P. A. Hsieh, *MODFLOW2000, The U.S.G.S Survey Modular Ground-Water Model User Guide to the LMT6 Package, the Linkage With MT3DMS for Multispecies Mass Transport Modeling*, user's guide ed., USGS, 2001.

[32] C. Zheng and P. P. Wang, *MT3DMS: A Modular Three-Dimensional Multispecies Transport Model for Simulation of Advection, Dispersion, and Chemical Reactions of Contaminants in Groundwater Systems*, documentation and user's guide ed., 1999.

[33] T. Blackwell, "Particle swarm optimization in dynamic environments," in *Evolutionary Computation in Dynamic and Uncertain Environments*. Springer, 2007, pp. 29–49.

[34] H. H. BALCI and J. F. VALENZUELA, "Scheduling electric power generators using particle swarm optimization combined with the lagrangian relaxation method," *International Journal of Applied Mathematics and Computer Sciences*, vol. 14, no. 3, pp. 411–421, 2004.

[35] L. S. Matott, *OSTRICH: An Optimization Software Tool; Documentation and UserâĂŹs Guide*, State University of New York at Buffalo, http://www.groundwater.buffalo.edu/software/Download.php.

[36] K. Y. Lee and M. A. El-Sharkawi, Eds., *Modern Heuristic Optimization Techniques: Theory and Applications to Power Systems*. John Wiley and Sons Inc., 2008.