Jaco F. Schutte
Department of Mechanical & Aerospace Engineering,
University of Florida,
Gainesville, FL 32611-6250

Byung-Il Koh
Department of Electrical and Computer Engineering,
University of Florida,
Gainesville, FL 32611-6250

Jeffrey A. Reinbolt
Department of Mechanical & Aerospace Engineering,
University of Florida,
Gainesville, FL 32611-6250

Raphael T. Haftka
Department of Mechanical & Aerospace Engineering,
University of Florida,
Gainesville, FL 32611-6250

Alan D. George
Department of Electrical & Computer Engineering, University of Florida,
Gainesville, FL 32611-6250

Benjamin J. Fregly[1]
e-mail: fregly@ufl.edu
Department of Mechanical & Aerospace Engineering and Department of Biomedical Engineering,
University of Florida,
Gainesville, FL 32611-6250

Evaluation of a Particle Swarm Algorithm For Biomechanical Optimization

Optimization is frequently employed in biomechanics research to solve system identification problems, predict human movement, or estimate muscle or other internal forces that cannot be measured directly. Unfortunately, biomechanical optimization problems often possess multiple local minima, making it difficult to find the best solution. Furthermore, convergence in gradient-based algorithms can be affected by scaling to account for design variables with different length scales or units. In this study we evaluate a recently-developed version of the particle swarm optimization (PSO) algorithm to address these problems. The algorithm's global search capabilities were investigated using a suite of difficult analytical test problems, while its scale-independent nature was proven mathematically and verified using a biomechanical test problem. For comparison, all test problems were also solved with three off-the-shelf optimization algorithms—a global genetic algorithm (GA) and multistart gradient-based sequential quadratic programming (SQP) and quasi-Newton (BFGS) algorithms. For the analytical test problems, only the PSO algorithm was successful on the majority of the problems. When compared to previously published results for the same problems, PSO was more robust than a global simulated annealing algorithm but less robust than a different, more complex genetic algorithm. For the biomechanical test problem, only the PSO algorithm was insensitive to design variable scaling, with the GA algorithm being mildly sensitive and the SQP and BFGS algorithms being highly sensitive. The proposed PSO algorithm provides a new off-the-shelf global optimization option for difficult biomechanical problems, especially those utilizing design variables with different length scales or units.
[DOI: 10.1115/1.1894388]

1 Introduction

Optimization methods are used extensively in biomechanics research to predict movement-related quantities that cannot be measured experimentally. Forward dynamic, inverse dynamic, and inverse static optimizations have been used to predict muscle, ligament, and joint contact forces during experimental or predicted movements (e.g., [1–12]). System identification optimizations have been employed to tune a variety of musculoskeletal model parameters to experimental movement data (e.g., [13–17]). Image matching optimizations have been performed to align implant and bone models to *in vivo* fluoroscopic images collected during loaded functional activities (e.g., [18–20]).

Since biomechanical optimization problems are typically nonlinear in the design variables, gradient-based nonlinear programming has been the most widely used optimization method. The increasing size and complexity of biomechanical models has also led to the parallelization of gradient-based algorithms, since gradient calculations can be easily distributed to multiple processors [1–3]. However, gradient-based optimizers can suffer from several important limitations. They are local rather than global by nature and so can be sensitive to the initial guess. Experimental or numerical noise can exacerbate this problem by introducing multiple local minima into the problem. For some problems, multiple local minima may exist due to the nature of the problem itself. In most situations, the necessary gradient values cannot be obtained analytically, and finite difference gradient calculations can be sensitive to the selected finite difference step size. Furthermore, the use of design variables with different length scales or units can produce poorly scaled problems that converge slowly or not at all [21,22], necessitating design variable scaling to improve performance.

Motivated by these limitations and improvements in computer speed, recent studies have begun investigating the use of nongradient global optimizers for biomechanical applications. Neptune [4] compared the performance of a simulated annealing (SA) algorithm with that of downhill simplex (DS) and sequential quadratic programming (SQP) algorithms on a forward dynamic optimization of bicycle pedaling utilizing 27 design variables. Simulated annealing found a better optimum than the other two methods and in a reasonable amount of CPU time. More recently, Soest and Casius [5] evaluated a parallel implementation of a genetic algorithm (GA) using a suite of analytical tests problems with up to 32 design variables and forward dynamic optimizations of jumping and isokinetic cycling with up to 34 design variables. The genetic algorithm generally outperformed all other algorithms tested, including SA, on both the analytical test suite and the movement optimizations.

In this study we evaluate a recent addition to the arsenal of

[1]Address correspondence to: B. J. Fregly, Ph.D., Assistant Professor, Department of Mechanical & Aerospace Engineering, 231 MAE-A Building, P.O. Box 116250, University of Florida, Gainesville, FL 32611-6250. Phone: (352) 392-8157; fax: (352) 392-7303.

Contributed by the Bioengineering Division for publication In the JOURNAL OF BIOMECHANICAL ENGINEERING. Manuscript received: July 9, 2003; revision received: January 1, 2005. Associate Editor: Maury Hull.

global optimization methods—particle swarm-optimization (PSO)—for use on biomechanical problems. A recently developed variant of the PSO algorithm is used for the investigation. The algorithm's global search capabilities are evaluated using a previously published suite of difficult analytical test problems with multiple local minima [5], while its insensitivity to design variable scaling is proven mathematically and verified using a biomechanical test problem. For both categories of problems, PSO robustness, performance, and scale independence are compared to that of three off-the-shelf optimization algorithms—a genetic algorithm (GA), a sequential quadratic programming algorithm (SQP), and a Broydon-Fletcher-Goldfarb-Shanno (BFGS) quasi-Newton algorithm. In addition, previously published results [5] for the analytical test problems permit a comparison with a more complex GA algorithm (GA*), a simulated annealing algorithm (SA), a different SQP algorithm (SQP*), and a downhill simplex (DS) algorithm.

2 Theory

2.1 Particle Swarm Algorithm. Particle swarm optimization is a stochastic global optimization approach introduced by Kennedy and Eberhart [23]. The method's strength lies in its simplicity, being easy to code and requiring few algorithm parameters to define convergence behavior. The following is a brief introduction to the operation of the particle swarm algorithm based on a recent implementation by Fourie and Groenwold [24] incorporating dynamic inertia and velocity reduction.

Consider a swarm of p particles, where each particle's position $\mathbf{x}_k^i$ represents a possible solution point in the problem design space $\mathbf{D}$. For each particle i, Kennedy and Eberhart [23] proposed that the position $\mathbf{x}_{k+1}^i$ be updated in the following manner:

$$\mathbf{x}_{k+1}^i = \mathbf{x}_k^i + \boldsymbol{\nu}_{k+1}^i, \tag{1}$$

with a pseudovelocity $\boldsymbol{\nu}_{k+1}^i$ calculated as follows:

$$\boldsymbol{\nu}_{k+1}^i = w_k \boldsymbol{\nu}_k^i + c_1 r_1 (\mathbf{p}_k^i - \mathbf{x}_k^i) + c_2 r_2 (\mathbf{g}_k - \mathbf{x}_k^i) \tag{2}$$

Here, subscript k indicates a (unit) pseudotime increment. The point $\mathbf{p}_k^i$ is the best-found cost location by particle i up to time step k, which represents the cognitive contribution to the search vector $\boldsymbol{\nu}_{k+1}^i$. Each component of $\boldsymbol{\nu}_{k+1}^i$ is constrained to be less than or equal to a maximum value defined in $\boldsymbol{\nu}_{k+1}^{\max}$. The point $\mathbf{g}_k$ is the global best-found position among all particles in the swarm up to time k and forms the social contribution to the velocity vector. Cost function values associated with $\mathbf{p}_k^i$ and $\mathbf{g}_k$ are denoted by f_{best}^i and f_{best}^g respectively. Random numbers r_1 and r_2 are uniformly distributed in the interval [0,1]. Shi and Eberhart [25] proposed that the cognitive and social scaling parameters c_1 and c_2 be selected such that $c_1 = c_2 = 2$ to allow the product $c_1 r_1$ or $c_2 r_2$ to have a mean of 1. The result of using these proposed values is that the particles overshoot the attraction points $\mathbf{p}_k^i$ and $\mathbf{g}_k$ half the time, thereby maintaining separation in the group and allowing a greater area to be searched than if the particles did not overshoot. The variable w_k, set to 1 at initialization, is a modification to the original PSO algorithm [23]. By reducing its value dynamically based on the cost function improvement rate, the search area is gradually reduced [26]. This dynamic reduction behavior is defined by w_d, the amount by which the inertia w_k is reduced, ν_d, the amount by which the maximum velocity $\boldsymbol{\nu}_{k+1}^{\max}$ is reduced, and d, the number of iterations with no improvement in $\mathbf{g}_k$ before these reductions take place [24] (see algorithm flow description below).

Initialization of the algorithm involves several important steps. Particles are randomly distributed throughout the design space, and particle velocities $\boldsymbol{\nu}_0^i$ are initialized to random values within the limits $0 \leq \boldsymbol{\nu}_0^i \leq \boldsymbol{\nu}_0^{\max}$. The particle velocity upper limit $\boldsymbol{\nu}_0^{\max}$ is calculated as a fraction of the distance between the upper and lower bound on variables in the design space $\boldsymbol{\nu}_0^{\max} = \kappa(\mathbf{x}_{UB} - \mathbf{x}_{LB})$ with $\kappa = 0.5$ as suggested in [26]. Iteration counters k and t are set to 0. Iteration counter k is used to monitor the total number of swarm iterations, while iteration counter t is used to monitor the number of swarm iterations since the last improvement in $\mathbf{g}_k$. Thus, t is periodically reset to zero during the optimization while k is not.

Table 1 Standard PSO algorithm parameters used in the present study

Parameter	Description	Value
p	Population size (number of particles)	20
c_1	Cognitive trust parameter	2.0
c_2	Social trust parameter	2.0
w_0	Initial inertia	1
w_d	Inertia reduction parameter	0.05
κ	Bound on velocity fraction	0.5
ν_d	Velocity reduction parameter	0.05
d	Dynamic inertia/velocity reduction delay (function evaluations)	200

The algorithm flow can be represented as follows:

1. Initialize
   - (a) Set constants κ, c_1, c_2, $k_{\max}$, $\boldsymbol{\nu}_0^{\max}$, w_0, ν_d, w_d, and d
   - (b) Set counters $k=0$, $t=0$. Set random number seed
   - (c) Randomly initialize particle positions $\mathbf{x}_0^i \in \mathbf{D}$ in $\Re$ for $i=1,\ldots,p$
   - (d) Randomly initialize particle velocities $0 \leq \boldsymbol{\nu}_0^i \leq \boldsymbol{\nu}_0^{\max}$ for $i=1,\ldots,p$
   - (e) Evaluate cost function values f_0^i using design space coordinates $\mathbf{x}_0^i$ for $i=1,\ldots,p$
   - (f) Set $f_{\text{best}}^i = f_0^i$ and $\mathbf{p}_0^i = \mathbf{x}_0^i$ for $i=1,\ldots,p$
   - (g) Set f_{best}^g to best f_{best}^i and $\mathbf{g}_0$ to corresponding $\mathbf{x}_0^i$
2. Optimize
   - (a) Update particle velocity vectors $\boldsymbol{\nu}_{k+1}^i$ using Eq. (2)
   - (b) If $\boldsymbol{\nu}_{k+1}^i > \boldsymbol{\nu}_{k+1}^{\max}$ for any component, then set that component to its maximum allowable value
   - (c) Update particle position vectors $\mathbf{x}_{k+1}^i$ using Eq. (1)
   - (d) Evaluate cost function values f_{k+1}^i using design space coordinates $\mathbf{x}_{k+1}^i$ for $i=1,\ldots,p$
   - (e) If $f_{k+1}^i \leq f_{\text{best}}^i$, then $f_{\text{best}}^i = f_{k+1}^i$, $\mathbf{p}_{k+1}^i = \mathbf{x}_{k+1}^i$ for $i=1,\ldots,p$
   - (f) If $f_{k+1}^i \leq f_{\text{best}}^g$, then $f_{best}^g = f_{k+1}^i$, $\mathbf{g}_{k+1} = \mathbf{x}_{k+1}^i$ for $i=1,\ldots,p$
   - (g) If f_{best}^g was improved in (e), then reset $t=0$, otherwise increment t
   - (h) If the maximum number of function evaluations is exceeded, then go to 3
   - (i) If $t=d$, then multiply w_{k+1} by $(1-w_d)$ and $\boldsymbol{\nu}_{k+1}^{\max}$ by $(1-\nu_d)$
   - (j) Increment k
   - (k) Go to 2(a)
3. Report results
4. Terminate

This algorithm was coded in the C programming language by the lead author [27] and was used for all PSO analyses performed in the present study. A standard population size of 20 particles was used for all runs, and other algorithm parameters were also selected based on standard recommendations (Table 1) [27–29]. The C source code for our PSO algorithm is freely available at http://www.mae.ufl.edu/~fregly/downloads/pso.zip.

2.2 Analysis of Scale Sensitivity. One of the benefits of the PSO algorithm is its insensitivity to design variable scaling. To

prove this characteristic, we will use a proof by induction to show that all particles follow an identical path through the design space regardless of how the design variables are scaled. In actual PSO runs intended to investigate this property, use of the same random seed in scaled and unscaled cases will ensure that an identical sequence of random r_1 and r_2 values are produced by the computer throughout the course of the optimization.

Consider an optimization problem with n design variables. An n-dimensional constant scaling vector $\boldsymbol{\zeta}$ can be used to scale any or all dimensions of the problem design space:

$$\boldsymbol{\zeta} = \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \zeta_3 \\ \vdots \\ \zeta_n \end{bmatrix} \tag{3}$$

We wish to show that for any time step $k \geqslant 0$,

$$\boldsymbol{\nu}_k' = \boldsymbol{\zeta}\boldsymbol{\nu}_k \tag{4}$$

$$\mathbf{x}_k' = \boldsymbol{\zeta}\mathbf{x}_k \tag{5}$$

where $\mathbf{x}_k$ and $\boldsymbol{\nu}_k$ (dropping superscript i) are the unscaled position and velocity, respectively, of an individual particle and $\mathbf{x}_k' = \boldsymbol{\zeta}\mathbf{x}_k$ and $\boldsymbol{\nu}_k' = \boldsymbol{\zeta}\boldsymbol{\nu}_k$ are the corresponding scaled versions.

First, we must show that our proposition is true for the base case, which involves initialization ($k=0$) and the first time step ($k=1$). Applying the scaling vector $\boldsymbol{\zeta}$ to an individual particle position $\mathbf{x}_0$ during initialization produces a scaled particle position $\mathbf{x}_0'$:

$$\mathbf{x}_0' = \boldsymbol{\zeta}\mathbf{x}_0 \tag{6}$$

This implies that

$$\mathbf{p}_0' = \boldsymbol{\zeta}\mathbf{p}_0, \quad \mathbf{g}_0' = \boldsymbol{\zeta}\mathbf{g}_0 \tag{7}$$

In the unscaled case, the pseudovelocity is calculated as

$$\boldsymbol{\nu}_0 = \kappa(\mathbf{x}_{UB} - \mathbf{x}_{LB}) \tag{8}$$

In the scaled case, this becomes

$$\boldsymbol{\nu}_0' = \kappa(\mathbf{x}_{UB}' - \mathbf{x}_{LB}') = \kappa(\boldsymbol{\zeta}\mathbf{x}_{UB} - \boldsymbol{\zeta}\mathbf{x}_{LB}) = \boldsymbol{\zeta}[\kappa(\mathbf{x}_{UB} - \mathbf{x}_{LB})] = \boldsymbol{\zeta}\boldsymbol{\nu}_0 \tag{9}$$

From Eqs. (1) and (2) and these initial conditions, the particle pseudovelocity and position for the first time step can be written as

$$\boldsymbol{\nu}_1 = w_0\boldsymbol{\nu}_0 + c_1r_1(\mathbf{p}_0 - \mathbf{x}_0) + c_2r_2(\mathbf{g}_0 - \mathbf{x}_0) \tag{10}$$

$$x_1 = x_0 + \nu_1 \tag{11}$$

in the unscaled case and

$$\begin{aligned} \boldsymbol{\nu}_1' &= w_0\boldsymbol{\nu}_0' + c_1r_1(\mathbf{p}_0' - \mathbf{x}_0') + c_2r_2(\mathbf{g}_0' - \mathbf{x}_0') \\ &= w_0\boldsymbol{\zeta}\boldsymbol{\nu}_0 + c_1r_1(\boldsymbol{\zeta}\mathbf{p}_0 - \boldsymbol{\zeta}\mathbf{x}_0) + c_2r_2(\boldsymbol{\zeta}\mathbf{g}_0 - \boldsymbol{\zeta}\mathbf{x}_0) \\ &= \boldsymbol{\zeta}[w_0\boldsymbol{\nu}_0 + c_1r_1(\mathbf{p}_0 - \mathbf{x}_0) + c_2r_2(\mathbf{g}_0 - \mathbf{x}_0)] = \boldsymbol{\zeta}\boldsymbol{\nu}_1 \end{aligned} \tag{12}$$

$$\mathbf{x}_1' = \mathbf{x}_0' + \boldsymbol{\nu}_1' = \boldsymbol{\zeta}\mathbf{x}_0 + \boldsymbol{\zeta}\boldsymbol{\nu}_1 = \boldsymbol{\zeta}[\mathbf{x}_0 + \boldsymbol{\nu}_1] = \boldsymbol{\zeta}\mathbf{x}_1 \tag{13}$$

in the scaled case. Thus, our proposition is true for the base case.

Next, we must show that our proposition is true for the inductive step. If we assume that our proposition holds for any time step $k=j$, we must prove that it also holds for time step $k=j+1$. We begin by replacing subscript k with subscript j in Eqs. (4) and (5). If we then replace subscript 0 with subscript j and subscript 1 with subscript $j+1$ in Eqs. (12) and (13), we arrive at Eqs. (4) and (5) where subscript k is replaced by subscript $j+1$. Thus, our proposition is true for any time step $j+1$.

Consequently, since the base case is true and the inductive step is true, Eqs. (4) and (5) are true for all $k \geqslant 0$. From Eqs. (4) and (5), we can conclude that any linear scaling of the design variables (or subset thereof) will have no effect on the final or any intermediate result of the optimization, since all velocities and positions are scaled accordingly. This fact leads to identical step intervals being taken in the design space for a scaled and an unscaled version of the same problem, assuming infinite precision in all calculations.

In contrast, gradient-based optimization methods are often sensitive to design variable scaling due to algorithmic issues and numerical approximations. First derivative methods are sensitive because of algorithmic issues, as illustrated by a simple example. Consider the following minimization problem with two design variables (x,y) where the cost function is

$$x^2 + \frac{y^2}{100} \tag{14}$$

with initial guess (1,1). A scaled version of the same problem can be created by letting $\tilde{x}=x$, $\tilde{y}=y/10$ so that the cost function becomes

$$\tilde{x}^2 + \tilde{y}^2 \tag{15}$$

with initial guess (1,10). Taking first derivatives of each cost function with respect to the corresponding design variables and evaluating at the initial guesses, the search direction for the unscaled problem is along a line rotated 5.7° from the positive x axis and for the scaled problem along a line rotated 45°. To reach the optimum in a single step, the unscaled problem requires a search direction rotated 84.3° and the scaled problem 45°. Thus, the scaled problem can theoretically reach the optimum in a single step while the unscaled problem cannot due to the effect of scaling on the calculated search direction.

Second derivative methods are sensitive to design variable scaling because of numerical issues related to approximation of the Hessian (second derivative) matrix. According to Gill et al. [21], Newton methods utilizing an exact Hessian matrix will be insensitive to design variable scaling as long as the Hessian matrix remains positive definite. However, in practice, exact Hessian calculations are almost never available, necessitating numerical approximations via finite differencing. Errors in these approximations result in different search directions for scaled versus unscaled versions of the same problem. Even a small amount of design variable scaling can significantly affect the Hessian matrix so that design variable changes of similar magnitude will not produce comparable magnitude cost function changes [21]. Common gradient-based algorithms that employ an approximate Hessian include Newton and quasi-Newton nonlinear programming methods such as BFGS, SQP methods, and nonlinear least-squares methods such as Levenberg—Marquardt [21]. A detailed discussion of the influence of design variable scaling on optimization algorithm performance can be found in Gill et al. [21].

3 Methodology

3.1 Optimization Algorithms. In addition to our PSO algorithm, three off-the-shelf optimization algorithms were applied to all test problems (analytical and biomechanical—see below) for comparison purposes. One was a global GA algorithm developed by Deb [30–32]. This basic GA implementation utilizes one mutation operator and one crossover operator along with real encoding to handle continuous variables. The other two algorithms were commercial implementations of gradient-based SQP and BFGS algorithms (VisualDOC, Vanderplaats R & D, Colorado Springs, CO).

All four algorithms (PSO, GA, SQP, and BFGS) were parallelized to accommodate the computational demands of the biomechanical test problem. For the PSO algorithm, parallelization was performed by distributing individual particle function evaluations to different processors as detailed in [33]. For the GA algorithm, individual chromosome function evaluations were parallelized as

described in [5]. Finally, for the SQP and BFGS algorithms, finite difference gradient calculations were performed on different processors as outlined in [34]. A master-slave paradigm using the Message Passing Interface (MPI) [35,36] was employed for all parallel implementations. Parallel optimizations for the biomechanical test problem were run on a cluster of Linux-based PCs in the University of Florida High-performance Computing and Simulation Research Laboratory (1.33 GHz Athlon CPUs with 256 MB memory on a 100 Mbps switched Fast Ethernet network).

While the PSO algorithm used standard algorithm parameters for all optimization runs, minor algorithm tuning was performed on the GA, SQP, and BFGS algorithms for the biomechanical test problem. The goal was to give these algorithms the best possible chance for success against the PSO algorithm. For the GA algorithm, preliminary optimizations were performed using population sizes ranging from 40 to 100. It was found that for the specified maximum number of function evaluations, a population size of 60 produced the best results. Consequently, this population size was used for all subsequent optimization runs (analytical and biomechanical). For the SQP and BFGS algorithms, automatic tuning of the finite difference step size (FDSS) was performed separately for each design variable. At the start of each gradient-based run, forward and central difference gradients were calculated for each design variable beginning with a relative FDSS of 10^{-1}. The step size was then incrementally decreased by factors of 10 until the absolute difference between forward and central gradient results was a minimum. This approach was taken since the amount of noise in the biomechanical test problem prevented a single stable gradient value from being calculated over a wide range of FDSS values (see Sec. 5). The forward difference step size automatically selected for each design variable was used for the remainder of the run.

3.2 Analytical Test Problems. The global search capabilities of our PSO implementation were evaluated using a suite of difficult analytical test problems previously published by Soest and Casius [5]. In that study, each problem in the suite was evaluated using four different optimizers: SA, GA*, SQP*, and DS, where an asterisk indicates a different version of an algorithm used in our study. One thousand optimization runs were performed with each optimizer starting from random initial guesses and using standard optimization algorithm parameters. Each run was terminated based on a predefined number of function evaluations for the particular problem being solved. We followed an identical procedure with our four algorithms to permit a comparison between our results and those published in [5]. Since two of the algorithms used in our study (GA and SQP) were of the same general category as algorithms used [5] (GA* and SQP*), comparisons could be made between different implementations of the same general algorithm. Failed PSO and GA runs were allowed to use up the full number of function evaluations, whereas failed SQP and BFGS runs were restarted from new random initial guesses until the full number of function evaluations was completed. Only 100 rather than 1000 runs were performed with the SQP and BFGS algorithms due to a database size problem in the VisualDOC software.

A detailed description of the six analytical test problems can be found in Soest and Casius [5]. Since the design variables for each problem possessed the same absolute upper and lower bound and appeared in the cost function in a similar form, design variable scaling was not an issue in these problems. The six analytical test problems are described briefly below.

H_1: This simple two-dimensional function [5] has several local maxima and a global maximum of 2 at the coordinates (8.6998, 6.7665).

$$H_1(x_1,x_2) = \frac{\sin^2\left(x_1 - \frac{x_2}{8}\right) + \sin^2\left(x_2 - \frac{x_1}{8}\right)}{d+1} \tag{16}$$

where

$$x_1,x_2 \in [-100,100]$$

$$d = \sqrt{(x_1 - 8.6998)^2 + (x_2 - 6.7665)^2}$$

Ten-thousand function evaluations were used for this problem.

H_2: This inverted version of the F6 function used by Schaffer et al. [37] has two dimensions with several local maxima around the global maximum of 1.0 at (0,0).

$$H_2(x_1,x_2) = 0.5 - \frac{\sin^2(\sqrt{x_1^2 + x_2^2}) - 0.5}{(1 + 0.001(x_1^2 + x_2^2))^2} \tag{17}$$

$$x_1,x_2 \in [-100,100]$$

This problem was solved using 20,000 function evaluations per optimization run.

H_3: This test function from Corana et al. [38] was used with dimensionality $n=4$, 8, 16, and 32. The function contains a large number of local minima (on the order of 10^{4n}) with a global minimum of 0 at $|x_i| < 0.05$.

$$H_3(x_1,\ldots,x_n) = \sum_{i=1}^{n} \begin{cases} (t \cdot \mathrm{sgn}(z_i) + z_i)^2 \cdot c \cdot d_i & \text{if} \quad |x_i - z_i| < t \\ d_i \cdot x_i^2 & \text{otherwise} \end{cases} \tag{18}$$

$$x_i \in [-1000,1000]$$

where

$$z_i = \left\lfloor \left| \frac{x_i}{s} \right| + 0.49999 \right\rfloor \cdot \mathrm{sgn}(x_i) \cdot s, \quad c = 0.15,$$

$$s = 0.2, \quad t = 0.05, \quad \text{and} \quad d_i = \begin{cases} 1 & i = 1,5,9,\ldots \\ 1000 & i = 2,6,10,\ldots \\ 10 & i = 3,7,11,\ldots \\ 100 & i = 4,8,12,\ldots \end{cases}$$

The use of the floor function in Eq. (18) makes the search space for this problem the most discrete of all problems tested. The number of function evaluations used for this problem was 50,000 ($n=4$), 100,000 ($n=8$), 200,000 ($n=16$), and 400,000 ($n=32$).

For all of the analytical test problems, an algorithm was considered to have succeeded if it converged to within 10^{-3} of the known optimum cost function value within the specified number of function evaluations [5].

3.3 Biomechanical Test Problem. In addition to these analytical test problems, a biomechanical test problem was used to evaluate the scale-independent nature of the PSO algorithm. Although our PSO algorithm is theoretically insensitive to design variable scaling, numerical round-off errors and implementation details could potentially produce a scaling effect. Running the other three algorithms on scaled and unscaled versions of this test problem also permitted investigation of the extent to which other algorithms are influenced by design variable scaling.

The biomechanical test problem involved determination of an ankle joint kinematic model that best matched noisy synthetic (i.e., computer generated) movement data. Similar to [13], the ankle was modeled as a three-dimensional linkage with two nonintersecting pin joints defined by 12 subject-specific parameters (Fig. 1). These parameters represent the positions and orientations of the talocrural and subtalar joint axes in the tibia, talus, and calcaneous. Position parameters were in units of centimeters and orientation parameters in units of radians, resulting in parameter

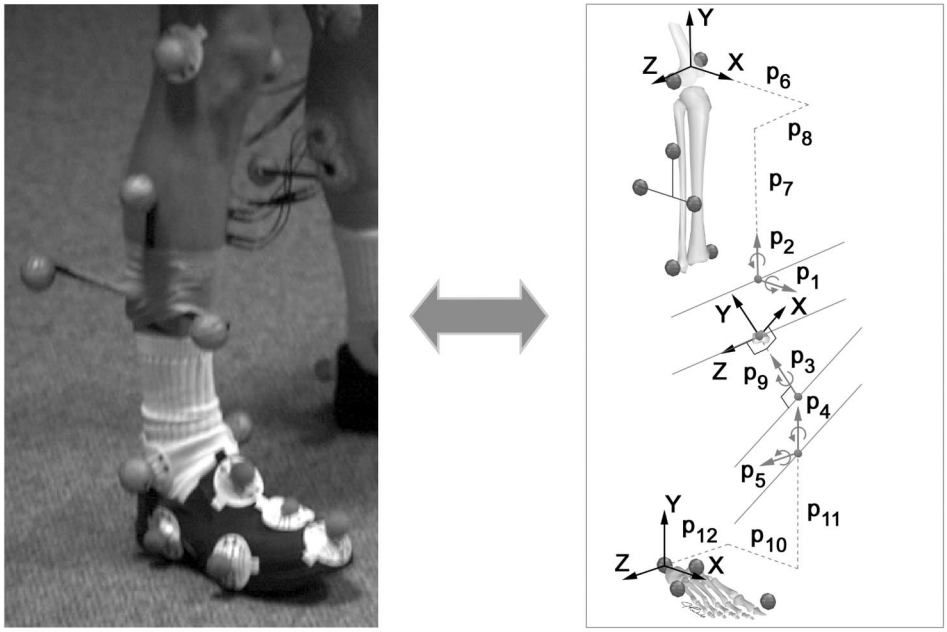

Fig. 1 Experimental shank and foot surface marker configuration (left) for developing a subject-specific kinematic ankle model defined by 12 parameters p_1 through p_{12} (right). Each parameter defines the position or orientation of a joint axis in one of the body segments.

values of varying magnitude. This model was part of a larger 27 degree-of-freedom (DOF) full-body kinematic model used to optimize other joints as well [15].

Given this model structure, noisy synthetic movement data were generated from a nominal model for which the "true" model parameters were known. Joint parameters for the nominal model along with a nominal motion were derived from *in vivo* experimental movement data using the optimization methodology described below. Next, three markers were attached to the tibia and calcaneous segments in the model at locations consistent with the experiment, and the 27 model DOFs were moved through their nominal motions. This process created synthetic marker trajectories consistent with the nominal model parameters and motion and also representative of the original experimental data. Finally, numerical noise was added to the synthetic marker trajectories to emulate skin and soft tissue movement artifacts. For each marker coordinate, a sinusoidal noise function was used with uniformly distributed random period, phase, and amplitude (limited to a maximum of ±1 cm). The values of the sinusoidal parameters were based on previous studies reported in the literature [39,40].

An unconstrained optimization problem with bounds on the design variables was formulated to attempt to recover the known joint parameters from the noisy synthetic marker trajectories. The cost function was

$$\min_{p} f(\mathbf{p}) \tag{19}$$

with

$$f(\mathbf{p}) = \sum_{k=1}^{50} \min_{q} \sum_{j=1}^{6} \sum_{i=1}^{3} (c_{ijk} - \hat{c}_{ijk}(\mathbf{p},\mathbf{q}))^2, \tag{20}$$

where $\mathbf{p}$ is a vector of 12 design variables containing the joint parameters, $\mathbf{q}$ is a vector of 27 generalized coordinates for the kinematic model, c_{ijk} is the ith coordinate of synthetic marker j at time frame k, and $\hat{c}_{ijk}(\mathbf{p},\mathbf{q})$ is the corresponding marker coordinate from the kinematic model. At each time frame, $\hat{c}_{ijk}(\mathbf{p},\mathbf{q})$ was computed from the current model parameters $\mathbf{p}$ and an optimized model configuration $\mathbf{q}$. A separate Levenberg–Marquardt nonlinear least-squares optimization was performed for each time frame in Eq. (20) to determine this optimal configuration. A relative convergence tolerance of 10^{-3} was chosen to achieve good accuracy with minimum computational cost. A nested optimization formulation [i.e., minimization occurs in Eqs. (19) and (20)] was used to decrease the dimensionality of the design space in Eq. (19). Equation (20) was coded in Matlab and exported as standalone C code using the Matlab Compiler (The Mathworks, Natick, MA). The executable read in a file containing the 12 design variables and output a file containing the resulting cost function value. This approach facilitated the use of different optimizers to solve Eq. (19).

To investigate the influence of design variable scaling on optimization algorithm performance, two versions of Eq. (20) were generated. The first used the original units of centimeters and radians for the position and orientation design variables, respectively. Bounds on the design variables were chosen to enclose a physically realistic region around the solution point in design space. Each position design variable was constrained to remain within a cube centered at the midpoint of the medial and lateral malleoli, where the length of each side was equal to the distance between the malleoli (i.e., 11.32 cm). Each orientation design variable was constrained to remain within a circular cone defined by varying its "true" value by ±30°. The second version normalized all 12 design variables to be within [−1,1] using

$$\mathbf{x}^{\text{norm}} = \frac{2\mathbf{x} - \mathbf{x}_{UB} - \mathbf{x}_{LB}}{\mathbf{x}_{UB} - \mathbf{x}_{LB}} \tag{21}$$

where *UB* and *LB* denote the upper and lower bounds, respectively, on the design variable vector [41].

Two approaches were used to compare PSO scale sensitivity to that of the other three algorithms. For the first approach, a fixed number of scaled and unscaled runs (10) was performed with each optimization algorithm starting from different random initial seeds, and the sensitivity of the final cost function value to algorithm choice and design variable scaling was evaluated. The stopping condition for PSO and GA runs was 10,000 function evaluations, while SQP and BFGS runs were terminated when a relative

Table 2 Fraction of successful optimizer runs for the analytical test problems. Top half: Results from the PSO, GA, SQP, and BFGS algorithms used in the present study. Bottom half: Results from the SA, GA, SQP, and DS algorithms used in Soest and Casius [5]. The GA and SQP algorithms used in that study were different from the ones used in our study. Successful runs were identified by a final cost function value within 10^{-3} of the known optimum value, consistent with [5].

				$\mathbf{H}_3$			
Study	Algorithm	H_1	H_2	(n=4)	(n=8)	(n=16)	(n=32)
Present	PSO	1.000	0.583	1.000	1.000	1.000	0.000
	GA	0.000	0.034	0.000	0.000	0.000	0.002
	SQP	0.00	0.11	0.00	0.00	0.00	0.00
	BFGS	0.00	0.32	0.00	0.00	0.00	0.00
Soest and Casius [5]	SA	1.000	0.027	0.000	0.001	0.000	0.000
	GA	0.990	0.999	1.000	1.000	1.000	1.000
	SQP	0.279	0.810	0.385	0.000	0.000	0.000
	DS	1.000	0.636	0.000	0.000	0.000	0.000

convergence tolerance of 10^{-5} or absolute convergence tolerance of 10^{-6} was met. For the second approach, a fixed number of function evaluations (10,000) were performed with each algorithm to investigate unscaled versus scaled convergence history. A single random initial guess was used for the PSO and GA algorithms, and each algorithm was terminated once it reached 10,000 function evaluations. Since individual SQP and BFGS runs require much fewer than 10,000 function evaluations, repeated runs were performed with different random initial guesses until the total number of function evaluations exceeded 10,000 at the termination of a run. This approach essentially uses SQP and BFGS as global optimizers, where the separate runs are like individual particles that cannot communicate with each another but have access to local gradient information. Finite difference step size tuning at the start of each run was included in the computation of number of function evaluations. Once the total number of runs required to reach 10,000 function evaluations was known, the lowest cost function value from all runs at each iteration was used to represent the cost over a range of function evaluations equal to the number of runs.

4 Results

For the analytical test problems, our PSO algorithm was more robust than our GA, SQP, and BFGS algorithms (Table 2, top half). PSO converged to the correct global solution 100% of the time on four of the six test problems (H_1 and H_3 with n=4, 8, and 16). It converged 58% of the time for problem H_2 and not at all for problem H_3 with n=32. In contrast, none of the other algorithms converged more than 32% of the time on any of the analytical test problems. Though our GA algorithm typically exhibited faster initial convergence than did our PSO algorithm (Fig. 2, left column), it leveled off and rarely reached the correct final point in design space within the specified number of function evaluations.

When PSO results were compared with previously published results for the same analytical test problems [5], PSO success rates were better than those of SA but worse than those of GA* (Table 2, bottom half). SA was successful 100% of the time only on problem H_1, while GA* was successful nearly 100% of the time on all six problems. Thus, for the global algorithms, GA* was the most robust overall, followed by PSO and then SA, with GA exhibiting the worst robustness. PSO converged more slowly than did GA* on all problems with available convergence plots (four of the six problems) and also more slowly than did SA on the one problem for which SA was successful (Fig. 2, right column).

For the biomechanical test problem, only the PSO algorithm was insensitive to design variable scaling, with the GA algorithm being only mildly sensitive. In ten out of ten trials, unscaled and scaled PSO runs converged to the same point in design space [Fig. 3(*a*)], while unscaled and scaled GA runs converged to nearly the same point [Fig. 3(*b*)]. PSO results were the most consistent from trial to trial, converging to a final cost function value between 69 and 71. (Table 3). GA results were the next most consistent with final cost function values ranging from 71 to 84. Typical unscaled and scaled PSO and GA runs produced root-mean-square (RMS) marker distance, position parameter, and orientation parameter errors of comparable magnitude, with PSO errors generally being slightly smaller.

In contrast, the SQP and BFGS algorithms were highly sensitive to design variable scaling in the biomechanical test problem. For the ten trials, unscaled and scaled SQP or BFGS runs rarely converged to similar points in design space (note the y axis scale in Fig. 3) and produced large differences in the final cost function value from one trial to the next [Figs. 3(*c*) and 3(*d*)]. Scaling improved the final result in seven out of ten SQP trials and in five of ten BFGS trials. The best unscaled and scaled SQP final cost function values were 255 and 121, respectively, while those of BFGS were 355 and 102 (Table 3). Thus, scaling yielded the best result found with both algorithms. The best SQP and BFGS trials generally produced larger RMS marker distance errors (up to two times worse), orientation parameter errors (up to five times worse), and position parameter errors (up to six times worse) than those found by PSO or GA.

When detailed convergence histories were plotted over 10,000 function evaluations for the biomechanical test problem (Fig. 4), unscaled and scaled histories for PSO were indistinguishable, while those of GA were similar and those of SQP or BFGS notably different. For PSO, only minute differences in the design variables on the order of 10^{-5} were observed, resulting from numerical round off errors caused by limitations in machine precision. To reach 10,000 function evaluations, 17 unscaled and 26 scaled SQP runs were required compared to 56 unscaled and 44 scaled runs for BFGS. The length and value of the initial flat region in the SQP and BFGS convergence histories was related to the number of FDSS tuning evaluations performed. More runs meant more tuning evaluations as well as an increased likelihood of finding a lower initial cost function value.

5 Discussion

In this paper we evaluated a recent variation of the PSO algorithm with dynamic inertia and velocity updating as a possible addition to the arsenal of methods that can be applied to difficult biomechanical optimization problems. For all problems investigated, our PSO algorithm with standard algorithm parameters performed better than did three off-the-self optimizers—GA, SQP,

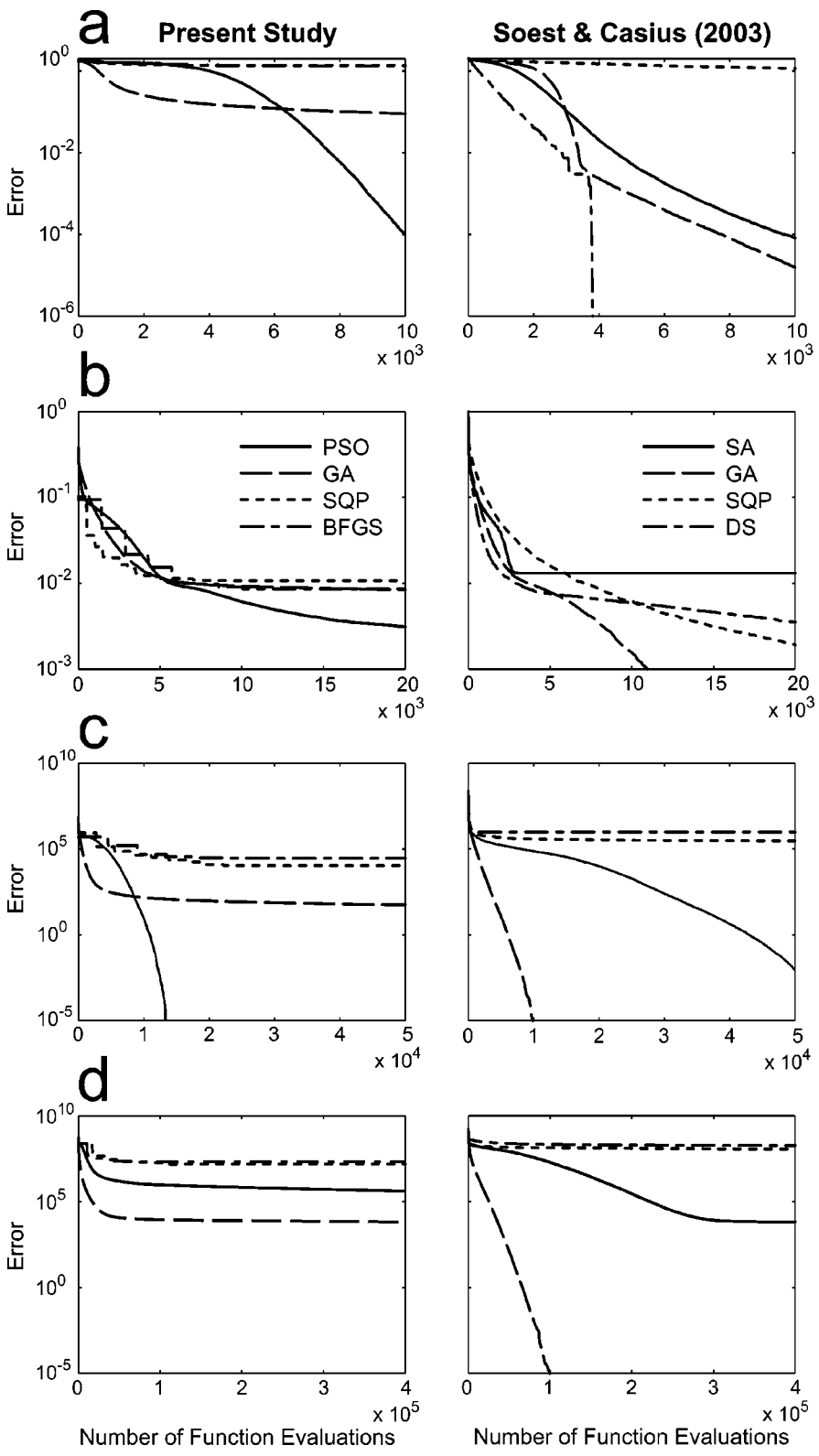

Fig. 2 A comparison of convergence history results for the analytical test problems. Left column: Results from the PSO, GA, SQP, and BFGS algorithms used in the present study. Right column: Results from the SA, GA, SQP, and DS algorithms used in Soest and Casius [5]. The GA and SQP algorithms used in that study were different from the ones used in our study. (*a*) Problem H_1. The SA results have been updated using corrected data provided by Soest and Casius, since the results in [5] accidentally used a temperature reduction rate of 0.5 rather than the standard value of 0.85 as reported. (*b*) Problem H_2. (*c*) Problem H_3 with $n=4$. (*d*) Problem H_3 with $n=32$. The error was computed using the known cost at the global optimum and represents the average of 1000 runs (100 multi-start SQP and BFGS runs in our study) with each algorithm.

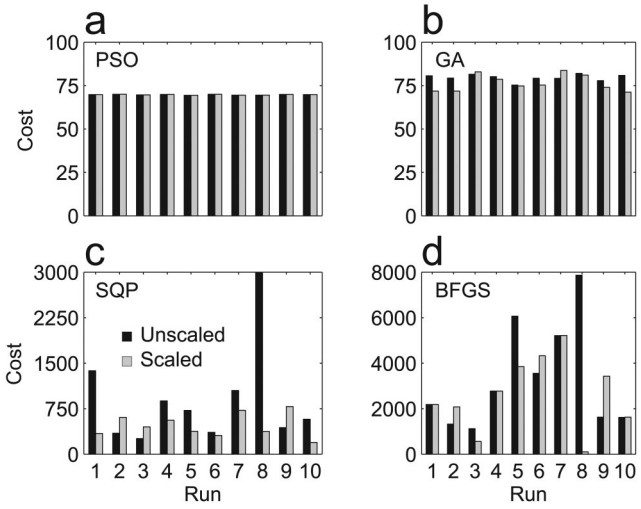

Fig. 3 Final cost function values for ten unscaled (dark bars) and scaled (gray bars) parallel PSO, GA, SQP, and BFGS runs for the biomechanical test problem. Each pair of unscaled and scaled runs was started from the same initial point(s) in design space, and each run was terminated when the specified stopping criteria was met (see the text).

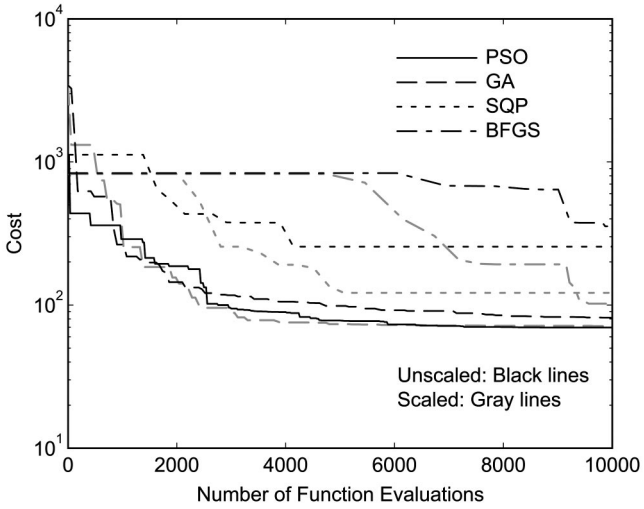

Fig. 4 Convergence history for unscaled (dark lines) and scaled (gray lines) parallel PSO, GA, SQP, and BFGS runs for the biomechanical test problem. Each algorithm run was terminated after 10,000 function evaluations. Only one unscaled and scaled PSO and GA run were required to reach 10,000 function evaluations, while repeated SQP and BFGS runs were required to reach that number. Separate SQP and BFGS runs were treated like individual particles in a single PSO run for calculating convergence history (see the text).

and BFGS. For the analytical test problems, PSO robustness was found to be better than that of two other global algorithms but worse than that of a third. For the biomechanical test problem with added numerical noise, PSO was found to be insensitive to design variable scaling while GA was only mildly sensitive and SQP and BFGS highly sensitive. Overall, the results suggest that our PSO algorithm is worth consideration for difficult biomechanical optimization problems, especially those for which design variable scaling may be an issue.

Though our biomechanical optimization involved a system identification problem, PSO may be equally applicable to problems involving forward dynamic, inverse dynamic, inverse static, or image matching analyses. Other global methods such as SA and GA have already been applied successfully to such problems [4,5,19], and there is no reason to believe that PSO would not perform equally well. As with any global optimizer, PSO utilization would be limited by the computational cost of function evaluations given the large number required for a global search.

Our particle swarm implementation may also be applicable to some large-scale biomechanical optimization problems. Outside the biomechanics arena [28,29,42–51], PSO has been used to solve problems on the order of 120 design variables [49–51]. In the present study, our PSO algorithm was unsuccessful on the largest test problem, H_3 with $n=32$ design variables. However, in a recent study, our PSO algorithm successfully solved the Griewank global test problem with 128 design variables using population sizes ranging from 16 to 128 [33]. When the Corana test problem (H_3) was attempted with 128 DVs, the algorithm exhibited worse convergence. Since the Griewank problem possesses a bumpy but continuous search space and the Corana problem a highly discrete search space, our PSO algorithm may work best on global problems with a continuous search space. It is not known how our PSO algorithm would perform on biomechanical problems with several hundred DVs, such as the forward dynamic optimizations of jumping and walking performed with parallel SQP in [1–3].

One advantage of global algorithms such as PSO, GA, and SA is that they often do not require significant algorithm parameter tuning to perform well on difficult problems. The GA used in [5] (which is not freely available) required no tuning to perform well on all of these particular analytical test problems. The SA algo-

Table 3 Final cost function values and associated marker distance and joint parameter root-mean-square (RMS) errors after 10,000 function evaluations performed by multiple unscaled and scaled PSO, GA, SQP, and BFGS runs. See Fig. 4 for the corresponding convergence histories.

			RMS error		
Optimizer	Formulation	Cost function	Marker distances (mm)	Orientation parameters (deg)	Position parameters (mm)
PSO	Unscaled	70.4	5.49	4.85	2.40
	Scaled	70.4	5.49	4.85	2.40
GA	Unscaled	77.9	5.78	2.65	6.97
	Scaled	74.0	5.64	3.76	4.01
SQP	Unscaled	255	10.4	3.76	14.3
	Scaled	121	7.21	3.02	9.43
BFGS	Unscaled	355	12.3	21.4	27.5
	Scaled	102	6.61	18.4	8.52

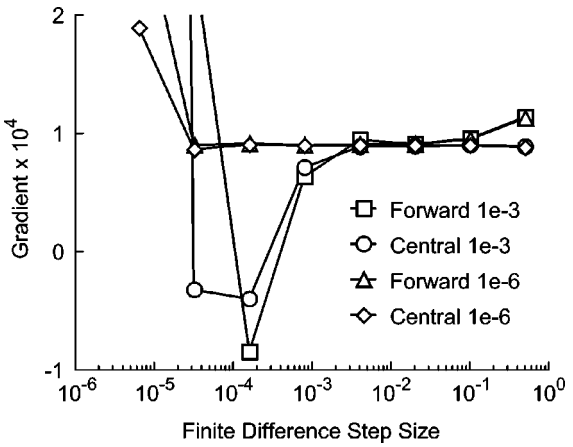

Fig. 5 Sensitivity of SQP and BFGS gradient calculations to the selected finite difference step size for one design variable. Forward and central differencing were evaluated using relative convergence tolerances of 10^{-3} and 10^{-6} for the nonlinear least squares suboptimizations performed during a cost function evaluation [see Eq. (20)].

rithm in [5] required tuning of two parameters to improve algorithm robustness significantly on those problems. Our PSO algorithm (which is freely available) required tuning of one parameter (w_o, which was increased from 1.0 to 1.5) to produce 100% success on the two problems where it had significant failures. For the biomechanical test problem, our PSO algorithm required no tuning, and only the population size of our GA algorithm required tuning to improve convergence speed. Neither algorithm was sensitive to the two sources of noise present in the problem—noise added to the synthetic marker trajectories, and noise due to a somewhat loose convergence tolerance in the Levenberg–Marquardt suboptimizations. Thus, for many global algorithm implementations, poor performance on a particular problem can be rectified by minor tuning of a small number of algorithm parameters.

In contrast, gradient-based algorithms such as SQP and BFGS can require a significant amount of tuning even to begin to approach global optimizer results on some problems. For the biomechanical test problem, our SQP and BFGS algorithms were highly tuned by scaling the design variables and determining the optimal FDSS for each design variable separately. FDSS tuning was especially critical due to the two sources of noise noted above. When forward and central difference gradient results were compared for one of the design variables using two different Levenberg–Marquardt relative convergence tolerances (10^{-3} and 10^{-6}), a "sweet spot" was found near a step size of 10^{-2} (Fig. 5). Outside of that "sweet spot," which was automatically identified and used in generating our SQP and BFGS results, forward and central difference gradient results diverged quickly when the looser tolerance was used. Since most users of gradient-based optimization algorithms do not scale the design variables or tune the FDSS for each design variable separately, and many do not perform multiple runs, our SQP and BFGS results for the biomechanical test problem represent best-case rather than typical results. For this particular problem, an off-the-shelf global algorithm such as PSO or GA is preferable due to the significant reduction in effort required to obtain repeatable and reliable solutions.

Another advantage of PSO and GA algorithms is the ease with which they can be parallelized [5,33] and their resulting high parallel efficiency. For our PSO algorithm, Schutte et al. [33] recently reported near ideal parallel efficiency for up to 32 processors. Soest and Casius [5] reported near ideal parallel efficiency for their GA algorithm with up to 40 processors. Though SA has historically been considered more difficult to parallelize [52], Higginson et al. [53] recently developed a new parallel SA implementation and demonstrated near ideal parallel efficiency for up to 32 processors. In contrast, Koh et al. [34] reported poor SQP parallel efficiency for up to 12 processors due to the sequential nature of the line search portion of the algorithm.

The caveat for these parallel efficiency results is that the time required per function evaluation was approximately constant and the computational nodes were homogeneous. As shown in [33], when function evaluations take different amounts of time, parallel efficiency of our PSO algorithm (and any other synchronous parallel algorithm, including GA, SA, SQP, and BFGS) will degrade with an increasing number of processors. Synchronization between individuals in the population or between individual gradient calculations requires slave computational nodes that have completed their function evaluations to sit idle until all nodes have returned their results to the master node. Consequently, the slowest computational node (whether loaded by other users, performing the slowest function evaluation, or possessing the slowest processor in a heterogeneous environment) will dictate the overall time for each parallel iteration. An asynchronous PSO implementation with load balancing, where the global best-found position is updated continuously as each particle completes a function evaluation, could address this limitation. However, the extent to which convergence characteristics and scale independence would be affected is not yet known.

To put the results of our study into the proper perspective, one must remember that optimization algorithm robustness can be influenced heavily by algorithm implementation details, and no single optimization algorithm will work for all problems. For two of the analytical test problems (H_2 and H_3 with $n=4$), other studies have reported PSO results using formulations that did not include dynamic inertia and velocity updating. Comparisons are difficult given differences in the maximum number of function evaluations and number of particles, but in general, algorithm modifications were (not surprisingly) found to influence algorithm convergence characteristics [54–56]. For our GA and SQP algorithms, results for the analytical test problems were very different from those obtained in [5] using different GA and SQP implementations. With seven mutation and four crossover operators, the GA algorithm used in [5] was obviously much more complex than the one used here. In contrast, both SQP algorithms were highly developed commercial implementations. Poor performance by a gradient-based algorithm can be difficult to correct even with design variable scaling and careful tuning of the FDSS. These findings indicate that specific algorithm implementations, rather than general classes of algorithms, must be evaluated to reach any conclusions about algorithm robustness and performance on a particular problem.

6 Conclusion

In summary, the PSO algorithm with dynamic inertia and velocity updating provides another option for difficult biomechanical optimization problems with the added benefit of being scale independent. There are few algorithm-specific parameters to adjust, and standard recommended settings work well for most problems [60,85]. The algorithm's main drawback is the high cost in terms of function evaluations because of slow convergence in the final stages of the optimization, a common trait among global search algorithms. In biomechanical optimization problems, noise, multiple local minima, and design variables of different scale can limit the reliability of gradient-based algorithms. The PSO algorithm presented here provides a simple-to-use off-the-shelf alternative for consideration in such cases.

Acknowledgment

This study was funded by NIH National Library of Medicine (R03 LM07332) and Whitaker Foundation grants to B. J. Fregly and an AFOSR (F49620-09-1-0070) grant to R. T. Haftka. The authors thank Dr. Knoek van Soest and Dr. Richard Casius for

providing the plot data in the right column of Fig. 2, Tushar Goel for providing the C source code for the GA algorithm, and Dr. Vladimir Balabanov of Vanderplaats R & D for assistance with VisualDOC modifications.

References

[1] Anderson, F. C., and Pandy, M. G., 1999, "A Dynamic Optimization Solution for Vertical Jumping in Three Dimensions," Comp. Meth. Biomech. Biomed. Eng., **2**, pp. 201–231.

[2] Anderson, F. C., and Pandy, M. G., 2001, "Dynamic Optimization of Human Walking," J. Biomech. Eng., **123**, pp. 381–390.

[3] Pandy, M. G., 2001, "Computer Modeling and Simulation of Human Movement," Annu. Rev. Biomed. Eng., **3**, pp. 245–273.

[4] Neptune, R. R., 1999, "Optimization Algorithm Performance in Determining Optimal Controls in Human Movement Analyses," J. Biomech. Eng., **121**, pp. 249–252.

[5] Soest, A. J. and Casius, L. J. R., 2003, "The Merits of a Parallel Genetic Algorithm in Solving Hard Optimization Problems," J. Biomech. Eng., **125**, pp. 141–146.

[6] Buchanan, T. S., and Shreeve, D. A., 1996, "An Evaluation of Optimization Techniques for the Prediction of Muscle Activation Patterns During Isometric Tasks," J. Biomech. Eng., **118**, pp. 565–574.

[7] Crowninshield, R. D., and Brand, R. D., 1981, "A Physiologically Based Criterion of Muscle Force Prediction in Locomotion," J. Biomech., **14**, pp. 793–801.

[8] Glitsch, U., and Baumann, W., 1997, "The Three-Dimensional Determination of Internal Loads in the Lower Extremity," J. Biomech., **11**, pp. 1123–1131.

[9] Kaufman, K. R., An, K.-N., Litchy, W. J., and Chao, E. Y. S., 1991, "Physiological Prediction of Muscle Forces—I. Theoretical Formulation," Neuroscience, **40**, pp. 781–792.

[10] Lu, T.-W. and O'Connor, J. J., 1999, "Bone Position Estimation from Skin Marker Coordinates using Global Optimisation with Joint Constraints," J. Biomech., **32**, pp. 129–124.

[11] Raasch, C. C., Zajac, F. E., Ma, B., and Levine, W. S., 1997, "Muscle Coordination of Maximum-Speed Pedaling," J. Biomech., **30**, pp. 595–602.

[12] Prilutsky, B. I., Herzog, W., and Allinger, T. L., 1997, "Forces of Individual Cat Ankle Extensor Muscles During Locomotion Predicted Using Static Optimization," J. Biomech., **30**, pp. 1025–1033.

[13] Bogert, A. J., Smith, G. D., and Nigg B. M., 1994, "In Vivo Determination of the Anatomical Axes of the Ankle Joint Complex: An Optimization Approach," J. Biomech., **12**, pp. 1477–1488.

[14] Mommerstag, T. J. A., Blankevoort, L., Huiskes, R., Kooloos, J. G. M., and Kauer, J. M. G., 1996, "Characterization of the Mechanical Behavior of Human Knee Ligaments: A Numerical–Experimental Approach," J. Biomech., **29**, pp. 151–160.

[15] Reinbolt, J. A., Schutte, J. F., Fregly, B. J., Haftka, R. T., George, A. D., and Mitchell, K. H., 2004, "Determination of Patient-Specific Multi-Joint Kinematic Models Through Two-Level Optimization," J. Biomech. **38**, pp. 621–626.

[16] Sommer, H. J., III, and Miller, N. R., 1980, "Technique For Kinematic Modeling of Anatomical Joints," J. Biomech. Eng., **102**, pp. 311–317.

[17] Vaughan, C. L., Andrews, J. G., and Hay, J. G., 1982, "Selection of Body Segment Parameters by Optimization Methods," J. Biomech. Eng., **104**, pp. 38–44.

[18] Kaptein, B. L., Valstar, E. R., Stoel, B. C., Rozing, P. M., and Reiber, J. H. C., 2003, "A New Model-Based RSA Method Validated Using CAD Models and Models from Reversed Engineering," J. Biomech., **36**, pp. 873–882.

[19] Mahfouz, M. R., Hoff, W. A., Komistek, R. D., and Dennis, D. A., 2003, "A Robust Method for Registration of Three-Dimensional Knee Implant Models to Two-Dimensional Fluoroscopy Images," IEEE Trans. Med. Imaging, **22**, pp. 1561–1574.

[20] You, B.-M., Siy, P., Anderst, W., and Tashman, S., 2001, "In Vivo Measurement of 3-D Skeletal Kinematics from Sequences of Biplane Radiographs: Application to Knee Kinematics," IEEE Trans. Med. Imaging, **20**, pp. 514–525.

[21] Gill, P. E., Murray, W., and Wright, M. H., 1986, *Practical Optimization*, Academic Press, New York.

[22] Wilcox, K. and Wakayama, S., 2003, Simultaneous Optimization of a Multiple-Aircraft Family, J. Aircr., **40**, pp. 616–622.

[23] Kennedy J., and Eberhart R. C., 1995, "Particle Swarm Optimization," *Proceedings of the IEEE International Conference on Neural Networks*, Perth, Australia Vol. 4, pp. 1942–1948.

[24] Groenwold, A. A., and Fourie, P. C., 2002, "The Particle Swarm Optimization in Size and Shape Optimization," Struct. Multidiscip. Optim., **23**, pp. 259–267.

[25] Shi, Y., and Eberhart, R. C., 1998, "Parameter Selection in Particle Swarm Optimization," *Lect. Notes Comput. Sc. 1447*, Springer-Verlag, Berlin, pp. 591–600.

[26] Fourie, P. C. and Groenwold, A. A., 2001, "The Particle Swarm Algorithm in Topology Optimization," *Proceedings of the 4th World Congress of Struct. Multidiscip. Opt.*, Dalian, China pp. 52–53.

[27] Schutte, J. F., 2001, "Particle Swarms in Sizing and Global Optimization," Master's thesis, University of Pretoria, South Africa.

[28] Schutte, J. F., and Groenwold, A. A., 2003, "Sizing Design of Truss Structures Using Particle Swarms," Struct. Multidiscip. Optim., **25**, pp. 261–269.

[29] Schutte, J. F., and Groenwold, A. A., 2004, "A Study of Global Optimization Using Particle Swarms," J. Global Opt. (in press).

[30] Deb, K., 2001, *Multi-Objective Optimization Using Evolutionary Algorithms*, Wiley Interscience Series in Systems and Optimization, Wiley, New York, Chap. 4.

[31] Deb, K., and Agrawal, R. B., 1995, "Simulated Binary Crossover for Continuous Search Space," Complex Syst., **9**, pp. 115–148.

[32] Deb, K., and Goyal, M., 1996, "A Combined Genetic Adaptive Search (GeneAS) for Engineering Design," Comput. Sci. Inform., **26**, pp. 30–45.

[33] Schutte, J. F., Reinbolt, J. A., Fregly, B. J., Haftka, R. T., and George, A. D., 2004, "Parallel Global Optimization with the Particle Swarm Algorithm," Int. J. Numer. Methods Eng. **61**, pp. 2296–2315.

[34] Koh, B. I., Reinbolt, J. A., Fregly, B. J., and George, A. D., 2004, "Evaluation of Parallel Decomposition Methods for Biomechanical Optimizations," Comp. Meth. Biomech. Biomed. Eng. **7**, pp. 215–225.

[35] Gropp, W., and Lusk, E., 1996, "User's Guide for MPICH, "A Portable Implementation of MPI," Argonne National Laboratory, Mathematics and Computer Science Division, http://www.mcs.anl.gov/mpi/mpiuserguide/paper.html.

[36] Gropp, W., Lusk, E., Doss, N., and Skjellum, A., 1996. "A High Performance, Portable Implementation of the MPI Message Passing Interface Standard," Parallel Comput., **22**, pp. 789–828.

[37] Schaffer, J. D., Caruana, R. A., Eshelman, L. J., and Das, R., 1989, "A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimizing," *Proceedings of the 3rd International Conference on Genetic Algebra*, edited by J. D., David, Morgan Kaufmann Publishers, San Mateo, California, pp. 51–60.

[38] Corana, A., Marchesi, M., Martini, C., and Ridella, S., 1987, "Minimizing Multimodal Functions of Continuous Variables with the "Simulated Annealing" Algorithm," ACM Trans. Math. Softw., **13**, pp. 262–280.

[39] Chèze, L., Fregly, B. J., and Dimnet, J. 1995, "A Solidification Procedure to Facilitate Kinematic Analyses Based on Video System Data," J. Biomech., **28**, pp. 879–884.

[40] Lu, T.-W. and O'Connor, J. J., 1999, "Bone Position Estimation from Skin Marker Co-ordinates using Global Optimization with Joint Constraints," J. Biomech., **32**, pp. 129–124.

[41] *Reference Manual for VisualDOC* C/C++ *API*, 2001, Vanderplaats Research and Development, Inc., Colorado Springs, CO.

[42] Boeringer, D. W., and Werner, D. H., 2004, "Particle Swarm Optimization Versus Genetic Algorithms For Phased Array Synthesis," IEEE Trans. Antennas Propag., **52**, pp. 771–779.

[43] Brandstatter, B., and Baumgartner, U., 2002, "Particle Swarm Optimization—Mass-Spring System Analogon," IEEE Trans. Magn., **38**, pp. 997–1000.

[44] Cockshott, A. R., and Hartman, B. E., 2001, "Improving the Fermentation Medium for Echinocandin B Production Part II: Particle Swarm Optimization," Process Biochem. (Oxford, U.K.), **36**, pp. 661–669.

[45] Costa, E. F. J., Lage, P. L. C., and Biscaia, E. C., Jr., 2003, "On the Numerical Solution and Optimization of Sstyrene Polymerization in Tubular Reactors," Comput. Chem. Eng., **27**, pp. 1591–1604.

[46] Lu, W. Z., Fan, H.-Y., and Lo, S. M., 2003, "Application of Evolutionary Neural Network Method in Predicting Pollutant Levels in Downtown Area of Hong Kong," Neurocomputing, **51**, pp. 387–400.

[47] Pidaparti, R. M., and Jayanti, S., 2003, "Corrosion Fatigue Through Particle Swarm Optimization," AIAA J., **41**, pp. 1167–1171.

[48] Tandon, V., El-Mounayri, H., and Kishawy, H., 2002, "NC End Milling Optimization Using Evolutionary Computation," Int. J. Mach. Tools Manuf., **42**, pp. 595–605.

[49] Abido, M. A., 2002, "Optimal Power Flow Using Particle Swarm Optimization," Int. J. Electr. Power Energy Syst., **24**, pp. 563–571.

[50] Abido, M. A., 2002, "Optimal Design of Power System Stabilizers using Particle Swarm Optimization," IEEE Trans. Energy Convers., **17**, pp. 406–413.

[51] Gies, D., and Rahmat-Samii, Y., 2003, "Particle Swarm Optimization for Reconfigurable Phase-Differentiated Array Design," Microwave Opt. Technol. Lett., **38**, pp. 168–175.

[52] Leite, J. P. B. and Topping, B. H. V., 1999, "Parallel Simulated Annealing for Structural Optimization," Compos. Struct., **73**, pp. 545–564.

[53] Higginson, J. S., Neptune, R. R., and Anderson, F. C., 2004, "Simulated Parallel Annealing Within a Neighborhood for Optimization of Biomechanical Systems," J. Biomech. (in press).

[54] Carlisle, A., and Dozier, G., 2001, "An Off-the-Shelf PSO," *Proceedings of the Workshop on Particle Swarm Optimization*, Indianapolis, IN.

[55] Parsopoulos, K. E., and Vrahatis, M. N., 2002, "Recent Approaches to Global Optimization Problems through Particle Swarm Optimization." Nat. Comp., **1**, pp. 235–306.

[56] Trelea, I. C., 2002, "The Particle Swarm Optimization Algorithm: Convergence Analysis and Parameter Selection," Inform. Process. Lett., **85**, pp. 317–325.