Swarm and Evolutionary Computation 18 (2014) 1–10

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo

Regular Paper

Using animal instincts to design efficient biomedical studies via particle swarm optimization

Jiaheng Qiu [a], Ray-Bing Chen [b], Weichung Wang [c], Weng Kee Wong [a,*]

[a] Department of Biostatistics, University of California, Los Angeles, CA 90095, USA
[b] Department of Statistics, National Cheng-Kung University, Tainan 70101, Taiwan
[c] Institute of Applied Mathematical Sciences, National Taiwan University, 10617, Taipei, Taiwan

ARTICLE INFO

Article history:
Received 14 June 2013
Received in revised form
9 May 2014
Accepted 27 June 2014
Available online 15 July 2014

Keywords:
Approximate design
c-optimal design
D-optimal design
Efficiency
Metaheuristic algorithms
Particle swarm optimization

ABSTRACT

Particle swarm optimization (PSO) is an increasingly popular metaheuristic algorithm for solving complex optimization problems. Its popularity is due to its repeated successes in finding an optimum or a near optimal solution for problems in many applied disciplines. The algorithm makes no assumption of the function to be optimized and for biomedical experiments like those presented here, PSO typically finds the optimal solutions in a few seconds of CPU time on a garden-variety laptop. We apply PSO to find various types of optimal designs for several problems in the biological sciences and compare PSO performance relative to the differential evolution algorithm, another popular metaheuristic algorithm in the engineering literature.

1. Introduction

Optimal experimental designs have been gaining attention in the last two decades [1]. A main reason is rising cost in conducting experiments and an increasing realization in more applied fields that optimal design ideas can save costs substantially without sacrifice in statistical efficiency. Some examples are given in [2–6], where the problems include designing reaction kinetics studies to medical studies with a time-to-event outcome. Berger and Wong [7] describes a collection of concrete applications of optimal designs to real problems that ranges from applications in biomedical and social science arenas, including a design problem to identify optimal locations to monitor groundwater wells in the Los Angeles basin.

Nonlinear models are frequently used to study outcomes or responses in biomedical experiments. This means that we assume a known nonlinear functional relationship between the mean response and the independent variables. This function has unknown parameters that determine the shape and properties of the mean response and one common goal in the study is to estimate the parameters in the mean function. In addition, the model also has an unobservable error term with mean zero and constant variance. Given a study objective, the design problem involves selecting the right number of combination levels of the independent variables to observe the outcome and what these levels are. The design optimality criterion for nonlinear models depends on the values of the model parameters and nominal values (or best guesses for these parameters) are required before the optimal design can be implemented. Because the optimal designs depend on the nominal values, they are termed locally optimal. Such optimal designs usually represent the fist step in finding an optimal design strategy and is the simplest to construct and study.

The analytical description of the locally optimal design for a nonlinear model is rarely available unless the model is very simple. When they do exist, they are usually complicated; see for example, the analytical description for the locally *D*-optimal design for estimating the two parameters in the logistic model [8]. Further, the formula or analytical description of the optimal design in a nonlinear model is invariably derived under a set of mathematical assumptions that may or may not apply in practice. For these reasons, it is desirable to have a flexible and effective algorithm that can find a variety of optimal designs quickly and reliably.

There are algorithms for finding optimal designs and most are based on heuristics or intuition and they do not have a theoretical basis. Only a couple of algorithms can be proven to converge to the optimal designs and prominent ones include Fedorov's and Wynn's algorithms for generating *D* and *c*-optimal designs [9,10]. The former designs are useful for estimating all parameters in the mean function and the latter targets estimation of a specific function of the model parameters by minimizing, respectively,

* Corresponding author.

http://dx.doi.org/10.1016/j.swevo.2014.06.003
2210-6502/

the volume of the confidence ellipsoid and the asymptotic variance of the estimated function of interest. For the few algorithms that can be shown to converge mathematically, problems may still exist and they include (i) they take too long to converge, (ii) they may fail to converge for more complicated setups that they are not designed, such as for nonlinear for mixed effects nonlinear models, and (iii) numerical issues due to rounding problems or the intrinsic nature of the sequential process; for example, many algorithms produce clusters of support points as the algorithm proceeds and these clusters require periodic and judicious collapsing into the correct distinct but unknown support points.

In the next section, we briefly review particle swarm optimization (PSO) methodology and show that it is an exciting, easy and effective algorithm to generate optimal designs for statistical models. This algorithm has been used for almost a dozen of years in the computer science and engineering circles, and increasingly more so in recent years due to its repeated successes in solving increasingly large class of applied problems. The main reasons for its popularity seem to be its flexibility, ease of implementation and utility, and general applicability to solve (or nearly solve) complex optimization problems without having to make specific assumptions on the objective function. In Section 3 we present the statistical background and demonstrate PSO can efficiently find a variety of optimal designs for different types of nonlinear models in biomedicine. Section 4 provides a discussion and also compares PSO performance relative to the differential evolution algorithm, which is another popular metaheuristic algorithm for solving optimization problems in engineering problems. Section 5 is the conclusion.

2. Particle swarm optimization (PSO)

Nature-inspired algorithms have been gaining popularity and dominance in the last decade both in academia and industrial applications after adjusting for different types of biases [11,12]. One of the most prominent examples of a nature-inspired algorithm is Particle Swarm Optimization (PSO) based on swarm intelligence. It is a metaheuristic algorithm and comes about from the research in fish and swarm movement behavior. PSO is intriguing in that they always seem to be able to quickly solve the optimization problem or provide good quality solutions for many types of complex optimization problems, even though the method itself lacks a firm theoretical justification to date. An attempt to explain its success from a biological viewpoint is given in Garnier et al. [13] and an overview of PSO is available in Poli et al. [14]. Common characteristics of PSO includes its ability to find the optimal solution to a complex problem or gets close to the optimal solution quickly without requiring any assumption on the function to be optimized. PSO codes are available on many websites such as http://www.swarmintelligence.org, http://particleswarm.info with the latter website more updated and consistently so. In addition, codes are available in books on metaheuristic methods like Yang [15] and also in software such as MATLAB where several PSO codes can be found at the Programs section at http://www.mathworks.com/matlabcentral/fileexchange/7506.

There are two basic equations that drive movement for the each particle in the PSO algorithm in its search to optimize an objective function $h(\cdot)$. At times t and $t+1$, the movement of particle i is governed by

$$\mathbf{v}_i^{t+1} = \tau_t \mathbf{v}_i^t + \gamma_1 \boldsymbol{\beta}_1 \odot (\mathbf{p}_i - \mathbf{x}_i^t) + \gamma_2 \boldsymbol{\beta}_2 \odot (\mathbf{p}_g - \mathbf{x}_i^t), \tag{1}$$

and

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1}. \tag{2}$$

Here, $\mathbf{v}_i^t$ is the particle velocity at time t and $\mathbf{x}_i^t$ is the current particle position at time t. The inertia weight τ_t adjusts the influence of the former velocity and can be a constant or a decreasing function with values between 0 and 1. For example, a linearly decreasing function over the specified time range with initial value 0.9 and end value 0.4 [16]. Further, the vector $\mathbf{p}_i$ is the personal best (optimal) position as perceived by the ith particle and the vector $\mathbf{p}_g$ is the global best (optimal) position as perceived by all particles, up to time t. This means that up to time t, the personal best for particle i is $pbest_i = h(\mathbf{p}_i)$ and $gbest = h(\mathbf{p}_g)$. The two random vectors in the PSO algorithm are $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$ and their components are usually taken to be independent random variables from $U(0,1)$. The constant γ_1 is the cognitive learning factor and γ_2 is the social learning factor. These two constants determine how each particle moves toward its own personal best position or overall global best position. For our design problems, we set $\gamma_1 = \gamma_2 = 2$ and they all worked well for the problems we discussed including almost all other design problems that we have investigated this far. In Eq. (1), the product in the last two terms is Hadamard product.

The pseudo code for the PSO procedure for a flock of size n is as follows.

Algorithm 1.

(1) Initialize particles
  (1.1) Initiate positions $\mathbf{x}_i$ and velocities $\mathbf{v}_i$ for $i=1,\ldots,n$.
  (1.2) Calculate the fitness values $h(\mathbf{x}_i)$ for $i=1,\ldots,n$.
  (1.3) Determine the personal best positions $\mathbf{p}_i = \mathbf{x}_i$ and the global position $\mathbf{p}_g$.
(2) Repeat until stopping criteria are satisfied.
  (2.1) Calculate particle velocity according to Eq. (1).
  (2.2) Update particle position according to Eq. (2).
  (2.3) Calculate the fitness values $h(\mathbf{x}_i)$.
  (2.4) Update personal and global best positions $\mathbf{p}_i$ and $\mathbf{p}_g$.
(3) Output $\mathbf{p}_g = \arg\min h(\mathbf{x})$ with $gbest = h(\mathbf{p}_g)$.

The initial velocity for each particle may be set equal to 0 or be randomly assigned from $U(0,1)$. A key difference in our PSO version is that our version requires that the weights in designs are positive and they sum to unity and we pull back particles that wander outside of the design space to the boundary. The rationale for having this feature is that many D-optimal designs have support points at the boundary of the design space, see [17] for example. Without this feature that we have introduced into the PSO algorithm, our experience is that PSO becomes either very slow or it fails to find the optimal design, especially in high dimensional problems.

3. Generating optimal designs for biomedical studies using PSO

In this section, we discuss the statistical background and apply PSO to find various types of optimal designs for common models in the biomedical studies. These models may appear small in terms of the number of parameters that they have but as noted in Konstantinou et al. [18], finding optimal designs for such models can still be problematic using traditional numerical methods or analytically.

Here and throughout, our focus is approximate designs, which are probability measures defined on the given design space X [19]. This means that once we are given a pre-determined sample size n, a given model and a design criterion (or objective function) $h(\cdot)$, our optimization problem is to find the number (k) of design points (or support points) required, the locations of the design points $x_1, \ldots, x_k$ in X and the weight distribution $w_1, \ldots, w_k$ at these points to optimize the given criterion $h(\cdot)$. These weights naturally