J Glob Optim (2013) 56:61–91
DOI 10.1007/s10898-012-9866-7

New horizons in sphere-packing theory, part II: lattice-based derivative-free optimization via global surrogates

Paul Belitz · Thomas Bewley

Received: 28 April 2011 / Accepted: 27 January 2012 / Published online: 24 March 2012

Abstract Derivative-free algorithms are frequently required for the optimization of non-smooth scalar functions in n dimensions resulting, for example, from physical experiments or from the statistical averaging of numerical simulations of chaotic systems such as turbulent flows. The core idea of all efficient algorithms for problems of this type is to keep function evaluations far apart until convergence is approached. *Generalized pattern search* (*GPS*) algorithms, a modern class of methods particularly well suited to such problems, accomplish this by coordinating the search with an underlying grid which is refined, and coarsened, as appropriate. One of the most efficient subclasses of GPS algorithms, known as the *surrogate management framework* (*SMF*; see Booker et al. in Struct Multidiscip Optim 17:1–13, 1999), alternates between an exploratory *search* over an interpolating function which summarizes the trends exhibited by existing function evaluations, and an exhaustive *poll* which checks the function on neighboring points to confirm or confute the local optimality of any given *candidate minimum point* (*CMP*) on the underlying grid. The original SMF algorithm implemented a GPS step on an underlying Cartesian grid, augmented with a Kriging-based surrogate search. Rather than using the n-dimensional Cartesian grid (the typical choice), the present work introduces for this purpose the use of lattices derived from n-dimensional sphere packings. As reviewed and analyzed extensively in Part I of this series (see Belitz, PhD dissertation, University of California, San Diego, 2011, Chap. 2), such lattices are significantly more uniform and have many more nearest neighbors than their Cartesian counterparts. Both of these facts make them far better suited for coordinating GPS algorithms, as demonstrated here in a variety of numerical tests.

Keywords N-dimensional sphere-packing theory · Generalized pattern search · Derivative-free optimization · Global optimization · Surrogate-based methods · Kriging interpolation

P. Belitz · T. Bewley (✉)
Flow Control & Coordinated Robotics Labs, UC San Diego, La Jolla, CA, USA
e-mail: bewley@ucsd.edu

P. Belitz
e-mail: pbelitz@ucsd.edu

1 Introduction

The minimization of computationally expensive, high-dimensional functions is often most efficiently performed via gradient-based optimization algorithms such as nonlinear conjugate gradients and L-BFGS-B. In complex systems for which an accurate computer model is available, the gradient required by such algorithms may often be found via adjoint analysis. However, when the function in question is not sufficiently smooth to leverage gradient information effectively during its optimization (see, e.g., Fig. 1), a derivative-free approach is necessary. Such a scenario is evident, for example, when optimizing a finite-time-average approximation of an infinite-time-average statistic of a chaotic system such as a turbulent flow. Such an approximation may be determined via simulation or experiment. The truncation of the averaging window used to determine this approximation renders derivative-based optimization strategies ill suited, as the truncation error, though small, is effectively decorrelated from one flow simulation/experiment to the next. This effective decorrelation of the truncation error is reflected by the exponential growth, over the entire finite time horizon considered, of the adjoint field related to the optimization problem of interest in the simulation-based setting.

Due to the often significant expense associated with performing repeated function evaluations (in the above example, turbulent flow simulations or experiments), a derivative-free optimization algorithm which converges to within an accurate tolerance of the global minimum of a nonconvex function of interest with a minimum number of function evaluations is desired. It is noted that, in the general case, proof of convergence of an optimization algorithm to a global minimum is possible only when, in the limit of a large number of function evaluations N, the function evaluations become dense in the feasible region of parameter space (Torn and Zilinskas 1987). Though the algorithm developed in the present work, when implemented properly, satisfies this condition, so do far inferior approaches, such as a rather unintelligent algorithm which we call exhaustive sampling (ES), which simply covers the feasible parameter space with a grid, evaluates the function at *every* gridpoint, refines the grid

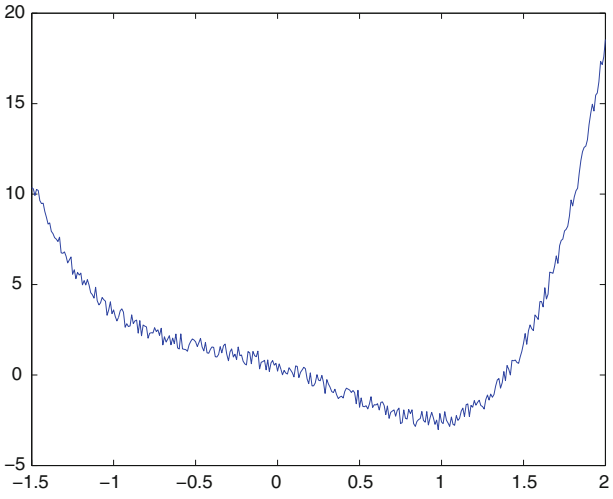

Fig. 1 Prototypical nonsmooth optimization problem for which local gradient information is ill suited to accelerate the optimization algorithm

by a factor of two, and repeats until terminated. Thus, a guarantee of global convergence is not sufficient to establish the *efficiency* of an optimization algorithm. If function evaluations are relatively expensive, and thus only a relatively small number of function evaluations can ultimately be afforded, effective heuristics for rapid convergence are perhaps even more important than rigorous proofs of the behavior of the optimization algorithm in the limit of large N, a limit that might actually be argued to be of limited relevance when function evaluations are expensive. Given that such algorithms are often used in applications in which only a few hundred function evaluations can be afforded, careful attention to such heuristics forms an important foundation for the present study.

One of the earliest derivative-free optimization approaches to appear in the literature is the *downhill simplex method* (see Spendley et al. 1962 and Nelder and Mead 1965). The downhill simplex method is inherently based on an iterative, amoeba-like evolution (moving one point at a time) of a set of $n + 1$ points in n dimensions towards the minimum of a (possibly, nonsmooth) function. A large body of literature appeared after the original introduction of this method, much of which was aimed at heuristic strategies designed to keep the evolving simplex as regular as possible as the iteration proceeds, while expanding or contracting as appropriate (see Nelder and Mead 1965 and the references contained therein). Other derivative-free optimization methods that do not utilize an underlying grid are available; notable examples include Simulated Annealing (Kirkpatrick et al. 1984), Stochastic Optimization methods (Ermoliev 1988), Particle Swarm algorithms (Kennedy and Eberhart 1995), Genetic Algorithms (Goldberg 1989), and similar. Another class of methods which can be formulated as optimization algorithms include space-filling methods such as Latin Hypercube Sampling (see McKay et al. 1979). These methods can be very useful for selecting points of interest in a space where absolutely no information is available; however, as optimization algorithm, their performance is substantially lower than that of the methods presented and discussed in this work. The grid-based methods considered in the present work are structurally and theoretically fundamentally different, so we will not dwell on such grid-free methods in this introduction.

However, it is worth noting the inherent dependence on the regularity of an evolving simplex (defined by $n + 1$ vertices in n dimensions) in this classical method, and the analogous focus in the present work on the identification and characterization of a maximally uniform *minimal positive basis*, also consisting of $n + 1$ points. The role of the simplex in both cases is essentially identical: to identify the best direction to move next using a minimum number of new function evaluations.

If, for the moment, we give up on the goal of global convergence, the perhaps simplest grid-based derivative-free optimization algorithm, which we call successive polling (SP), proceeds as follows:

- Start with a coarse grid and evaluate the function at some starting point on this grid, identified as the first candidate minimum point (CMP).
- Then, poll (that is, evaluate) the function values on gridpoints which neighbor the CMP in parameter space, at a sufficient number of gridpoints to *positively span*[1] the feasible neighborhood of the CMP (this step ensures convergence, as discussed further in Torczon 1997; Booker et al. 1999; Coope and Price 2001). When polling:

  (a) If any poll point is found to have a function value less than that of the CMP, immediately consider this new point the new CMP and terminate the present poll step.

[1] That is, such that any feasible point in the neighborhood of the CMP can be reached via a *linear combination with non-negative coefficients* of the vectors from the CMP to the poll points.

(b) If no poll points are found to have function values less than that of the CMP, refine the grid by a factor of two.

- Initiate a new poll step, either (a) around the new CMP or (b) around the old CMP on the refined grid, and repeat until terminated.

Though the basic SP algorithm described above, on its own, is not very efficient, there are a variety of effective techniques for accelerating it. All grid-based schemes which effectively build on this basic SP idea are classified as generalized pattern search (GPS) algorithms.

The most efficient subclass of GPS algorithms, known as the surrogate management framework (SMF; see Booker et al. 1999), leverages inexpensive interpolating "surrogate" functions (often, Kriging interpolations are used) to summarize the trends of the existing function evaluations, and to provide suggested new regions of parameter space in which to perform one or more additional function evaluation(s) between each poll step. SMF algorithms thus alternate beween two steps:

(i) *Search* over the inexpensive interpolating function to identify, based on the existing function evaluations, the most promising gridpoint at which to perform a new function evaluation. Perform a function evaluation at this point, update the interpolating function, and repeat. The search step may be terminated either when it returns a gridpoint at which the function has already been evaluated, or when the function, once evaluated, has a value greater than that of the CMP.

(ii) *Poll* the neighborhood of the new CMP identified by the search algorithm, following rules (a) and (b) above.

There is substantial flexibility during the search step described above. An effective search is essential for an efficient SMF algorithm. In the case that the search behaves poorly and fails to return improved function values, the SMF algorithm essentially reduces to the SP algorithm. If, however, the surrogate-based search is effective, the SMF algorithm will converge to a minimum far faster than a simple SP-based minimization. As the search and poll steps are essentially independent of each other, we will discuss them each in turn in the sections that follow, then present how we have combined them.

Note that historically the Poll step of the SMF algorithm has been designed to be as computationally inexpensive as possible (Booker et al. 1999; Marsden et al. 2007; Yang et al. 2010), which led directly to the use of a minimal positive basis (that is, a basis utilizing $n + 1$ points), rather than, e.g. a maximal positive basis (that is, $2n$ points, historically the Cartesian identity). The fundamental motivation is to allow the Search step to provide efficiency in the optimization, using the Poll step as infrequently as possible while ensuring good local convergence properties. The subject of maximally efficient locally convergent SP-like algorithms is the subject of a different discussion (see Belitz 2011, Chapter 4, for in-depth discussion of the application of efficient lattices to a highly sophisticated class of local derivative-free optimization methods).

Note that if the search produces a new CMP which is several gridpoints away from the previous function evaluations, which occasionally happens when exploring functions with multiple minima, the grid may be *coarsened* appropriately in order to explore the vicinity of this new CMP efficiently (that is, with a coarse grid first, then refined as necessary). Note also that the interpolating surrogate function of the SMF may be used to *order* the function evaluations of the poll step, such that those poll points which are most likely to have a function value lower than that of the CMP are evaluated first. By so doing, the poll steps will, on average, terminate sooner, and the computational cost of the overall algorithm may be reduced further.

Table 1 Characteristics of select distinct lattices in dimensions 2, 3, and 8, ordered from dense to rare (for a more complete characterization, see Tables 2 and 3 of Part I)

n	Lattice	Name	Δ	Θ	G	τ
2	A_2	Hexagonal	0.90690	1.2092	0.080188	6
	$\mathbb{Z}^2$	Square	0.78540	1.5708	0.083333	4
3	A_3	Face-centered cubic (FCC)	0.74048	2.0944	0.078745	12
	$\mathbb{Z}^3$	Cubic	0.52360	2.7207	0.083333	6
8	E_8	Gosset	0.25367	4.0587	0.071682	240
	D_8		0.12683	32.470	0.075914	112
	A_8	Zero-sum	0.08456	32.993	0.077391	72
	$\mathbb{Z}^8$	Cartesian	0.01585	64.939	0.083333	16

Listed (see Part I) are the packing density, Δ, covering thickness, Θ, mean squared quantization error per dimension, G, and kissing number, τ. Note that $\mathbb{Z}^n$ is significantly outperformed in every standard metric in every dimension $n > 1$ by the available alternatives

Table 2 The densest, most uniform lattices available in several dimensions, and two factors quantifying the degree to which these lattices are better than the corresponding Cartesian grid in the same dimension; f_Δ denotes the factor of improvement in the packing density, an indication of the uniformity of the lattice, and f_τ denotes the factor of improvement in the kissing number, an indication of the flexibility available in selecting a positive basis from the nearest neighbors on the lattice

	A_2	A_3	D_4	D_5	E_6	E_7	E_8	K_{12}	Λ_{16}	Λ_{24}
f_Δ	1.155	1.414	2	2.83	4.62	8	16	152	4096	1.68×10^7
f_τ	1.5	2	3	4	6	9	15	31.5	135	4095

Note that the improvements becoming especially pronounced as the dimension n is increased

To the best of our knowledge, all previous GPS and SMF implementations have been coordinated using Cartesian grids (see Booker et al. 1999; Marsden et al. 2007; Audet and Dennis 2006; Yang et al. 2010; Coope and Price 2001). A primary goal of the present work is to demonstrate convincingly that significant performance gains may be realized simply by eschewing this dominant Cartesian paradigm. Like in the game of checkers (contrast "American" checkers with "Chinese" checkers), Cartesian grids are not the only choice for discretizing parameter space. Other structured choices arising from n-dimensional sphere packing theory (see Tables 1 and 2, and further discussion in the precursor to this manuscript, New Horizons in Sphere Packing Theory, Part I, Belitz & Cessna, hereafter referred to as 'Part I') are significantly more uniform and have many more nearest neighbors, especially as the dimension of the problem in question is increased; both of these properties suit these alternative lattices well for coordinating grid-based optimization algorithms.

The definitive comprehensive reference on the subject of n-dimensional sphere packing theory is Conway and Sloane (1998).[2] Part I (Belitz & Cessna) of this study contains a concise summary of this involved subject, describing essentially everything one needs to know about lattices up to dimension $n = 24$ in order to use them effectively in practical engineering applications. For simplicity, the present investigation focuses on the use of just three such

[2] In fact, as pointed out in Part I, Conway and Sloane (1998, p. 12) state: "A related application that has not yet received much attention is the use of these packings for solving n-dimensional search or approximation problems"; this is exactly the focus of the present paper.

lattices, the zero-sum lattice A_n, which is an n-dimensional analog of the 2-dimensional hexagonal lattice and the 3-dimensional face-centered-cubic lattice, then D_n lattice, which is an n-dimensional analog to the $n = 3$ BCC lattice, and the Gosset lattice E_8, which is an 8-dimensional analog of the 3-dimensional diamond packing, and is particularly uniform. The definitions and algorithms for enumerating nearest neighbor points and quantization to the lattice are described completely in Part I. For the sake of completeness, some material from Part I is reiterated below.

An n-dimensional infinite sphere packing is an array of nodal points in R_n obtained via the packing of identical n-dimensional spheres. By packing, we mean an equilibrium configuration of spheres, each with at least 2 nearest neighbors, against which a repellant force is applied. An n-dimensional real lattice (a.k.a. lattice packing) is a sphere packing which is shift invariant (that is, which looks identical upon shifting any nodal point to the origin).

The characteristics of such sphere packings may be quantified by the following measures: The packing radius (a.k.a. the error-correction radius) of a packing, ρ, is the maximal radius of the spheres in a set of identical nonoverlapping spheres centered at each nodal point. The packing density of a packing, δ, is the fraction of the volume of the domain included within a set of identical non-overlapping spheres of radius r centered at each nodal point on the packing. Packings that maximize this metric are referred to as close-packed. The covering radius of a packing, R, is the maximum distance between any point in the domain and its nearest nodal point on the packing. The deep holes of a packing are those points which are at a distance R from all of their nearest neighbors. The covering thickness of a packing, Ω, is the number of spheres of radius R centered at each nodal point containing an arbitrary point in the domain, averaged over the domain. The mean squared quantization error per dimension of a lattice or uninodal nonlattice packing, G, is the average mean square distance of any point in the domain to its nearest nodal point, normalized by n times the appropriate power of the volume, V, of the Voronoi cell, and can be intuitively understood as the average distance from an arbitrary point in the domain to the nearest lattice point, normalized by dimension.

The efficiency of a lattice in quantizing or discretizing space can be captured via these metrics. Figure 2 captures an intuitive understanding of how the A_2, or honeycomb, lattice, is significantly denser than the equivalent Cartesian grid. That is, the packing density is higher, the covering thickness is lower, and the mean squared quantization error is higher. Analogously, the A_n and D_n lattices offer similarly greater efficiency than the Cartesian grid in the appropriate dimension. As described in Part I, there are several lattices in particular dimensions that offer especially good performance, as measured by these metrics. Such a lattice is the E_8 lattice, which is used in $n = 8$ throughout this manuscript.

2 Extending lattice theory for the coordination of derivative-free optimization

To extend the lattice theory described in Part I of this study in order to coordinate a derivative-free optimization, a few additional component algorithms are needed. Most importantly, in order to implement the SP algorithm described above, a positive basis must be located upon the lattice points neighboring the CMP (referred to as the 'nearest neighbors'). Calculating the nearest neighbor points is simple, and well described in Part I. Once the nearest neighbors have been calculated, an algorithm for locating a positive basis is necessary.

Note that in the special case of coordinating an SP algorithm via the Cartesian lattice, the polling set can be selected very simply as the maximal positive basis consisting of all the nearest neighbors (simply the Cartesian basis vectors), or a simple minimal positive basis consisting of the positive Cartesian basis vectors as well as the vector $[-1 -1 -1 \ldots -1]$. The

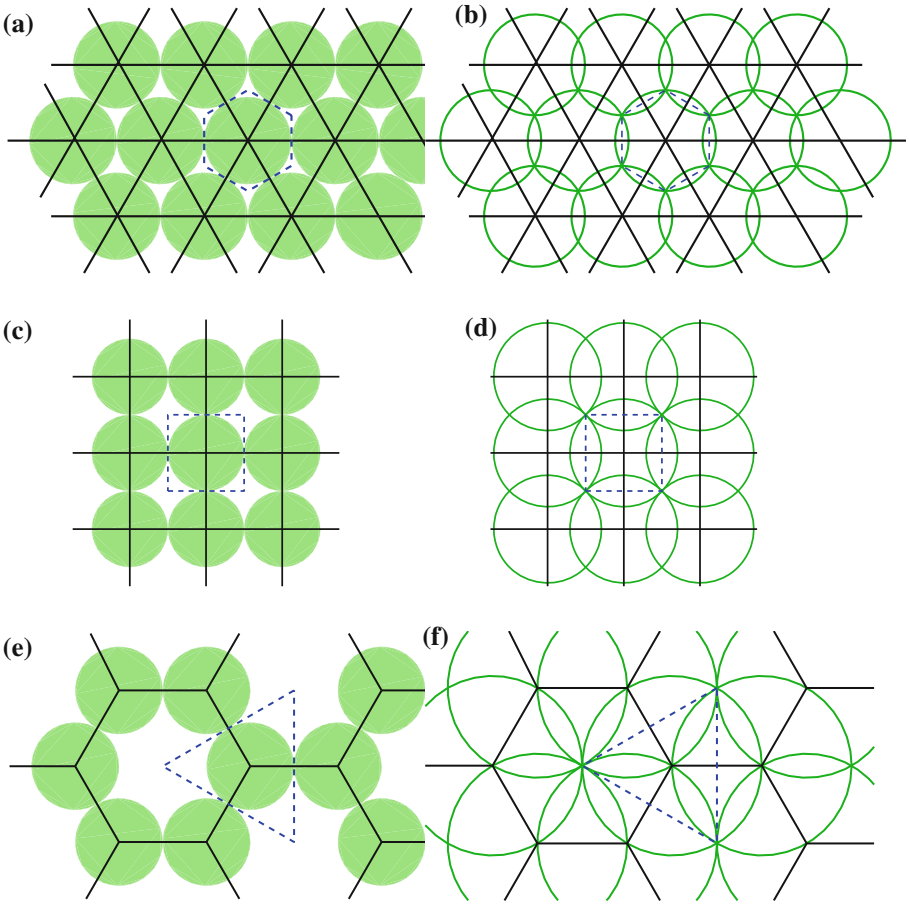

Fig. 2 The *triangular* lattice (**a**, **b**), the *square* lattice (**c**, **d**), and the honeycomb nonlattice packing (**e**, **f**). Indicated in the *left* three subfigures is the *packing* with spheres of radius ρ, the corresponding *net* or *contact graph* (*solid lines*), a typical *Voronoï cell* (*dashed line*), and the *kissing* (that is, the contact points between the spheres). Indicated in the *right* three subfigures is the *covering* with spheres of radius R. Looking at their respective packing densities in Table 1, as compared with the *square* lattice, the *triangular* lattice is said to be *dense*, and the honeycomb packing is said to be *rare*

highly efficient lattices introduced above produce a more challenging scenario as the number of nearest neighbor points is much greater than that of the Cartesian lattice; therefore, an efficient algorithm for locating a positive basis is needed.

Secondly, an algorithm for maintaining convergence characteristics of the SP optimization in the presence of hard constraints is needed. Previous implementations on the Cartesian grid assume hard 'box' constraints where, upon appropriate scaling, grid points are always present on the constraint surfaces; therefore simply by switching to utilizing a maximal basis near the constraint surfaces ensures appropriate convergence. However, as alternative lattices do not necessarily have points discretizing the constraint surfaces, a more sophisticated approach is necessary.

Algorithms for selecting an appropriate basis on an arbitrary lattice and defining a SP optimization for linear constraints are considered next.

We begin with a short historical retrospective.

Thomson (1904), in his study of the structure of the atom, is credited with being the first to address the problem[3]: "Where should k inimical dictators settle on a planet in order to be as far away from each other as possible?" This question extends naturally to n-dimensional planets, and has received significant attention in the years since Thomson's original paper. The question is readily answered numerically by assigning an identical "charge" to each of n identical "particles", restricting particle motion to the surface of the sphere, and iteratively moving each particle (with some damping applied) in the direction of the force caused by the other particles (projected onto the sphere) until all particles come to equilibrium. The precise solution reached is a function of the distance metric and power law used when computing the force between any two particles; in the electrostatic setting, Thomson used the Euclidian distance between the particles, and a force which is proportional to the inverse square of this distance. The setting based on other distance measures (e.g., measured along the surface of the sphere instead of along a straight line) and other power laws are referred to as generalized Thomson problems; in particular, the case based on the p'th power in the limit that $p \to \infty$ (that is, the max value) was studied in Tammes (1930), in his study of the boundaries of pollen grains.

Next, note that $n+1$ charged particles on an n dimensional sphere will move to minimize the sum potential energy. Note also that at equilibrium, these $n+1$ points form precisely a uniformly distributed minimal positive basis—exactly the distribution that is desired for the SP algorithm.

We now generalize this classical question in two ways, and introduce a new metric to characterize the solution found:

- First, the locations where the particles are allowed to settle are restricted to a discrete set of points on a sphere, which are specified as the nearest-neighbor lattice points to the CMP.
- Next, we allow some the particles' locations on the sphere to be specified (that is, fixed) in advance, and only move the remaining (free) particles to arrive at the best solution possible.
- Finally, the new metric we introduce is a check of whether or not the distribution produced by numerical solution of the resulting "discrete Thomson problem" forms a *positive basis* of the feasible neighborhood of the CMP; that is, in the case with no active constraints (cf. Sect. 2.3), whether or not all points on the unit sphere around the CMP can be reached via a linear combination *with non-negative coefficients* of the vectors from the CMP to the optimized particle locations.

After developing a method to test for a positive basis, the remainder of this section develops three efficient algorithms to iterate on this "discrete Thomson problem" until a positive basis is found. To accomplish this, these algorithms first solve the discrete Thomson problem numerically for $n+m$ particles where $m = 1$. If the optimization algorithm succeeds in producing a positive basis, the algorithm exits; otherwise, m is increased by one and the process repeated until a positive basis is determined. The resulting algoroithm is leveraged heavily during the poll step of the lattice-based SMF algorithm developed later in this paper.

[3] This curious problem, articulated by Meschkowski (1960) in terms of inimical dictators (see also Fejes Tóth 1971), assumes that all locations on the planet's surface are equally desirable, and that the inimical dictators all cooperate.

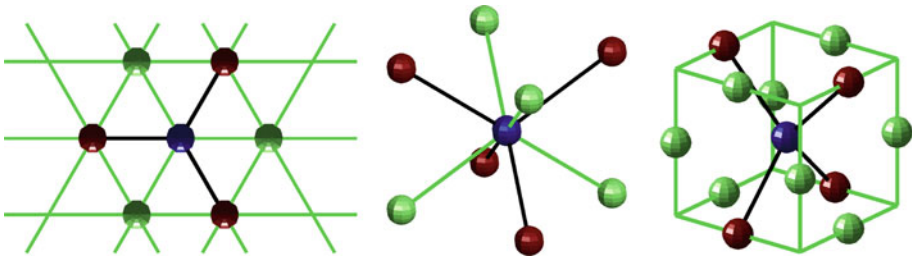

Fig. 3 Various minimal positive bases (shown in *red*) around the origin (shown in *blue*) in the (*left*) triangular, (*center*) BCC, and (*right*) FCC lattices. Note that the triangular and BCC lattices each have two perfectly distributed minimal positive bases. In contrast, there are several choices for selecting a minimal positive basis in the FCC lattice, but none is perfectly distributed. (Color figure online)

2.1 Testing for a positive basis

Given a subset of the nearest-neighbor lattice points, we will at times need an efficient test to determine whether or not the vectors to these points from the CMP form a positive basis of the feasible domain around the CMP. Without loss of generality, we will shift this problem so that the CMP corresponds to the origin in the discussion that follows.

A set of vectors $\{\tilde{\mathbf{x}}^1, \ldots, \tilde{\mathbf{x}}^k\}$ for $k \geq n+1$ is said to *positively span* $\mathbb{R}^n$ if any point in $\mathbb{R}^n$ may be reached via a linear combination of these vectors with non-negative coefficients. Since the $2n$ basis vectors $\{\mathbf{e}^1, \ldots, \mathbf{e}^n, -\mathbf{e}^1, \ldots, -\mathbf{e}^n\}$ positively span $\mathbb{R}^n$, a convenient test for whether or not the vectors $\{\tilde{\mathbf{x}}^1, \ldots, \tilde{\mathbf{x}}^k\}$ positively span $\mathbb{R}^n$ is to determine whether or not each vector in the set $\mathsf{E} = \{\mathbf{e}^1, \ldots, \mathbf{e}^n, -\mathbf{e}^1, \ldots, -\mathbf{e}^n\}$ can be reached by a positive linear combination of the vectors $\{\tilde{\mathbf{x}}^1, \ldots, \tilde{\mathbf{x}}^k\}$. That is, for each vector $\mathbf{e} \in \mathsf{E}$, a solution $\mathbf{z}$, with $z_i \geq 0$ for $i = 1, \ldots, k$, to the equation $\tilde{X}\mathbf{z} = \mathbf{e}$ is sought, where $\tilde{X} = \left(\tilde{\mathbf{x}}^1 \ \ldots \ \tilde{\mathbf{x}}^k\right)$. If such a $\mathbf{z}$ exists for each vector $\mathbf{e} \in \mathsf{E}$, then the vectors $\{\tilde{\mathbf{x}}^1, \ldots, \tilde{\mathbf{x}}^k\}$ positively span $\mathbb{R}^n$; if such a $\mathbf{z}$ does not exist, then the vectors $\{\tilde{\mathbf{x}}^1, \ldots, \tilde{\mathbf{x}}^k\}$ do not positively span $\mathbb{R}^n$.

Thus, testing a set of vectors to determine whether or not it positively spans $\mathbb{R}^n$ reduces simply to testing for the existence of a solution to $2n$ well-defined *linear programs* in standard form. Techniques to perform such tests, such as Matlab's `linprog` algorithm, are well developed and readily available. Further, if a set of k vectors positively spans $\mathbb{R}^n$, it is a simple matter to check whether or not this set of vectors is also a positive basis of $\mathbb{R}^n$, if such a check is necessary, simply by checking whether or not any subset of $k-1$ vectors chosen from this set also positively span $\mathbb{R}^n$. Note that a positive basis with k vectors will necessarily have k in the range $n+1 \leq k \leq 2n$; the case with $k = n+1$ is referred to as a *minimal* positive basis, and the case with $k = 2n$ is referred to as a *maximal* positive basis.

2.2 Selecting a positive basis from the nearest-neighbor lattice points

In Sect. 6 of Part I, we described how to enumerate all points which are nearest neighbors of the origin of a lattice (and thus, with the appropriate shift, all points which are nearest neighbors of any CMP on the lattice). In Sect. 2.1 above, we described how to test a subset of such points to see if the vectors from the origin to these points form a positive basis around the CMP. We now present a general algorithm to solve the problem of selecting a positive basis from the nearest-neighbors of the CMP using a minimal number of new poll points, while creating the maximum achievable angular uniformity between the vectors from the CMP to each of these points (that is, while minimizing the skewness of the resulting poll set). Note in Fig. 3 that, as the number of nearest neighbors increases, the flexibility in solving

this (apparently, NP-hard) problem also increases, though a perfectly distributed minimal positive basis (using $n+1$ points) is not always available. Ideally, for $m=1$, the solution to the discrete Thomson problem will produce a positive basis with good angular uniformity; if it does not, we may successively increment m by one and try again until we succeed in producing a positive basis. We have studied three algorithms for solving this problem:

Algorithm A If the kissing number τ of the lattice under consideration is relatively large (that is, if $\tau \gg n$; for example, for the Leech lattice Λ_{24}), then a straightforward algorithm can first be used to solve Thomson's problem on a continuous sphere in n dimensions. This can be done simply and quickly by fixing $q \geq 0$ repulsive particles at the prespecified lattice points, and initializing $n+m-q$ free repulsive particles on the sphere randomly. Then, at each iteration, a straightforward force-based algorithm may be used to move each free particle along the surface of the sphere a small amount in the direction that the other particles are tending to push it, and iterating until the set of particles approaches an equilibrium. The free particle that is nearest to a nearest-neighbor lattice point around the CMP is then moved to said lattice point and fixed there, and the remaining free particles adjusted until they reach a new equilibrium. This adjust/fix/adjust/fix sequence is repeated until all particles are fixed at lattice points.

Algorithm B If the kissing number τ of the lattice under consideration is relatively small (that is, if τ is *not* well over an order of magnitude larger than n), then it turns out to be more expedient to solve the discrete Thomson problem directly. To accomplish this, again taking the q prespecified repulsive particles as fixed, we initialize $n+m-q$ free repulsive particles randomly on $n+m-q$ nearest-neighbor lattice points around the CMP and then, at each iteration, move the two or three[4] free particles that are furthest from equilibrium in the force-based model described above (that is, those free particles which have the highest force component projected onto the surface of the sphere) into new positions selected from the available locations in such a way as to minimize the maximum force (projected onto the sphere) over the entire set of (fixed and free) particles. Though each iteration of this algorithm involves an exhaustive search for placing the two or three free particles in question, it converges quickly when τ is $O(100)$ or less.

Algorithm C For intermediate kissing numbers τ, a hybrid approach may be used: a "good" initial distribution may be found using Algorithm *A*, then this distribution may be refined using Algorithm *B*.

In each of these algorithms, to minimize the number of new function evaluations required at each poll step, a check is first made to determine whether any previous function evaluations have already been performed on the nearest-neighbor lattice points around the CMP. If so, then particles are fixed at these locations, while the remaining particles are adjusted via one of the three algorithms described above. By so doing, previously-calculated function values may be used with maximum effectiveness during the polling procedure. When performing the poll step of a surrogate-based search, in order to orient the new poll set favorably (and, on average, exit the poll step quickly), a particle may also be fixed at the nearest neighbor point with the lowest value of the surrogate function; when polling, this poll point is thus evaluated first.

[4] Moving more than two or three particles at a time in this algorithm makes each iteration computationally intensive, and has little impact on overall convergence of the algorithm, whereas moving only one at a time is found to significantly impede convergence to the optimal solution.

The iterative algorithms described above, though in practice quite effective, are not guaranteed to converge from arbitrary initial conditions to a positive basis for a given value of m, even if such a positive basis exists. To address this issue, if the algorithm used fails to produce a positive basis, the algorithm may be repeated using a new random starting distribution. Our numerical tests indicate that this repeated random initialization scheme usually generates a positive basis within a few initializations when such a positive basis indeed exists. Since at times, for a given m, there exists no configuration of the free particles on the nearest-neighbor lattice points that produces a positive basis, particularly when the previous function evaluations being leveraged are poorly configured, the number of new random initializations is limited to a prespecified value. Once this value is reached, m is increased by one and the process repeated. As the cost of each function evaluation increases, the user can increase the number of random initializations attempted using one of the above algorithms for each value of m in order to avoid the computation of extraneous poll points that might in fact be unnecessary if sufficient exploration by the discrete Thomson algorithm described above is performed.

Numerical tests have demonstrated the efficacy of this rather simple strategy, which reliably generates a positive basis while keeping computational costs to a minimum even when leveraging a relatively poor configuration of previous function evaluations and when working in relatively high dimension n. Additionally, the algorithm itself is independent of the lattice being used; the only inputs to the algorithm are the dimension of the problem, the locations of the nearest-neighbor lattice points, and the identification of those nearest-neighbor lattice points for which previous function evaluations are available.

2.3 Implementation of feasible domain boundaries

When implementing a global search in n dimensions, or even when implementing a local search on a function which is ill-defined for certain nonphysical values of the parameters (such as negative concentrations of chemicals), it is important to restrict the optimization algorithm to look only over a prespecified "feasible" region of parameter space. For simplicity, the present work assumes rectangular constraints on this feasible domain (that is, simple upper and lower bounds on each parameter value). An efficient n-dimensional lattice with packing radius ρ_n is used to quantize the interior of the feasible domain, efficient $(n-1)$-dimensional lattices with packing radius $\rho_{n-1} = \rho_n/2$ are used to quantize the portions of the boundary of the feasible domain with one active constraint (that is, the "faces"), efficient $(n-2)$-dimensional lattices with packing radius $\rho_{n-2} = \rho_n/4$ are used to quantize the portions of the boundary of the feasible domain with two active constraints (that is, the "edges"), etc. The present section describes how to search over the boundaries of the feasible domain, and how to move on and off of these boundaries as appropriate, while carefully restricting all function evaluations to the interior and boundary lattices in order to coordinate an efficient search.

We distinguish between two scenarios in which the polling algorithm as described thus far must be adjusted to avoid violating the $(n-1)$-dimensional boundaries[5] of the feasible domain. In the first scenario, the CMP is relatively far (that is, greater than ρ_n but less than $2\rho_n$) from the boundary of the feasible domain, and thus one or more of the poll points as determined by one of the algorithms proposed in Sect. 2.2 might land slightly outside this boundary. In this scenario, an effective remedy is simply to *eliminate* all lattice points which land outside of the feasible domain from the list of potential poll points, and then

[5] That is, the portions of the boundary with a single active constraint.

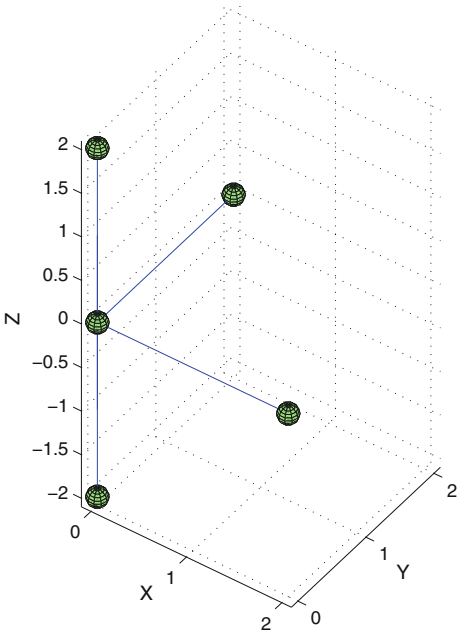

Fig. 4 A scenario in which a CMP at $\mathbf{x} = (0\,0\,0)^T$ sits on an $(n-2) = 1$-dimensional edge of an $n = 3$-dimensional feasible region with bounds $x_1 \geq 0$ and $x_2 \geq 0$. Note that the feasible neighborhood of this edge is positively spanned by the nearest neighbors on the integer lattice, and that two additional vectors are added to the poll set to facilitate moving off of each of these active constraint boundaries

to *augment* this restricted list of potential poll points with all lattice points on the nearby $(n-1)$-dimensional constraint surface which are less than $2\rho_n$ from the CMP. From this modified list of potential poll points, the poll set may be selected in the usual fashion using one of the algorithms described in Sect. 2.2.

In the second scenario, the CMP is relatively close (that is, less than ρ_n) to the boundary of the feasible domain. In this scenario, it is most effective simply to shift the CMP onto the nearest lattice point on the $(n-1)$-dimensional constraint surface. With the CMP on the feasible domain boundary, each poll step explores a minimum positive basis selected on the lattice quantizing the $(n-1)$-dimensional boundary and, in addition, polls an additional lattice point on the interior of the feasible domain to allow the algorithm to move back off this constraint boundary. Ideally, this additional point would be located on a inward-facing vector normal to the $(n-1)$-dimensional feasible domain boundary a distance ρ_n from the CMP; we thus choose the interior lattice point closest to this location.

Multiple active constraints are handled in an analogous manner (see Fig. 4). In an n-dimensional optimization problem with $p \geq 2$ active constraints, the CMP is located on an active constraint "surface" of dimension $n-p$. An efficient $(n-p)$-dimensional lattice with packing radius $\rho_{n-p} = \rho_n/2^p$ is used to quantize this active constraint surface, and a poll set is constructed by creating a positive basis selected from the points neighboring the CMP within the $(n-p)$-dimensional active constraint surface, together with p additional points located on the $(n-p+1)$-dimensional constraint surfaces neighboring the CMP. Ideally, these p additional points would be located on vectors normal to the $(n-p)$-dimensional active constraint surface a distance $\rho_{n-p+1} = \rho_n/2^{p-1}$ from the CMP; we thus choose the lattice points on the $(n-p+1)$-dimensional feasible domain boundaries closest to these locations.

In practice, it is found that, once an optimization routine moves onto $p \geq 1$ feasible domain boundaries, it only somewhat infrequently moves back off. To account for this, the

p additional poll points mentioned in the previous paragraph are polled *after* the other poll points forming the positive basis within the $(n - p)$-dimensional active constraint surface.

In the case that the constraints on the cost function are not of the simple 'box' variety, these algorithms can still be applied. In the case of linear constraints the above methodology can be extended to again define a lattice on the active constraints. When the constraint surfaces are nonlinear or otherwise exhibit more challenging behavior, the convergence characteristics of the SP step can be extended by substituting a more sophisticated class of algorithms known as mesh adaptive direct search (or MADS) algorithms (see Audet and Dennis 2006; Abramson et al. 2008; Belitz 2011). A comprehensive discussion of the application of lattices to the MADS class of algorithms, and the resultant algorithm, Λ-MADS, can be found at Belitz (2011), Chapter 4. Analogously to the SP algorithms presented herein, the application of lattice in Λ-MADS leads to significantly improved convergence rates, while maintaining all the convergence characteristics of, the leading Cartesian-based MADS algorithm.

2.4 Quantifying the skewness of best nearest-neighbor minimal positive bases

A final relevant metric of a lattice that relates to the performance of the corresponding lattice-based optimization is the deviation from perfect uniformity of the best minimal positive basis available on nearest-neighbor lattice points. The *best nearest-neighbor minimal positive basis skewness* of a lattice, s, is thus now defined as the ratio between the largest and the smallest angles between any two vectors in the best minimal positive basis available on nearest-neighbor lattice points, minus one. Therefore, $s = 0$ indicates a perfectly uniform minimal positive basis on nearest-neighbor lattice points, as exhibited by A_2 (see Fig. 3a). In contrast, A_3 through A_8 all have $s = 0.3333$ (see, e.g., A_3 in Fig. 3c).

Surprisingly, the best nearest-neighbor minimal positive basis skewness of E_8 is also $s = 0.3333$; one might initially expect it to be much smaller than this (indeed, one might hope that it would be fairly close to $s = 0$) due to the relatively large kissing number ($\tau = 240$) of this $n = 8$ lattice. Interestingly, the best nearest-neighbor positive basis of E_8 when using $n + 2$ points (that is, instead of a minimal positive basis with $n + 1$ points) is perfectly uniform. The tests reported in Sect. 5 thus use $n + 2$ points instead of $n + 1$ points when polling on the E_8 lattice.

A minimal positive basis on nearest-neighbor lattice points doesn't even exist on the $\mathbb{Z}^n$ lattice (indeed, a positive basis on nearest neighbors of the $\mathbb{Z}^n$ lattice requires a full $2n$ points). This was, in fact, a matter of significant inconvenience in previous work when using the Cartesian lattice as the default choice for such problems, as using a maximal positive basis rather than a minimal positive basis essentially doubles the cost of each complete poll step for large n. When developing a minimal positive basis for the $\mathbb{Z}^n$ lattice, it is thus common (see, e.g., Booker et al. 1999) to select the Cartesian unit vectors $\mathbf{e}^1$ through $\mathbf{e}^n$ and one additional "oddball" vector in the $(-1, -1, \ldots, -1)$ direction which is $\sqrt{n}$ longer. Note the "clustering" of the Cartesian unit vectors in directions generally opposite to the oddball vector. To quantify, the skewness of this minimal positive basis is $\cos^{-1}(-1/\sqrt{n})/(\pi/2) - 1$, which in dimensions $n = 2$ through 8 is given by 0.5, 0.3918, 0.3333, 0.2952, 0.2677, 0.2468, and 0.2301. Note that, while the skewness of the angular distribution of this minimal positive basis actually decreases gradually as the dimension of the problem increases, the ratio in lengths of the vectors to the nearest-neighbor lattice points and the oddball vector in this basis increases like $\sqrt{n}$ (that is, from 1.4142 in $n = 2$ to 2.8284 in $n = 8$). This is quite unfortunate, as it leads to a peculiar non-isotropic behavior of the optimization algorithm over parameter space (for further discussion on this point, see the sixth paragraph of Sect. 5.1). The tests reported in Sect. 5 use

Table 3 Performance comparison between the A_n-based SP algorithm and the $\mathbb{Z}^n$-based SP algorithm applied to randomly shifted quadratic bowls for $n = 2$ to 8

n	2	3	4	5	6	7	8
p	74.77	81.32	84.03	84.53	84.43	84.56	85.28
r	0.4290	0.4161	0.3273	0.3585	0.3150	0.3345	0.3060

It is seen that the A_8-based SP algorithm outperformed the $\mathbb{Z}^8$-based SP algorithm 85% of the time, and on average required 30% as many function evaluations to reach the same level of convergence

this peculiar minimum positive basis, with a long oddball vector, when polling on the $\mathbb{Z}^n$ lattice.

We now have all of the ingredients necessary to coordinate a GPS algorithm, as laid out previously, with any of the lattices listed in Tables 2, 3 of Part I, while both reusing previous function evaluations and respecting sharp bounds on the feasible region of parameter space. Numerical testing of such an algorithm is reported in Sect. 5.

3 A review of the Kriging interpolation strategy

3.1 Interpolation: basic concepts

The purpose of the search step of an SMF algorithm is to interpolate, and extrapolate, the trends exhibited by the existing function evaluations in order to suggest new regions of parameter space, perhaps far from the CMP, where the function value is anticipated, with some reasonable degree of probability, to be lower than that of the CMP. There are a variety of possible ways of accomplishing this; we leverage here the Kriging interpolation strategy (Krige 1951; Matheron 1963; Jones 2001; Rasmussen and Williams 2006).

The problem of interpolation is the problem of drawing a smooth curve through data points in order to estimate the function values in regions where the function itself has not yet been computed. The problem of interpolation, thus, necessarily builds on some hypothesis that models the function behavior in order to "connect the dots". The most common such model is a mechanical one, based on a thin piece of wood, or "spline", that is "bent" in order to touch all the data points; this mechanical model leads directly to the mathematical algorithm known as cubic spline interpolation. A perhaps equally valid hypothesis, which forms the foundation for the Kriging interpolation strategy, is to *model the underlying function as a realization of some stochastic process*. The stochastic model used in this approach is selected to be general enough to model a broad range of functions reasonably well, yet simple enough to be fairly inexpensive to tune appropriately based on the measured data. There are many such stochastic models which one can select; the simple stochastic model considered here leads to the easy-to-use interpolation strategy commonly referred to as ordinary Kriging.

3.2 Statistical modeling assumptions of the ordinary Kriging model

The PDF of the random vector $\mathbf{f} = \mathbf{f}_{n\times 1}$ in this analysis is modelled as Gaussian, and is thus restricted to the generic form

$$p_{\mathbf{f}}(\mathbf{f}') = \frac{1}{(2\pi)^{n/2}|C_{\mathbf{f}}|^{1/2}} \exp \frac{-(\mathbf{f}' - \bar{\mathbf{f}})^T C_{\mathbf{f}}^{-1} (\mathbf{f}' - \bar{\mathbf{f}})}{2}, \tag{1a}$$

where the covariance $C_{\mathbf{f}}$ is modelled as a constant σ^2, referred to as the variance, times a correlation matrix R whose $\{i, j\}$'th component r_{ij} is given by a model of the correlation of the random function f between points $\mathbf{x}^i$ and $\mathbf{x}^j$, where this correlation model $r(\cdot, \cdot)$ itself decays exponentially with the distance between points $\mathbf{x}^i$ and $\mathbf{x}^j$; that is,

$$C_{\mathbf{f}} \triangleq \sigma^2 R, \quad \text{where} \quad r_{ij} \triangleq r(\mathbf{x}^i, \mathbf{x}^j) \quad \text{and} \quad r(\mathbf{x}, \mathbf{y}) \triangleq \prod_{\ell=1}^{n} \exp(-\theta_\ell |x_\ell - y_\ell|^{p_\ell}) \tag{1b}$$

for some yet-to-be-determined constants $\sigma^2, \theta_\ell > 0$, and $0 < p_\ell \leq 2$ for $\ell = 1, \ldots, n$. The mean $\bar{\mathbf{f}}$ in the Gaussian model (1a) is itself modelled as uniform over all of its components:

$$\bar{\mathbf{f}} \triangleq \mathbf{1}\mu \tag{1c}$$

for some yet-to-be-determined constant μ. There is extensive debate in the recent literature (see, e.g., Isaaks and Srivastav 1989; Rasmussen and Williams 2006) on the statistical modeling assumptions one should use in a Kriging model of this sort. It is straightforward to extend the present investigation to incorporate less restrictive Kriging models; the ordinary Kriging model is used here primarily due to its simplicity.

3.3 Adjusting the coefficients of the model based on the data

If the vector of observed function values is

$$\mathbf{f}^o = \begin{pmatrix} f_1^o \\ \vdots \\ f_N^o \end{pmatrix},$$

then the PDF corresponding to this observation in the statistical model proposed in (1) can be written as

$$p_{\mathbf{f}}(\mathbf{f}^o) = \frac{1}{(2\pi)^{n/2}(\sigma^2)^{n/2}|R|^{1/2}} \exp \frac{-(\mathbf{f}^o - \mu\mathbf{1})^T R^{-1}(\mathbf{f}^o - \mu\mathbf{1})}{2\sigma^2}. \tag{2}$$

The process of Kriging modeling boils down to selecting the parameters σ^2, θ_ℓ, p_ℓ, and μ in the statistical model proposed in (1) to maximize the PDF evaluated for the function values actually observed, $\mathbf{f} = \mathbf{f}^o$, as given in (2).

Maximizing $p_{\mathbf{f}}(\mathbf{f}^o)$ is equivalent to minimizing the negative of its log. Thus, for simplicity, consider

$$J = -\log[p_{\mathbf{f}}(\mathbf{f}^o)] = \frac{n}{2}\log(2\pi) + \frac{n}{2}\log(\sigma^2) + \frac{1}{2}\log(|R|) + \frac{(\mathbf{f}^o - \mu\mathbf{1})^T R^{-1}(\mathbf{f}^o - \mu\mathbf{1})}{2\sigma^2}. \tag{3}$$

Setting the derivatives of J with respect to μ and σ^2 equal to zero and solving, the optimal values of μ and σ^2 are determined immediately:

$$\mu = \frac{\mathbf{1}^T R^{-1}\mathbf{f}^o}{\mathbf{1}^T R^{-1}\mathbf{1}}, \qquad \sigma^2 = \frac{(\mathbf{f}^o - \mu\mathbf{1})^T R^{-1}(\mathbf{f}^o - \mu\mathbf{1})}{n}. \tag{4}$$

With these optimal values of μ and σ^2 applied, noting that the last term in (3) is now constant, what remains to be done is to minimize

$$J_1 = \frac{n}{2}\log(\sigma^2) + \frac{1}{2}\log(|R|) \tag{5}$$

with respect to the remaining free parameters[6] θ_ℓ and p_ℓ, where σ^2 is given as a function of R in (4) and R, in turn, is given as a function of the free parameters θ_ℓ and p_ℓ in (1b). This minimization must, in general, be performed numerically. However, the function J_1 is smooth in the parameters θ_ℓ and p_ℓ, so this optimization may be performed efficiently with a standard gradient-based algorithm, such as the nonquadratic conjugate gradient algorithm, where the gradient itself, for simplicity, may easily be determined via a simple finite difference or complex-step derivative approach.

Note that, after each new function evaluation, the Kriging parameters often adjust only slightly, and thus the previously-converged values of these parameters form a good initial guess for this gradient-based optimization algorithm. Note also that, while performing this optimization, the determinant of the correlation matrix occasionally reaches machine zero. To avoid the numerical difficulty that taking the log of zero would otherwise induce, a small $[O(10^{-6})]$ term may be added to the diagonal elements of R. By so doing, the Kriging predictor does not quite have the value of the sampled data at each sampled point; however, it remains quite close, and the algorithm is made numerically robust (Booker et al. 1999).

3.4 Using the tuned statistical model to predict the function value at new locations

Once the parameters of the stochastic model have been tuned as described above, the tuned Kriging model facilitates the computationally inexpensive prediction of the function value at any new location $\bar{\mathbf{x}}$. To perform this prediction, consider now the $N+1$ points $\{\mathbf{x}^1, \ldots, \mathbf{x}^N, \bar{\mathbf{x}}\}$, and model the function's value at these $N+1$ points with the vector

$$\bar{\mathbf{f}} = \begin{pmatrix} \mathbf{f} \\ f(\bar{\mathbf{x}}) \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ \bar{f} \end{pmatrix},$$

where $\mathbf{f}$ is the $N \times 1$ random vector considered previously and $\bar{f}$ is the random scalar modeling the function at the new point. Analogous statistical assumptions as laid out in (1) are again applied, with the correlation matrix now written as

$$\bar{R} = \begin{bmatrix} R & \bar{\mathbf{r}} \\ \bar{\mathbf{r}}^T & 1 \end{bmatrix}, \qquad P_{\bar{\mathbf{f}}} \triangleq \sigma^2 \bar{R}, \tag{6}$$

where R is the $N \times N$ correlation matrix considered previously and, consistent with this definition, the vector $\bar{\mathbf{r}}$ is constructed with components

$$\bar{r}_i = r(\mathbf{x}^i, \bar{\mathbf{x}}), \quad \text{where} \quad r(\mathbf{x}, \mathbf{y}) \triangleq \prod_{\ell=1}^{n} \exp(-\theta_\ell |x_\ell - y_\ell|^{p_\ell}).$$

Following Jones (2001), note by the matrix inversion lemma that $\bar{R}^{-1}$ may be written

$$\bar{R}^{-1} = \begin{bmatrix} R & \bar{\mathbf{r}} \\ \bar{\mathbf{r}}^T & 1 \end{bmatrix}^{-1} = \begin{bmatrix} R^{-1} + R^{-1}\bar{\mathbf{r}}(1 - \bar{\mathbf{r}}^T R^{-1}\bar{\mathbf{r}})^{-1}\bar{\mathbf{r}}^T R^{-1} & -R^{-1}\bar{\mathbf{r}}(1 - \bar{\mathbf{r}}^T R^{-1}\bar{\mathbf{r}})^{-1} \\ -(1 - \bar{\mathbf{r}}^T R^{-1}\bar{\mathbf{r}})^{-1}\bar{\mathbf{r}}^T R^{-1} & (1 - \bar{\mathbf{r}}^T R^{-1}\bar{\mathbf{r}})^{-1} \end{bmatrix}. \tag{7}$$

[6] To simplify this optimization, p_ℓ may be specified by the user instead of being determined via optimization; this is especially appropriate to do when the number of function evaluations N is relatively small, and thus there is not yet enough data to determine both the θ_ℓ and p_ℓ uniquely. If this approach is followed, $p_\ell = 1$ or 2 are natural choices; the case with $p_\ell = 1$ is referred to as an Ornstein-Uhlenbeck process, whereas the case with $p_\ell = 2$ is infinitely differentiable everywhere.

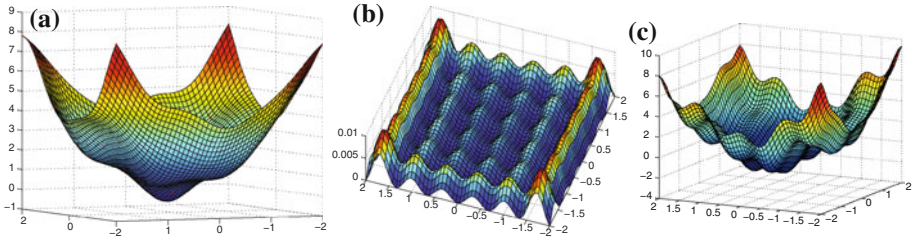

Fig. 5 **a** The Kriging predictor, $\hat{f}(\mathbf{x})$, and **b** its associated uncertainty, $s^2(\mathbf{x})$, for a perturbed quadratic bowl sampled on a square grid of 7×7 points. **c** The corresponding $J(\mathbf{x}) = \hat{f}(\mathbf{x}) - c \cdot s^2(\mathbf{x})$ search function used for a global search in two dimensions (see Sect. 4)

Keeping the paramter values $\sigma^2, \theta_\ell, p_\ell$, and μ as tuned previously, we now examine the variation of the PDF in the remaining unknown random variable, $\bar{f}$. Substituting (6) and (7) into a PDF of the form (1a), we may write

$$p_{\bar{\mathbf{f}}}(\bar{\mathbf{f}}\,') = C_1 \cdot \exp \frac{-(\bar{\mathbf{f}}\,' - \mu\mathbf{1})^T \bar{R}^{-1} (\bar{\mathbf{f}}\,' - \mu\mathbf{1})}{2\sigma^2} = C_1 \cdot \exp \frac{-\begin{bmatrix} \mathbf{f}\,' - \mu\mathbf{1} \\ \bar{f}' - \mu \end{bmatrix}^T \bar{R}^{-1} \begin{bmatrix} \mathbf{f}\,' - \mu\mathbf{1} \\ \bar{f}' - \mu \end{bmatrix}}{2\sigma^2}$$
$$= \cdots = C_2 \cdot \exp \frac{-[\bar{f}' - \hat{f}]^T [\bar{f}' - \hat{f}]}{2s^2}, \tag{8}$$

where, with a minor amount of algebraic rearrangement, the mean and variance of this scalar Gaussian distribution modeling the random scalar $\bar{f}$ work out to be[7]

$$\hat{f}(\bar{\mathbf{x}}) = \mathcal{E}\{f(\bar{\mathbf{x}})\} = \mathcal{E}\{\bar{f}\} = \mu + \mathbf{r}^T R^{-1} (\mathbf{f}^o - \mu\mathbf{1}), \tag{9a}$$
$$s^2(\bar{\mathbf{x}}) = \mathcal{E}\{[f(\bar{\mathbf{x}}) - \hat{f}]^2\} = \mathcal{E}\{[\bar{f} - \hat{f}]^2\} = \sigma^2 (1 - \mathbf{r}^T R^{-1} \mathbf{r}). \tag{9b}$$

Equations (9a–9b) give the final formulae for the Kriging predictor, $\hat{f}(\bar{\mathbf{x}})$, and its associated uncertainty, $s^2(\bar{\mathbf{x}})$.

When applied numerically to a representative test problem, as expected, the Kriging predictor function, which we denote $\hat{f}(\bar{\mathbf{x}})$, interpolates [that is, it goes through every observed function value at points $\bar{\mathbf{x}} = \mathbf{x}^1$ to $\bar{\mathbf{x}} = \mathbf{x}^N$], whereas the uncertainty function, denoted $s^2(\bar{\mathbf{x}})$, is zero at each sampled point, and resembles a Gaussian "bump" between these sampled points, as seen in Fig. 5. Note that, once the parameters of the statistical model have been determined, as described in Sect. 3.3, the formula (9a)–(9b) for the Kriging predictor $\hat{f}(\bar{\mathbf{x}})$ and its corresponding uncertainty $s^2(\bar{\mathbf{x}})$ at any test point $\bar{\mathbf{x}}$ is computationally quite inexpensive.[8]

[7] An alternative interpretation of this process models the constant μ itself as a stochastic variable rather than as a constant. Following this line of reasoning ultimately gives the same formula for the predictor $\hat{f}(\bar{\mathbf{x}})$ as given in (9a), and a slightly modified formula for its associated uncertainty,

$$s^2(\bar{\mathbf{x}}) = \sigma^2 \Big(1 - \mathbf{r}^T R^{-1} \mathbf{r} + \frac{(1 - \mathbf{r}^T R^{-1} \mathbf{r})^2}{\mathbf{1}^T R^{-1} \mathbf{1}}\Big). \tag{9b′}$$

Which formula (9b) or (9b′) is used in the present model is ultimately a matter of little consequence as far as the overall derivative-free optimization algorithm is concerned; we thus prefer the form given in (9b) due to its computational simplicity.

[8] Note that, for maximum efficiency, R^{-1} should be saved between function evaluations and reused for every new computation of $\hat{f}$ and s^2 required.

4 A review of global optimization strategies leveraging Kriging-based interpolation

The previous section reviewed the Kriging interpolation strategy which, based on a sparse set of observed function values $f^o(\mathbf{x}^i)$ for $i = 1, \ldots, N$, develops a function predictor $\hat{f}(\mathbf{x})$ and a model of the uncertainty $s^2(\mathbf{x})$ associated with this prediction for any given set of parameter values $\mathbf{x}$. Leveraging this Kriging model, an efficient search algorithm can now be developed for the derivative-free optimization algorithm summarized in Sect. 1.

The effectiveness of the various Kriging-based search strategies which one might propose may be tested by applying them repeatedly to simple test problems via the following procedure:

- a search function $J(\mathbf{x})$ is first developed based on a Kriging model fit to the existing function evaluations,
- a gradient-based search is used to minimize this (computationally inexpensive, smoothly-varying) search function,
- the function $f(\mathbf{x})$ is sampled at the point $\tilde{\mathbf{x}}$ which minimizes the search function,[9]
- the Kriging model is updated, and the search is repeated.

In the present work, we consider a scalar test problem with multiple minima, $f(x) = \sin(x) + x^2$, on the interval $x \in [-10, 10]$, and use four starting points to initialize the search $x = -10$, $x = -5.2$, $x = 6$, and $x = 10$. Ineffective search strategies will not converge to the global minimum of $f(x)$ in this test, and may not even converge to a local minimum. More effective search strategies converge to the global minimum following this approach, and the number of function evaluations required for convergence indicates the effectiveness of the search strategy used.

Perhaps the most "obvious" strategy to use in such problems is simply fitting a Kriging model to the known data, then searching the Kriging predictor itself, $J(\mathbf{x}) = \hat{f}(\mathbf{x})$, for its minimum value. This simple approach has been implemented in a variety of examples with reasonably good results (see Booker et al. 1999). However, as shown clearly in Fig. 6, this approach can easily break down. The Kriging predictor does not necessarily model the function accurately, and its minimization fails to guarantee convergence to even a local minimum of the function $f(\mathbf{x})$. This observed fact can be motivated informally by identifying the Kriging predictor as an *interpolating* function which only under extraordinary conditions predicts a function value significantly lower than all of the previously-computed function values; under ordinary conditions, a strategy of minimizing the predictor will thus often stall in the vicinity of the previously-evaluated points.

To avoid the shortcomings of a search defined solely by the minimization of the predictor, another strategy explored by Booker et al. (1999) is to evaluate the function at *two* points in parameter space during the search: one point chosen to minimize the predictor, and the other point chosen to maximize the predictor uncertainty. Such a heuristic provides a guarantee of global convergence, as the seach becomes dense in the parameter space as the total number of function evaluations, N, approaches infinity. However, this approach generally does not converge quickly as compared with the improved methods described below, as the extra search point has no component associated with the predictor, and is thus often evaluated in relatively "poor" regions of parameter space.

We are thus motivated to develop a more flexible strategy to explore *slightly away* from the minima of the predictor. To achieve this, consider the minimization of $J(\mathbf{x}) = \hat{f}(\mathbf{x}) - c \cdot s^2(\mathbf{x})$, where c is some constant (see Cox and John 1997 and Jones 2001). A search coordinated by

[9] For the moment, to focus our attention on the behavior of the search algorithm itself, no underlying grid is used to coordinate the search in order to keep function evaluations far apart.

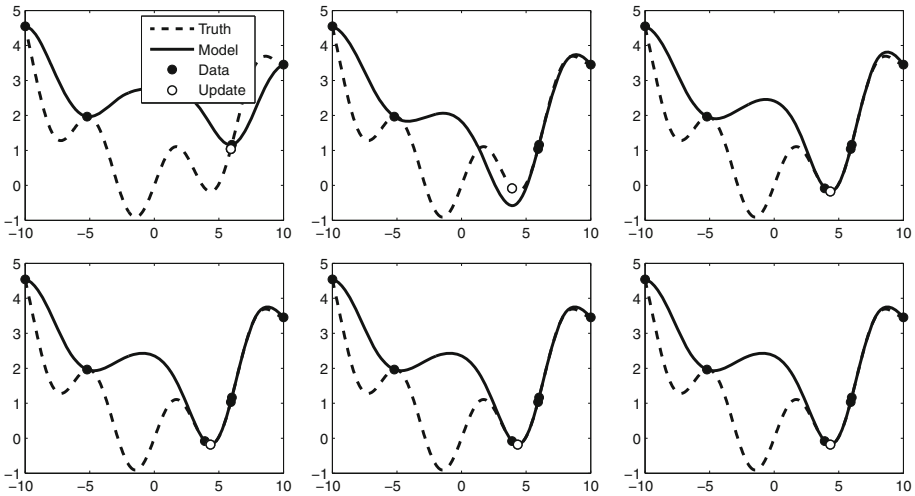

Fig. 6 Convergence of a search algorithm based on minimizing the Kriging predictor, $J(\mathbf{x}) = \hat{f}(\mathbf{x})$, at each iteration. This algorithm does not necessarily converge to even a local minimum, and in this example has stalled, far from the global minimum, after six iterations

this function will tend to explore regions of parameter space where both the predictor of the function value is relatively low *and* the uncertainty of this prediction in the Kriging model is relatively high. With this strategy, the search is driven to regions of higher uncertainty, with the $-c \cdot s^2(\mathbf{x})$ term in $J(\mathbf{x})$ tending to cause the algorithm to explore away from previously evaluated points. Additionally, minimizing $\hat{f}(\mathbf{x}) - c \cdot s^2(\mathbf{x})$ allows the algorithm to explore the vicinity of *multiple* local minima in successive iterations in order to determine, with an increasing degree of certainty, which local "bowl" in fact has the deepest minimum. The parameter c provides a natural means to "tune" the degree to which the search is driven to regions of higher uncertainty, with smaller values of c focusing the search more on refining the vicinity of the lowest function value(s) already found, and larger values of c focusing the search more on exploring regions of parameter space which are still relatively poorly sampled. This parameter may tuned based on knowledge of the function being minimized: if the function is suspected to have multiple minima, c can be made relatively large to ensure a more exploratory search, whereas if the function is suspected of having a single minimum, c can be made relatively small to ensure a more focused search in the vicinity of the CMP. For an appropriate intermediate value of c, the resulting algorithm is often quite effective at both global exploration and local refinement of the minimum, as illustrated in Fig. 7. The strategy of searching $J(\mathbf{x}) = \hat{f}(\mathbf{x}) - c \cdot s^2(\mathbf{x})$ also extends naturally to multiple dimensions, as illustrated for a two-dimensional problem in Fig. 5c. Note also that, in the spirit of Booker et al. (1999) [who effectively suggested, in the present notation, exploring based on both $c = 0$ and $c \to \infty$ at each search step], one can perform a search using multiple but finite values of c at each search step, returning a set of points designed to focus, to varying degrees, on the competing objectives of global exploration and local refinement. If at each search step k at least one point is included which minimizes $\hat{f}(\mathbf{x}) - c_k \cdot s^2(\mathbf{x})$ for a value of c_k which itself approaches ∞ as $k \to \infty$, then the search drives at least some new function evaluations sufficiently far from the existing points that the function evaluations eventually become dense over the feasible domain, thus guaranteeing global convergence. Thus, an $\hat{f}(\mathbf{x}) - c \cdot s^2(\mathbf{x})$ search, when used properly, can indeed be used in a globally convergent manner.

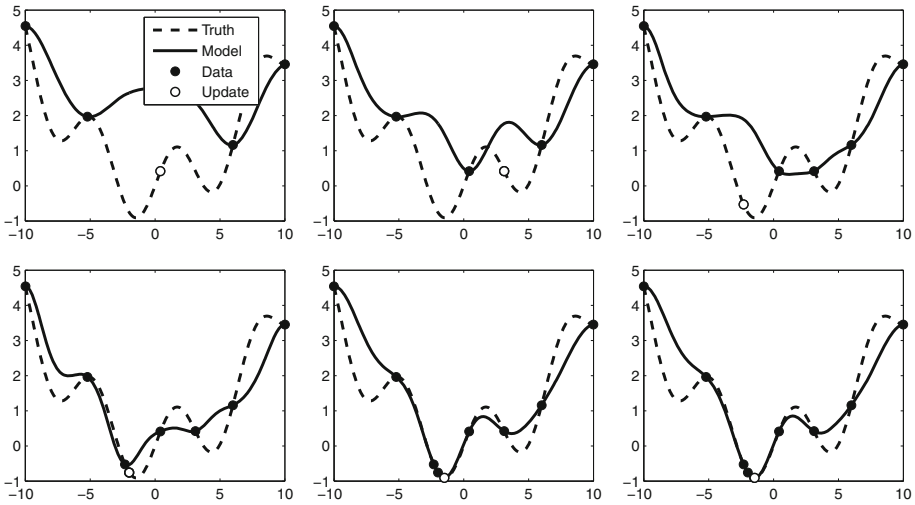

Fig. 7 Convergence of a search algorithm based on minimizing the search function $J(\mathbf{x}) = \hat{f}(\mathbf{x}) - c \cdot s^2(\mathbf{x})$ at each iteration, taking $c = 1$. Note that the global minimum is found after just a few iterations. However, global convergence is not guaranteed

Minimizing $J(\mathbf{x}) = \hat{f}(\mathbf{x}) - c \cdot s^2(\mathbf{x})$ is not the only strategy to take advantage of the estimate of the uncertainty of the predictor provided by the Kriging model. Another effective search strategy involves maximizing the probability of achieving a target level of improvement below the current CMP; this is called the *maximum likelihood of improvement* (*MLI*) approach (see Kushner 1964; Stuckman 1988; Perttunen 1991; Elder 1992; Mockus 1994). If the current CMP has a function value $f_{\min}$, then this search strategy seeks that $\mathbf{x}$ for which the probability of finding a function value $f(\mathbf{x})$ less than some prespecified target value f_{target} [that is, for which $f(\mathbf{x}) \leq f_{\text{target}} < f_{\min}$] is maximized in the Kriging model. If $f(\mathbf{x})$ is known to be a positive function, a typical target value in this approach is $f_{\text{target}} = f_{\min} - \delta(f_{\max} - f_{\min})$, where $\delta > 0$. Computationally, this involves an optimization nearly identical to the strategy previously discussed, where the minimum of the follow function gives the point returned by the Search step.

$$\breve{J}(\mathbf{x}) = \Phi((f_{\min} - \hat{f}(\mathbf{x}))/s^2(\mathbf{x}))$$

where Φ is the Normal cumulative distribution function.

As for the parameter c discussed in the previous paragraph, the parameter δ in this strategy tunes the degree to which the search is driven to regions of higher uncertainty, with smaller values of δ focusing the search more on refining the vicinity of the lowest function value(s) already found, and larger values of δ focusing the search more on exploring regions of parameter space which are still relatively poorly sampled. As seen in Fig. 8, the MLI search offers performance similar to the $\hat{f}(\mathbf{x}) - c \cdot s^2(\mathbf{x})$ method discussed previously. In contrast with the $\hat{f}(\mathbf{x}) - c \cdot s^2(\mathbf{x})$ approach, even for a fixed (finite) value of δ, the MLI approach eventually drives the function evaluations far enough away from existing points that the function evaluations eventually become dense over the feasible domain, thus guaranteeing global convergence. Thus, the MLI approach is inherently globally convergent.

Even more sophisticated search strategies can also be proposed, as reviewed elegantly by Jones (2001). However, potential performance gains become negligible as theoretical

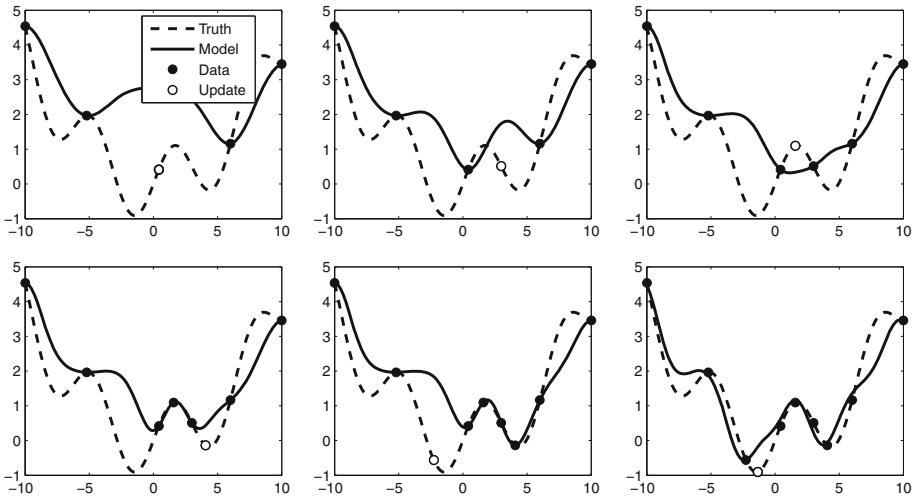

Fig. 8 MLI search with a target $T = 10\%$. Note convergence to global minimum, as well as exploratory nature of the search which guarantees global convergence

algorithm differences pale in comprison to judicious numerical application thereof. Minimizing either $J(\mathbf{x}) = \hat{f}(\mathbf{x}) - c \cdot s^2(\mathbf{x})$ or the MLI cost function give excellent results, with MLI maintaining an advantage for highly nonlocal optimizations.

Note that the selection of the weighting parameters c and δ is a challenging question of heuristics. Leveraging previous understanding of the cost function to be minimized can, in some cases, indicate whether a highly global or more local Search is desired. Under circumstances where no a priori knowledge of the cost function behavior is available, an effective approach is to perform multiple Searches, each for a different weight, ranging from a small to large weight. Then, the results are 'clustered' (see Jones 2001), returning local regions of interest in parameter space.

Since both the $J(\mathbf{x})$ search function and the MLI cost function $\breve{J}(\mathbf{x})$ are inexpensive to compute, continuous, and smooth, but in general have multiple minima (in fact, the behaviors of these two cost functions are extremely similar in a computational sense), an efficient gradient-based search, initialized from several well-selected points in parameter space, may be used to to minimize them. As the uncertainty $s^2(\mathbf{x})$ goes to zero at each sample point, $J(\mathbf{x})$ will tend to dip between each sample point. Analogously, the likelihood of improvement $\breve{J}(\mathbf{x})$ becomes zero at sampled points and increases far from sampled points. Thus, a search is initialized on $2n \cdot N$ total points forming a positive basis near (say, at a distance of $\rho_n/2$) to each of the N sample points, and each of these starting points is marched to a local minima of the search function using an efficient gradient-based search (which is constrained to remain within the feasible domain of $\mathbf{x}$ via the ubiquitous penalty or barrier function methods). The lowest point of the paths so generated will very likely be the global minima of the search function. For simplicity, the necessary gradients for this search may be computed via a simple second-order central finite difference scheme applied to the Kriging model.

5 Results

Putting everything together, we now develop and test what we identify as the *lattice based derivative-free optimization via global surrogates* (*LABDOGS*) algorithm. This algorithm

consists of an SMF-based optimization coordinated by uniform n-dimensional lattices (see Sect.2, and further discussion in Part I of this study) while leveraging a Kriging interpolant (see Sect. 3) to perform an efficient global search based on the search function $J(\mathbf{x}) = \hat{f}(\mathbf{x}) - c \cdot s^2(\mathbf{x})$ or the MLI cost function $\breve{J}(\mathbf{x})$ (see Sect. 4), implementing the clustering scheme described above if necessary. The full algorithm has been implemented in an efficient numerical code, and is tested in this section in $n = 2$ to $n = 8$ dimensions using the $\mathbb{Z}^n$, A_n, and E_8 lattices to coordinate the search, and is applied here to:

- randomly shifted quadratic bowls: $f_Q(\mathbf{x}) = (\mathbf{x} - \mathbf{x}^o)^T A(\mathbf{x} - \mathbf{x}^o)$,
- randomly shifted Rosenbrock functions: $f_R(\mathbf{x}) = \sum_{i=0}^{n-1}\{[1 - (x_i - x_i^o)]^2 + (-1)^n 500[(x_{i+1} - x_{i+1}^o) - (x_i - x_i^o)^2]^2\}$,
- the Branin function: $f_B(\mathbf{x}) = [1 - 2x_2 + 0.05\sin(4\pi x_2) - x_1]^2 + [x_2 - 0.5\sin(2\pi x_1)]^2$, and
- the "T_1" function: $f_M(\mathbf{x}) = \sin(5x_1) + \sin(5x_2) + 0.02[(5x_1 + 1.5)^2 + (5x_2 + 1.5)^2]$.

Note that the first two test functions are n-dimensional and have unique minima, whereas the last two test functions are 2-dimensional and have multiple minima.

5.1 SP applied to randomly-shifted quadratic bowls and randomly shifted Rosenbrock functions

To test the hypothesis that the efficiency of a pattern search is significantly affected by the packing efficiency and/or the nearest-neighbor distribution of the lattices which coordinate it, a large number of SP optimizations were first performed on randomly-shifted quadratic bowls to gather and compare statistical data on the performance of $\mathbb{Z}^n$-based, A_n-based, and E_8-based SP optimizations. The positive-definite matrices $A > 0$ and offsets $\mathbf{x}^o$ defining the quadratic bowls to be minimized, as well as the starting points used in the searches, were selected at random for every set of tests, and the initial $\mathbb{Z}^n$, A_n, and E_8 lattices were scaled such that the initial number of points per unit volume of parameter space was identical.

The $\mathbb{Z}^n$-based, A_n-based, and E_8-based SP algorithms were run from the same starting points on the same quadratic test functions to the same level of convergence. Note that several of the significant built-in acceleration features of the full LABDOGS code were in fact turned off for this baseline comparison. Most notably, complete polls were performed (that is, the poll steps were not terminated immediately upon finding a lower CMP), and no attempt was made to reuse previously-computed points when forming each successive poll set, or to orient optimally any given poll set. In fact, the angular distribution of the poll set around the CMP was fixed from one step to the next in these initial tests.

Two quantitative measures of the relative efficiency of the optimization algorithms to be tested are now defined. The metric p is defined as the *percentage of runs* in which the lattice-based algorithm requires fewer function evaluations than does the $\mathbb{Z}^n$-based algorithm to converge 99.99% of the way from the initial value of $J(\mathbf{x})$ to the optimal value of $J(\mathbf{x})$ [which, in these test problems, is easy to compute analytically]. The metric r is defined as the *ratio of the average number of function evaluations* required for the lattice-based algorithm to converge 99.99% of the way from the initial value of $J(\mathbf{x})$ to the optimal value of $J(\mathbf{x})$ divided by the average number of function evaluations needed for the $\mathbb{Z}^n$-based algorithm to converge the same amount.

Table 4 Performance comparison between the E_8-based SP algorithm and the $\mathbb{Z}^8$-based SP algorithm applied to randomly shifted quadratic bowls

n	8
p	90.65
r	0.1554

It is seen that the E_8-based SP algorithm outperformed the $\mathbb{Z}^8$-based SP algorithm 91% of the time, and on average required 17% as many function evaluations to reach the same level of convergence, thus offering nearly twice the performance of A_n

Table 5 Performance comparison between the A_n-based SP algorithm utilizing a Poll set of $2n$ points and the $\mathbb{Z}^n$-based SP algorithm utilizing a maximal positive basis

n	2	3	4	5	6	7	8
p	74.2	80.4	62.5	71.9	74.8	86.6	78.13
r	0.7477	0.7016	0.995	0.9438	0.8214	0.5825	0.7428

Both algorithms were applied to randomly shifted quadratic bowls for $n = 2$ to 5. The results demonstrate how the implementation of an efficient lattice increases the efficiency of the SP algorithm even when the most uniform Cartesian Poll set is used

The p and r measures described above (averaged over 5,000 runs for each value of n) were calculated in the case of the A_n lattice (for $n = 2$ to $n = 8$) and the E_8 lattice, and are reported in Tables 3 and 4. Note that values of p over 50% and values of r less than 1 indicate that, on average, the lattice-based SP algorithm outperforms the $\mathbb{Z}^n$-based SP algorithm, with p quantifying how often and r quantifying how much.

Note in Table 1 that the "best" lattice in $n = 2$ and $n = 3$, according to several standard metrics, is A_n; however, as the dimension of the problem increases, several other lattices become available, and that by $n = 8$ the E_8 lattice appears to be the best choice. This observation is consistent with the numerical results reported in Tables 3 and 4, which indicates that the A_n-based optimizations provided a consistent and substantial improvement over the $\mathbb{Z}^n$-based optimizations over the entire range $n = 2$ to 8, and that, in $n = 8$, the E_8-based optimization significantly outperformed the A_8-based optimization.

A natural question, when discussing the efficiency of the (very simple) SP algorithm alone, is questioning the performance difference between the Λ_n and Cartesian-based algorithms, but using a maximal positive basis in the Cartesian case, and a Poll set of size $2n$ in the case of the lattice. The uniformity of the Cartesian maximal basis avoids the primary disadvantage of the minimal Cartesian basis, and should return superior convergence rates as a result. The maximal SP comparison was performed as described above from the minimal base; the results are summarized in Table 5. Looking at Table 5, we see that the lattice-based algorithm still outperforms the Cartesian equivalent, though to a lesser degree. Note, however, that the SP algorithm alone is quite unsophisticated. In the SMF algorithm, a key point is the utilization of a minimal positive basis in the Poll step, to give the Search step the maximum computation time possible. The methodology revolves around the philosophy that the Search is absolutely crucial for convergence on nonconvex cost functions, and the assumption that the optimization becomes comparatively easy once the region of the global minimum has been located. For this reason, the reader should not assume that the SP results definitively indicate what the efficient Poll step in the full LABDOGS algorithm (see Booker et al. 1999; Marsden et al. 2007; Yang et al. 2010). As such, both the SMF and LABDOGS utilize a

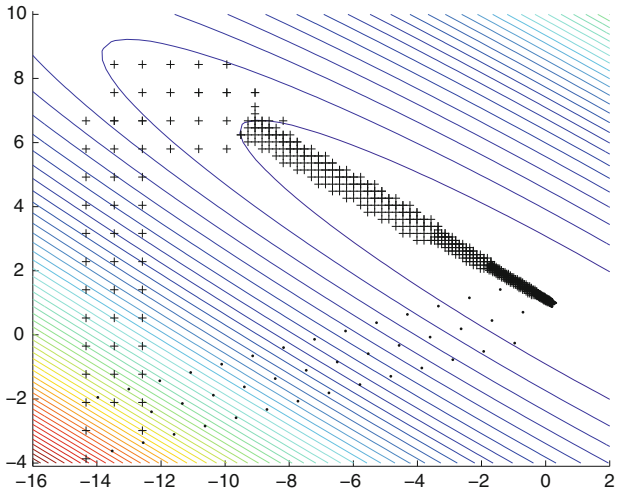

Fig. 9 Typical paths taken by the A_2-based SP algorithm (*dots*) and the $\mathbb{Z}^2$-based SP algorithm (+) on a randomly-shifted quadratic bowl

minimal positive basis whenever possible, though the implementation of a different Poll step is quite simple.

The efficiency factors p and r are normalized to specifically emphasis the performance differences between algorithms, rather simply comparing raw number of function evaluations required to convergence. The latter is most relevant when discussing small numbers of test optimizations, rather than the statistically significant number of optimizations performed to establish generalized performance above. However, the number of function evaluations required to convergence is ultimately the metric of interest in research application of these algorithms. Thus, note that the average number of function evaluations required in the tests summarized in Table 5 in $n = 2, 3, 5, 8$, respectively, were 44, 80, 181, 350. This establishes the relevance of the tests performed, as the level of convergence chosen is sufficiently fine to realistically evaluate algorithmic performance. Typically the algorithms required on the order of 5 to 10 mesh refinements to reach convergence; values that indicate a realistic performance evaluation.

The mechanism by which the lattice-based SP algorithms outperform the $\mathbb{Z}^n$-based SP algorithm on quadratic test problems is now examined in detail. As described previously, the $\mathbb{Z}^n$ minimal positive basis vectors are distributed with poor angular uniformity and can not be selected on nearest-neighbor lattice points. When the optimal descent direction is poorly approximated by these $n+1$ vectors (such as when the optimal descent direction is configured somewhere approximately midway between the oddball vector and one of the Cartesian unit vectors), the search path must "zig-zag" to move towards the actual minimum. If the local curvature of the function is small compared to the current lattice spacing, then the search algorithm must take several steps in a rather poor direction before it must eventually turn back down the "valley floor", as illustrated by the path of the $\mathbb{Z}^n$-based SP algorithm in Fig. 9. Once in this valley, the lattice spacing must be diminished such that each step of the "zig-zag" path required to proceed down the valley floor in fact decreases the function; this leads to otherwise unnecessary lattice refinement and thus very slow progress by the SP algorithm. This effect is exacerbated when the vectors of the poll set are of substantially different length, as the entire set of vectors must be scaled down until movement along the direction of the

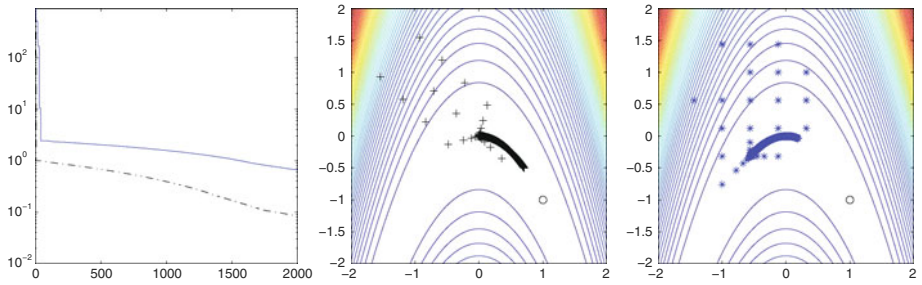

Fig. 10 A sample SP minimization comparing the A_n-based case (*dash-dot line* at *left* and *black* $+$ at *center*) with the $\mathbb{Z}^n$-based case (*solid line* at *left* and *blue* $*$ at *right*) on a randomly shifted Rosenbrock function. Note the superior convergence rate of the A_n-based approach (as illustrated in the convergence plot at *left*), resulting in further progress toward the minimum at $[1, -1]$ (as illustrated in the subfigures at *center* and *right*). (Color figure online)

longest poll vector during this zig-zagging motion still decreases the function. This leads to the poor convergence behavior demonstrated by the $\mathbb{Z}^n$-based SP algorithm along the narrow valley floor of the quadratic bowl indicated in Fig. 9. Of course, the present arguments are statistical in nature, and in specific cases either the A_n-based SP algorithm or the $\mathbb{Z}^n$-based SP algorithm will sometimes get "lucky" and converge remarkably quickly.

This stochastic aspect of the performance of the SP optimization is due to the random CMP at which the optimization is begun. When the algorithm is "lucky", the best descent direction is very well approximated by the discrete polling directions of the SP algorithm, leading to very efficient convergence. However, if the polling directions offer a poor approximation of the best descent direction, then convergence rates are reduced. This is a key aspect of understanding the efficacy of this class of algorithms, and is the motivation behind the presentation of statistically significant convergence metrics.

On average, it is clear that the optimal descent direction at any given iteration is more likely to be "far" from the poll vectors when the poll set is distributed with poor angular uniformity. This explains the superior convergence rate of the lattice-based SP algorithm compared to the Cartesian equivalent.

The A_n-based and $\mathbb{Z}^n$-based SP algorithms were also applied to a randomly-shifted Rosenbrock function in a similar fashion. Figure 10 demonstrates a typical case, indicating the respective rates of convergence of the two SP algorithms. The A_n-based SP algorithm demonstrates a substantially improved convergence rate compared to the $\mathbb{Z}^n$-based SP algorithm.

These results demonstrate that the efficiency of the SP portion of a pattern search can be substantially improved simply by implementing a more efficient lattice to discretize parameter space, for both maximal and minimal positive spanning sets.

5.2 LABDOGS applied to randomly shifted Rosenbrock functions

To test the hypothesis that the efficiency of the full LABDOGS algorithm is significantly affected by the choice of the lattices which coordinate it, a more demanding test than a quadratic bowl is required. We thus consider here the application of the full LABDOGS algorithm to randomly shifted Rosenbrock functions. The "valley" in which the minimum of the Rosenbrock function lies is narrow, curved, and relatively flat (that is, with a vanishing second derivative) along the bottom. This makes it a difficult test case for any SMF-like algorithm to approximate with a surrogate function of sufficient accuracy to be particularly useful along the valley floor, other than simply to indicate where the function evaluations are

Table 6 Performance comparison between the A_n-based LABDOGS algorithm and the $\mathbb{Z}^n$-based LABDOGS algorithm applied to randomly shifted Rosenbrock functions

n	2	3	4	5
$\bar{p}$	64.0	56.0	63.0	68.0
$\bar{r}$	0.651	0.699	0.773	0.758

For $n = 2$, it is seen that the A_n-based SP algorithm outperformed the $\mathbb{Z}^n$-based SP algorithm about 64% of the time, and on average converged to a function value 65% better using the same number of function evaluations

currently relatively sparse. In other words, both the search and poll components of the LABDOGS algorithm are put to the test when searching along the valley floor of the Rosenbrock function.

Two comparisons of the efficiencies of the A_n-based and $\mathbb{Z}^n$-based LABDOGS algorithms (using $c = 5$) applied to randomly shifted Rosenbrock functions are reported here. As in the SP tests described previously, the initial A_n and $\mathbb{Z}^n$ lattices were scaled appropriately so as to be of the same initial density.

Recall in the SP tests the metric p, which quantified *how often* the lattice-based method outperformed the Cartesian-based method, and the metric r, which quantifying *how much* the lattice-based method outperformed the Cartesian-based method. In this section, we use two similar metrics, $\bar{p}$ and $\bar{r}$, but now terminate each optimization after a particular number of iterations rather than after convergence to a given percentage of the (known) optimal solution. Specifically, the metric $\bar{p}$ is defined as the percentage of runs in which the A_n-based LABDOGS algorithm converged further than did the $\mathbb{Z}^n$-based LABDOGS algorithm after 300 function evaluations, whereas the metric $\bar{r}$ is defined as the ratio of the average function value to which the A_n-based LABDOGS algorithm converged after 300 function evaluations divided by the average function value to which the $\mathbb{Z}^n$-based LABDOGS algorithm converged after 300 function evaluations. The results for $n = 2$ to 5 (averaged over 200 runs for $n = 2$, 3, and 4, and 100 runs for $n = 5$) are reported in Table 6. Note that values of $\bar{p}$ over 50% and values of $\bar{r}$ less than 1 indicate that, on average, the lattice-based LABDOGS algorithm outperforms the $\mathbb{Z}^n$-based LABDOGS algorithm, with $\bar{p}$ quantifying how often and $\bar{r}$ quantifying how much. It is seen that the A_n-based LABDOGS algorithm consistently and significantly outperforms the $\mathbb{Z}^n$-based LABDOGS algorithm.

Figure 11 compares the convergence of the A_n-based and $\mathbb{Z}^n$-based LABDOGS algorithms on a representative realization of the Rosenbrock function in $n = 6$. The convergence of the two algorithms are similar in behavior during the first 20 iterations, during which they share a nearly identical search, with the differences between the two becoming more and more apparent as convergence is approached. Initially, the poll steps return much smaller improvements than the search steps. Once the surrogate model adequately represents the walls of the Rosenbrock function, thereby identifying the "valley floor", the search becomes less effective, and both algorithms rely more heavily on the polling algorithm to identify the minimum.

5.3 LABDOGS applied to Branin and T_1: demonstrating global exploration with local refinement

Thus far, only functions with unique minima have been explored. As the LABDOGS algorithm has the capability to locate and explore multiple local minima in an attempt to identify

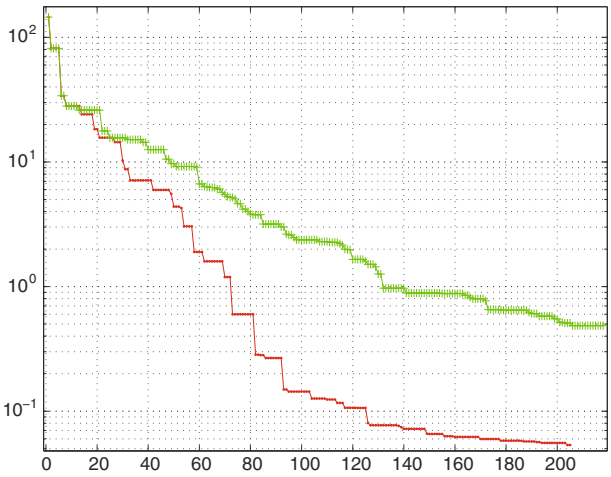

Fig. 11 Convergence of the checkers code using A_n (*red*) vs $\mathbb{Z}^n$ (*green*), on an $n = 6$ Rosenbrock function. (Color figure online)

and refine an estimate of the global minimum, some searches were performed on two test functions with multiple minima, Branin and T_1, to demonstrate this capability.

Note that exhaustive efficiency comparisons between algorithms were not performed on the Branin function. Such tests were avoided as the number of function evaluations required to convergence on a nonlocal optimization is heavily dependent on the efficacy of the Search algorithm chosen. The number of function evaluations required to attain convergence varys greatly depending on the initial conditions of the optimization. As such, averaging efficiency metrics over all runs performed would very easily give misleading results, as the variance of the results would render the averaged metrics untrustworthy. It is very challenging to establish computation superiority based on a small number of runs on a very deceptive (e.g. nonlocal) cost function. In the interest of methodically testing all components of the LABDOGS algorithm, we do not calculate averaged metrics on nonlocal functions.

On the interval $-2 < x < 2$, $-2 < y < 2$, the Branin function has five local minima. As seen in Fig. 12, with the search parameter $c = 2$, the LABDOGS algorithm does an excellent job of locating and exploring all of these local minima, eventually converging to an accurate estimate of the global minimum. With $c = 10,000$, the search tends to be more "space-filling", acting at each step to reduce the maximum uncertainty of the Kriging surrogate. It is clearly evident that, as the number of function evaluations gets large in the $c = 10,000$ case, this search will tend to explore nearly uniformly over the entire feasible domain. [In the limit that c is infinite, the function evaluations become dense as $N \to \infty$, thereby assuring global convergence.] However, for a small number of total function evaluations N [which should be the primary problem of interest if function evaluations are expensive!], the strategy with smaller c in fact identifies and refines the estimate of the global minimum point much sooner, as the case with large c wastes a lot of computational effort reducing the uncertainty of the surrogate in areas predicted to have poor function values. The implementation of the MLI Search function returns extremely similar results, and the question of tuning the Search weight remains.

Similar behavior can be seen for the T_1 test function in Fig. 13. Initially, the algorithm happens upon the local minimum in the lower-left corner of the feasible domain. With its

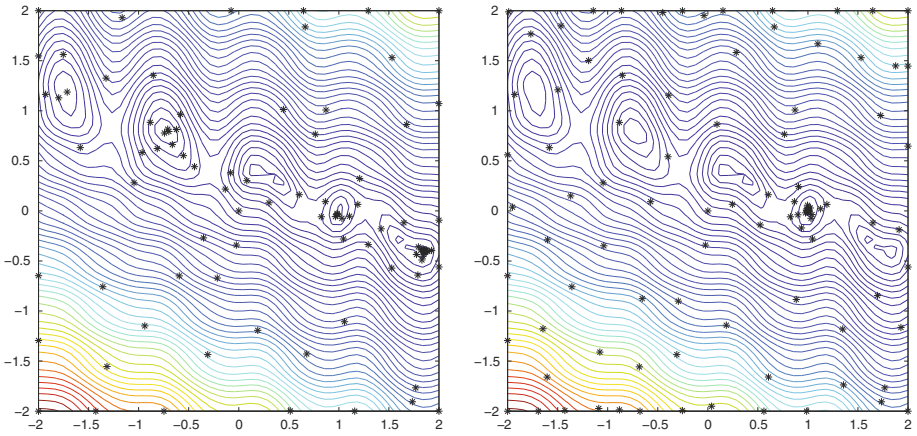

Fig. 12 Points evaluated by the LABDOGS algorithm when exploring the Branin function (with multiple minima), with (*left*) $c = 2$ and (*right*) $c = 10000$. Note the more "focused" sampling when c is small and the more "exploratory" sampling when c is large. These results indicate the need for either the selection of an appropriate weight, given a priori knowledge of the cost function, or the implementation of a clustering scheme where multiple searches are performed for varying values of c

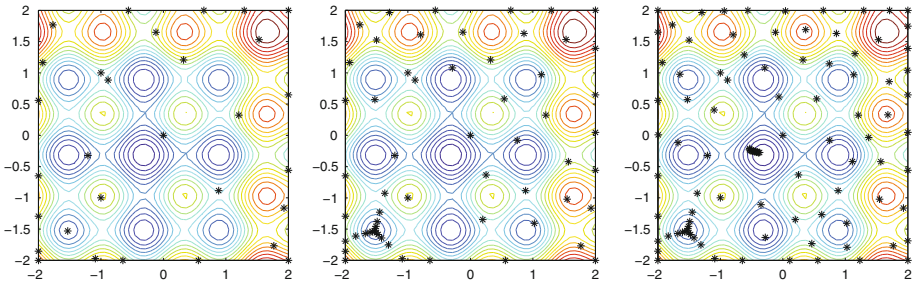

Fig. 13 Points evaluated by the LABDOGS algorithm when exploring the T_1 function (with multiple minima) with $c = 1000$ after (*left*) 30 function evaluations, (*center*) 60 function evaluations, and (*right*) 100 function evaluations. Note (after 30 function evaluations) that the LABDOGS algorithm initially identifies and converges to a local minimum near the lower-left corner. Ultimately (after 100 function evaluations), the LABDOGS algorithm successfully identifies a refined estimate of the global minimum

exploratory function evaluations, however, the algorithm ultimately identifies and refines its estimate of the global minimum.

The question that remains unaddressed by these computational results is how the weighting of the Search method should be chosen. The drawback of the majority of sophisticated Search strategies is that they rely upon some scalar weight, e.g. the parameter c above. The MLI Search method requires a 'target' value which only needs to be lower than the cost function's value at the CMP; minimizing $J(\mathbf{x})$ above requires some scalar weight $c > 0$.

The weighting parameter directly controls the exploratory nature of the Search step. As depicted above, when the parameter is small, the Search emphasises the region near the CMP; when the parameter is large, the Search emphasises the regions where the uncertainty is high. Thus, the selection of an appropriate weight is somewhat user-dependent. If a priori knowledge of the cost function is available, the user might have developed intution as to how nonlocal the behavior of the cost function may be. In the case that the cost function is

generally smooth and offers a local optimization, a small weight should be chosen. If the cost function is highly nonconvexl, a larger weight is more appropriate. Of course, if very little is understood about the cost function, there is no prescribed approach. In this case, as mentioned above, configuring the Search to return more than one point of interest is wise. For example, performing three Search optimizations, one for a relatively local, nonlocal, and highly nonlocal Search, and evaluating the cost function at each of the three points, would give an appropriately broad distribution of results. This methodology can of course be extended to much greater than three optimizations per Search step.

As is suggested in the interpolation-based optimization literature (e.g. Jones 2001), the most natural strategy for overcoming this limitation is to perform a Search for varying weights, then clustering the resulting minima to establish the regions of greatest interest. Of course, for testing purposes, such methodology is unnecessary to establish algorithm performance.

While these results indicate encouraging global exploration, further testing of the LABDOGS algorithm on nonconvex functions is certainly warranted, particularly in high-dimensional problems. The application of LABDOGS to challenging optimizations in research applications is currently underway.

6 Conclusions

The present work proposes a new algorithm, dubbed LABDOGS, for derivative-free optimization formed via the tight integration of

- the efficient SMF algorithm for a surrogate-based search coordinated by an underlying grid, in order to keep function evaluations far apart until convergence is approached,
- a uniform "grid" selected from those available in lattice theory (see Sect.2 and further discussion in Part I of this study) to coordinate such an optimization algorithm, in order to reduce the average quantization error of a grid of a given density and to better distribute the poll points during the poll step, and
- a highly effective search algorithm, leveraging a Kriging interpolant (see Sect. 3) to construct the search function $J(\mathbf{x}) = \hat{f}(\mathbf{x}) - c \cdot s^2(\mathbf{x})$ combining both the function predictor and a model of its associated uncertainty, in order to provide a flexible combination of global exploration and local refinement during the search (see Sect. 4).

The numerical results achieved via this algorithm (see Sect. 5) indicate effective convergence of the resulting algorithm on a range of benchmark optimization problems. Additionally, they reveal a clear advantage for using an efficient lattice derived from an n-dimensional sphere packing to coordinate such a search, rather than the heretofore default choice, $\mathbb{Z}^n$, which is simply untenable in light of the clear advantages of using alternative lattices, especially as the dimension of the cost function increases.

The flexible numerical code we have developed which implements this algorithm has been written from scratch, and each subroutine of the code has been scrutinized to maximize its overall efficiency for systems with expensive function evaluations. The LABDOGS code is currently be applied by various collaborators on challenging cost functions of great research interest; results will be reported in future publications.

Much interesting work remains to be done. The possible applications of such a derivative-free optimization code are quite broad. Natural extensions of the algorithm proposed herein include the implementation and testing of a variety of lattices, more sophisticated versions of Kriging interpolation, and appropriate penalities for online parameter tuning; such extensions are all well underway, and will be reported in future work.

Acknowledgments The authors would like to thank Dave Higdon, Haoxiang Luo, Alison Marsden, Sebastien Michelin, and Daniel Tartakovsky for helpful discussions related to this work.

References

Abramson, M.A., Audet, C., Dennis, J.E., Le Digabel, S.: OrthoMads: a deterministic Mads instance with orthogonal directions. SIAM J. Optim, **20**, 948–966 (2008)

Audet, C., Dennis, J.E. Jr.: Mesh adaptive direct search algorithms for constrained optimization. SIAM Optim. (2006)

Belitz, P: Applications on multi-dimensional sphere packings: derivative-free optimization. PhD dissertation, University of California, San Diego (2011)

Booker, A., Dennis, J.R., Frank, P., Serafini, D., Torczon, V., Trosset, M.: A rigorous framework for optimization of expensive functions by surrogates. Struct. Multidiscip. Optim. **17**, 1–13 (1999)

Conway, J.H., Sloane, N.J.A.: Sphere Packings, Lattices, and Groups. Springer, Berlin (1998)

Coope, I.D., Price, C.J.: On the convergence of grid-based methods for unconstrained optimization. SIAM J. Optim. **11**, 859–869 (2001)

Cox, D.D., John, S. : SDO: a statistical method for global optimization. In: Alexandrov, N., Hussaini, M.Y. (eds.) Multidisciplinary Design Optimization: State of the Art, pp. 315–329. SIAM, Philadelphia (1997)

Elder, J.F. IV: Global Rd optimization when probes are expensive: the GROPE algorithm. In: Proceedings of the 1992 IEEE International Conference on Systems, Man, and Cybernetics, vol. 1, pp. 577–582. Chicago (1992)

Ermoliev, Y.: Numerical Techniques for Stochastic Optimization. Springer, New York (1988)

Fejes Tóth, L.: Perfect distribution of points on a sphere. Periodica Mathematica Hungarica **1**, 25–33 (1971)

Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading (1989)

Isaaks, E.H., Srivastav, R.M.: An Introduction to Applied Geostatistics. Oxford University Press, Oxford (1989)

Jones, D.R.: A taxonomy of global optimization methods based on response surfaces. J. Global Optim. **21**, 345–383 (2001)

Kennedy, J., Eberhart, R: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, pp. 1942–1948. v.4. (1995)

Kirkpatrick, S., Gelatt, C.D. Jr.., Vecchi, M.P.: Optimization vi simulated annealing. Science **20**(4598), 671–680 (1984)

Krige, D.G.: A statistical approach to some mine valuations and allied problems at the Witwatersrand. Master's thesis of the University of Witwatersrand, South Africa (1951)

Kushner, H.J.: A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. J. Basic Eng. **86**, 97–106 (1964)

Marsden, A.L., Vignon-Clementel, I.E., Chan, F., Feinstein, J.A., Tyalor, C.A.: Trailing-edge noise reduction using derivative-free optimization and large-eddy simulation. J. Fluid Mech. **572**, 250–263 (2007)

Matheron, G.: Principles of geostatistics. Econ. Geol. **58**, 1246–1266 (1963)

McKay, M.D., Beckman, R.J., Conover, W.J.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Techonometrics **21**, 239–245 (1979)

Meschkowski, H.: Ungelöste und unlösbare Probleme der Geometrie. Bibliographisches Institut, Mannheim (1960)

Mockus, J.: Application of Bayesian approach to numerical methods of global and stochastic optimization. J. Global Optim. **4**, 347–365 (1994)

Nelder, J.A., Mead, R.: A simplex method for function minimization. Comput. J. **7**, 308–313 (1965)

Perttunen, C.: A computational geometric approach to feasible region division in constrained global optimization. In: Proceedings of the 1991 IEEE Conference on Systems, Man, and Cybernetics (1991)

Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. MIT Press, Cambridge (2006)

Spendley, W., Hext, G.R., Himsworth, F.R.: Sequential application of simplex designs in optimisation and evolutionary operation. Technometrics **4**, 441–461 (1962)

Stuckman, B.E.: A global search method for optimizing nonlinear systems. IEEE Trans. Syst. Man Cybern. **18**, 965–977 (1988)

Tammes, P.M.L.: On the origin of number and arrangement of the places of exit on the surface of pollen-grains. Recueil Des Travaux Botaniques néerlandais **27**, 1–84 (1930)

Thomson, J.J.: On the structure of the atom: an investigation of the stability and periods of oscillation of a number of corpuscles arranged at equal intervals around the circumference of a circle; with application of the results to the theory of atomic structure. Philos. Mag. Ser. 6 **7**(39), 237–265 (1904)
Torczon, V.: On the convergence of pattern search algorithms. SIAM J. Optim. **7**, 1–25 (1997)
Torn, A., Zilinskas, A.: Global Optimization. Springer, Berlin (1987)
Yang, W., Feinstein, J.A., Marsden, A.L.: Constrained optimization of an idealized Y-shaped baffle for the Fontan surgery at rest and exercise. CMAME **199**, 2135–2149 (2010)