Computer Physics Communications 233 (2018) 84–109

Contents lists available at ScienceDirect

Computer Physics Communications

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

An iterative procedure for finding locally and globally optimal arrangements of particles on the unit sphere[☆]

Wesley J.M. Ridgway [*], Alexei F. Cheviakov

Department of Mathematics and Statistics, University of Saskatchewan, Saskatoon, S7N 5E6, Canada

ARTICLE INFO

Article history:
Received 19 July 2017
Received in revised form 15 March 2018
Accepted 31 March 2018
Available online 30 May 2018

Keywords:
Local/global optimization
Optimal configuration
Pairwise energy
Narrow escape
Narrow capture
Coulomb energy
Steepest descent
Packing problem

ABSTRACT

The problem of determination of globally optimal arrangements of N pairwise-interacting particles arises in a variety of biological, physical, and chemical applications. At the same time, the important related question of finding all, or many, local minima of the corresponding energy functions, and the study of structure of these minima, has received relatively little attention.

A computational procedure is proposed to compute locally optimal and putative globally optimal arrangements of N particles constrained to a sphere. The procedure is able to handle a wide class of pairwise potentials, and can be generalized to other kinds of surfaces and interactions.

As computational examples, locally and globally energy-minimizing arrangements of particles on the unit sphere, interacting via the Coulombic, logarithmic, and inverse square law potentials, are computed. We present new results for the logarithmic potential consisting of 45 new local minima for $N \leq 65$ and two new global minima ($N = 19, 46$), as well as results for the inverse square law potential which has not previously been studied. We provide comprehensive tables of all minima found, and exclude saddle points. The algorithm can perform computations exceeding $N = 100$ with reasonable execution times.

Program summary

Program Title: EOPS 1.0 - Energy Optimizer for Particles on the Sphere
Program Files doi: `http://dx.doi.org/10.17632/cbn8jt2ffw.1`
Licensing provisions: GPLv3
Programming language: MATLAB 2015b, C++98, Maple
Nature of problem: Computation of locally and globally optimal arrangements of N particles on the sphere for different pairwise potentials. This constitutes a constrained local optimization problem with $2N - 3$ degrees of freedom.
Solution method: For N particles, the pairwise potential energy is minimized via steepest descent trajectory from a starting configuration generated from known putative $(N - 1)$-particle optimal configurations.
Restrictions: Spherical domain in $\mathbb{R}^3$ and pairwise potentials. The number of particles is limited by the computing power and memory of the machine.
Unusual features: The programs are executed from MATLAB scripts which call C++ and Maple procedures which perform the bulk of the computations.

1. Introduction

The general problem of finding optimal arrangements of particles on the boundary of a domain in $\mathbb{R}^n$ dates back over 100 years to the Thomson problem, which concerns finding the arrangement of N identically charged particles on the surface of a sphere in $\mathbb{R}^3$ that minimizes the Coulombic energy. This problem arose in J.J. Thomson's early "plum pudding" model of the atom in which the electrons are point charges that are suspended in a "jellium" of positive charge. Similar problems now arise in molecular biophysics and material science.

[☆] This paper and its associated computer program are available via the Computer Physics Communication homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).
[*] Corresponding author.
E-mail addresses: wjr704@mail.usask.ca (W.J.M. Ridgway), cheviakov@math.usask.ca (A.F. Cheviakov).

https://doi.org/10.1016/j.cpc.2018.03.029
0010-4655/© 2018 Elsevier B.V. All rights reserved.

An *optimal configuration* is defined as follows. Given a domain $\mathcal{D}$ in $\mathbb{R}^n$, $n \geq 2$, an arrangement of N identical particles is said to be an optimal configuration when the set of particle coordinates $\{\mathbf{x}_i\}_{i=1}^N$ minimizes the pairwise potential,

$$\mathcal{H}(\mathbf{x}_1, \ldots, \mathbf{x}_N) = \sum_{i<j}^{N} h(|\mathbf{x}_i - \mathbf{x}_j|), \tag{1.1}$$

under the constraint $\mathbf{x}_i \in \partial\mathcal{D}$ where h is a pairwise energy associated with a single pair of particles. One obtains the Thomson problem when h is the Coulombic potential

$$h(d) = \frac{1}{d}. \tag{1.2}$$

Finding optimal configurations constitutes a constrained local optimization problem in which the objective function is given by (1.1).

The problem of distributing points on the boundary of a domain has applications in the *narrow escape problem* in biophysics. In this problem, a Brownian particle diffuses inside of a bounded domain with a reflecting boundary except for N small absorbing "traps", or windows. The problem consists in finding the mean first passage time (MFPT) of Brownian particles through the traps. Narrow escape problems can be used to model the motion of Brownian particles (proteins, ions, etc.) which must exit a confining domain in order to accomplish a biological function. Some examples include chemical reactions in microdomains (such as synapses and microvesicles) [1], the time required for diffusive particles inside a biological cell to react with proteins on the cell membrane [2], the dynamics of receptors undergoing Brownian motion on the cell membrane [3], and virus transport inside the cell nucleus [4]. A specific example of a narrow escape problem for the sphere is a model of a biological microstructure known as a dendritic spine (see [1] and references therein). These structures are found in neurons as the postsynaptic part of a synapse and consist of a head at the end of a long neck. The mean time for a calcium ion undergoing Brownian motion to escape the spine head (i.e. the MFPT) is an important quantity related to synaptic plasticity.

Multi-term asymptotic expressions for the MFPT have recently been obtained for various domains using the method of matched asymptotic expansions [5–7]. In the case where the domain is a sphere in $\mathbb{R}^3$ with N small absorbing spherical caps on the boundary, it has been shown [5,8] that finding arrangements of traps that minimize the average MFPT requires minimization of an energy-like function given by (1.1) over the surface of the sphere where

$$h(d) = \frac{1}{d} - \frac{1}{2}\log d - \frac{1}{2}\log(2+d). \tag{1.3}$$

A related problem, known as the *narrow capture problem*, involves finding the MFPT for a Brownian particle diffusing inside a domain with a reflecting boundary but small absorbing interior targets. This problem also has applications in molecular biophysics (see [9] and references therein). The case of a general domain with a single interior target consisting of absorbing spherical caps on an otherwise reflecting boundary has been studied recently [10,11]. In particular, it was shown that when the target is spherical with N absorbing circular pores, the MFPT is minimized when the target is at the center and the arrangement of pores minimizes a pairwise energy-like function in which the pairwise energy is similar to (1.3)

$$h(d) = \frac{1}{d} + \frac{1}{2}\log\left(\frac{d}{2+d}\right). \tag{1.4}$$

Computation of optimal configurations also relates to packing problems. The best-packing problem for the sphere consists of finding the most efficient way of packing circles (spherical caps) onto the surface of a sphere, or equivalently maximizing the smallest radius of N circles on the sphere. This problem is also referred to as the Tammes problem after the Dutch botanist who studied the arrangement of exit places on grains of pollen [12]. Recently this problem has been solved exactly for $N = 13, 14$ [13], and exact solutions are now known for all N up to 14. The arrangements of spherical caps minimize an extremely short-range energy function (see [14] and references therein) which is given by the limiting case

$$h(d) = \frac{1}{d^m}, \qquad m \to \infty. \tag{1.5}$$

In the low-temperature limit, the geometry of a crystalline material is determined by the lowest energy configuration or "ground-state" of the system. The ground state configurations for planar crystals contain no defects in their structure, however, in general this is impossible for non-planar crystals due to the Euler theorem of topology. The geometry and arrangements of defects in such materials play a role in determining their electrical and mechanical properties. As a well-known example of a crystal with non-planar geometry, consider carbon nanotubes (CNTs). The chirality of a CNT determines whether it is eclectically conducting, insulating, or semiconducting.

Spherical structures occur frequently in material science, notably the class of carbon fullerene molecules discovered in 1985 [15]. Spherical structures also occur in colloidosomes which are shells consisting of colloidal particles surrounding a liquid center. The arrangements of charged colloids correspond to solutions of the Thomson problem[16]. The Thomson problem also serves as a reasonable model of multielectron bubbles in liquid helium [16,17]. These bubbles are formed above the liquid surface when an electrode is submerged in the liquid and the electric field strength is increased beyond a critical value. This causes electrons that are initially outside the surface to enter the liquid via formation of multielectron bubbles. The bubbles contain approximately 10^5 to 10^8 electrons spread across the surface and the bubble radius is between 10 μm and 100 μm. The inter-electron spacing is usually at least 0.2 μm and therefore the electrons can be considered classical particles distributed such that the Coulomb energy is minimized [17–19]. Furthermore, it has been shown that a spherical bubble is energetically stable against perturbations under an appropriately pressure [20]. Thus multielectron bubbles indeed closely resemble Thomson's original problem albeit in a different context.

The Thomson problem today remains interesting both as a mathematical problem but also as a benchmark for testing optimization software. A comprehensive list of putative globally energy-minimizing designs has been produced for all N up to 132 and some selected N up to 282 [21]. Local minima are not as well known, although many putative local minima have been discovered up to $N = 112$ [22] and up to $N = 150$ for a few selected N [23]. Putative global minima for particles interacting via the logarithmic potential have also been studied in [24], and in d-dimensional space in [25]. Local minima for the logarithmic potential have not been studied until now. The logarithmic potential is believed to model, for example, interacting vortex defects in thin-film superconductors [26].

Various global optimization algorithms have been and are being developed that aim at finding global extrema in general or specific classes of optimization problems. The well-known general algorithms include, for example, genetic algorithms and simulated annealing [27–30].

By contrast, systematic computations of not only global but also *multiple local extrema* in realistic physical settings have received significantly less attention. A function can have a very large number of local minima which are not global minima. Even for standard benchmarking optimization problems, the numbers and structure of such local minima is not well-known. A common approach in the works where local and global minima were sought, including, for example, the study of particles on a sphere interacting through the Coulombic and logarithmic potential in three dimensions [22,24], performed local optimizations based on an initial "guess" or starting configuration to begin the iterations. Many previous works have used randomly generated starting configurations [22,24]. It has been demonstrated that a large number of starting configurations is required even for relatively small N since many local minima have small basins of attraction. In a recent paper [23], a unique algorithm was employed to systematically find starting configurations for local optimization. Transition states, defined as a saddle points in which the Hessian matrix of the energy function has exactly one negative eigenvalue, were optimized by taking a small step in the two lower-energy directions and then using a limited memory BFGS algorithm [31] to relax to a local minimum. Transition states were identified using known local minima which allows the process to be repeated until no new local minima are identified. The numbers of local minima identified with this method are significantly higher than the numbers predicted by previous studies [32].

The method of [23] allows for paths to be constructed between local minima on the energy surface which pass through transition states. This is quite powerful as it gives the energy required for configurations to convert between two given local minima which is of interest when, for example, studying chemical reaction rates. They have also demonstrated that relaxing to the global minimum is typically easier than relaxing to other local minima.

In this work we present a different, iterative numerical technique for finding locally and globally energy-minimizing arrangements of particles on the sphere. For N particles constrained to the unit sphere, the problem involves $2N$ spherical coordinates, and $2N-3$ degrees of freedom. The technique, described in Section 2, allows the construction of N-particle starting configurations from known $(N-1)$-particle local and global minima. The starting configurations are optimized using a modified steepest descent method implemented in C++. We also present a robust method for reliably identifying geometrically equivalent configurations using a coordinate-independent invariant. Saddle points are identified and excluded using the eigenvalues of the Hessian matrix. With the exception of the optimization algorithm, the technique is implemented in MATLAB. In Section 3 we apply the technique to the familiar Coulombic potential as well as the logarithmic and inverse square law potentials given by

$$h(d) = \frac{1}{d^2}, \tag{1.6}$$

$$h(d) = -\log d \tag{1.7}$$

respectively. The inverse square law potential is easier to work with than the Coulomb case in the sense that the forces between particles decay faster with distance and are thus more short range. A similar argument suggests that the logarithmic potential is more difficult due to the slow decay of the forces. However, it has been noted in [23] (and references therein) that the number of local minima for long range potentials is smaller than for short range potentials. We perform sample computations up to $N = 65$ particles to be consistent with previous literature (see [14,22,24]) but computations exceeding $N = 100$ are certainly possible within a reasonable execution time.

We find that the technique reproduces most (and possibly all) previous local and global minima for the Coulomb potential. Direct comparisons are difficult due to the presence of saddle points in previous literature. We also find two new global minima for the logarithmic potential ($N = 19$ and $N = 46$). All local minima for the logarithmic potential are new. The inverse square law has not been studied and therefore all putative minima for this potential are new as well.

2. Systematic computation of optimal arrangements

Optimal configurations for the sphere in $\mathbb{R}^3$ are those that minimize (1.1) with the constraint $|\mathbf{x}_i| = 1$. A systematic numerical routine was developed to find locally and globally optimal arrangements. The algorithm consists of the following general steps:

1. From a known set of optimal configurations of N particles, $M(N)$, generate a set of starting configurations for $N+1$ particles, $K_1(N+1)$ which are non-optimal and not geometrically equivalent through rotational and reflectional symmetries.
2. Perform local optimization on each configuration in $K_1(N+1)$ to obtain the set of optimal configurations $K_2(N+1)$.
3. Many of the configurations in $K_2(N+1)$ are geometrically equivalent. These *redundant* configurations are removed to obtain the set of optimal $(N+1)$-particle configurations, $M(N+1)$.
4. Let $N \to N+1$ and repeat steps 1 through 3, up to some specified N_{max}.
5. Remove all non-optimal saddle-point configurations.

There are four central parts in the above steps:

- A local optimization algorithm which performs the energy minimization given a starting configuration.
- An algorithm for generating starting configurations.
- An algorithm for the identification of equivalent configurations.
- A method to remove the non-optimal saddle-point configurations that occasionally result from optimization.

The remainder of this section is devoted to describing these parts in more detail.

2.1. The local optimization algorithm

A dynamical system is constructed in which the particles interact through forces defined as,

$$\mathbf{F}_i = \sum_{j\neq i}^{N} \mathbf{F}_{ij} = -\nabla_i \mathcal{H}(\mathbf{x}_1, \dots, \mathbf{x}_N), \tag{2.8}$$

where $\mathbf{F}_i$ is the total force acting on particle i, $\mathbf{F}_{ij}$ is the force acting on particle i from particle j, and ∇_i is the gradient operator with respect to the cartesian coordinates of particle i.

The particle radial distances are fixed; $|\mathbf{x}_i| = 1$, and thus only the component of $\mathbf{F}_i$ tangential to the surface, $\mathbf{F}_i^\tau$ is relevant to the interaction. The particles are allowed to move in the direction of the tangential force with the constraint that each must remain on the surface of the unit sphere. For a fixed radial coordinate in (1.1), the gradient in spherical coordinates of particle i will point in the direction of $\mathbf{F}_i^\tau$. It is therefore more natural to work with tangential forces.

A modified steepest-descent increment is used:

$$\mathbf{x}_i \to \frac{\mathbf{x}_i + \gamma \mathbf{F}_i^\tau}{\left|\mathbf{x}_i + \gamma \mathbf{F}_i^\tau\right|} \tag{2.9}$$

where γ is a proportionality constant. We define a characteristic distance, a_0, as the radius of a circle which encloses the average surface area on the unit sphere per particle, $4\pi/N$:

$$\pi a_0^2 = \frac{4\pi}{N}. \tag{2.10}$$

For large N, the typical distance between two particles is $d_0 \sim 2a_0$. Define F_{init}^τ as the largest tangential force in the starting configuration. We then make the choice

$$\gamma = \frac{\beta a_0}{F_{init}^\tau}, \tag{2.11}$$

where β is a constant specified by the user to maximize convergence, hereafter referred to as the weight parameter. The idea is that the position increments should be proportional to the characteristic distance. The algorithm is designed to be robust when used with judiciously chosen starting configurations where the particles start relatively close to a local minimum.

The following iteration scheme was implemented in C++:

1. Initialize the particle positions, $\mathbf{x}_i$ to a specified starting configuration (see Section 2.3).
2. Compute the characteristic distance a_0 (2.10).
3. Compute all tangential force vectors

$$\mathbf{F}_i^\tau = \sum_{j\neq i}^{N} \left[\mathbf{F}_{ij} - (\mathbf{x}_i \mathbf{F}_{ij})\mathbf{x}_i\right]. \tag{2.12}$$

   The calculation of tangential forces can also be done directly using the total force on a particle without the need of a sum over pairwise forces. However, this leads to cancellation of large forces because the normal components dominate the tangential components when the tangential forces are small. The largest of the tangential forces, $F_{init}^\tau = \max_i(|\mathbf{F}_i^\tau|)$, is computed on the first iteration only.
4. Increment particle positions using (2.9).
5. On every 100th iteration compute the maximum tangential force for the current configuration $F_{max}^\tau = \max_i(|\mathbf{F}_i^\tau|)$. If $F_{max}^\tau/F_{init}^\tau < 10^{-15}$ or the current number of iterations exceeds the maximum number, stop iterating, otherwise go to step 3.

This routine works effectively when a suitable starting configuration is used. However, it performs poorly with randomly generated starting configurations because the stopping condition is met relatively soon when the initial forces are large.

2.2. The identification of different configurations

The configurations are defined up to rotations and reflections and are therefore not unique. For N particles, there are $2N$ angular coordinates, and $2N - 3$ degrees of freedom. A challenge then arises in distinguishing between arrangements of particles that are rotated or reflected with respect to each other.

It is natural to use an invariant (coordinate independent) quantity to distinguish geometrically different configurations. One has several choices such as energy, coordination number, etc. Two algorithms are presented. The first uses the set of all pairwise distances between particle pairs. The second uses coordination numbers as well as suitable pairwise energies. A significant difference between the two is the size of the vector assigned to each configuration. A MATLAB clustering routine requiring the Statistics Toolbox is used in both cases to group the configurations into bins of geometrically equivalent designs.

In the algorithms below, $\mathbf{x}_i^{(k)}$ denotes the coordinates of the i^{th} particle in the k^{th} configuration.

2.2.1. Algorithm 1: pairwise distances

The following steps are performed to extract the non-redundant configurations.

1. Compute all scalar pairwise distances $d_{ij}^{(k)} = |\mathbf{x}_i^{(k)} - \mathbf{x}_j^{(k)}|$, $i < j \leq N$ (in parallel) and store them in a vector. Each configuration now has a vector of pairwise distances, which are denoted $\mathbf{d}^{(k)}$.

2. For each configuration, sort elements of $\mathbf{d}^{(k)}$ in ascending numerical order. Also normalize the elements in each array such that $\max_k \left(d_{ij}^{(k)} \right) = 1$ for a given i and j. Denote the resulting normalized vector by $\tilde{\mathbf{d}}^{(k)}$.
3. A tolerance is required for clustering in the next step. A suitable choice of tolerance is based on the understanding of the relationship between the elements in $\tilde{\mathbf{d}}^{(k)}$ as well as their relationship with N (see Remark 2.1). The tolerance can either be specified directly or one can specify the parameter *tol* (see Appendix B) in which case one of two choices can be made for the tolerance:

$$\delta_1 = \frac{|tol|}{\max_k \|\tilde{\mathbf{d}}^{(k)}\|_{L^2}},$$
$$\delta_2 = |tol| \times \max_k \|\tilde{\mathbf{d}}^{(k)}\|_{L^2} \tag{2.13}$$

4. Built-in MATLAB clustering functions are used to put the vectors $\tilde{\mathbf{d}}^{(k)}$ into clusters. The clustering is based on the Euclidean distance between vectors. The tolerance specified in the previous step is used.

Given a suitable tolerance, all configurations within one of the resulting clusters are considered equivalent.

Remark 2.1. The relationship between the elements in $\tilde{\mathbf{d}}^{(k)}$ and N is not well understood. The choices of δ_1 and δ_2 were made heuristically. Values of *tol* are chosen such that the tolerance δ falls within a suitable range determined through experimentation.

2.2.2. *Algorithm 2: pairwise energies*

1. Cluster the configurations according to coordination numbers such that configurations with different defect structures are placed in separate clusters. Let there be m such clusters.
2. For each of the configurations in a given cluster, compute the pairwise energy in (1.1) with the choice

$$h(|\mathbf{x}_i - \mathbf{x}_j|) = \frac{1}{|\mathbf{x}_i - \mathbf{x}_j|^n} \tag{2.14}$$

   where n is user specified and can be an array (see Appendix B). Note that the choice is not required to be the same as the potential used in the optimization algorithm. Thus each configuration has a vector of pairwise energies, each element of which corresponds to a choice of n. Denote the kth vector in a given cluster as $\mathbf{E}^{(k)}$.
3. For each of the m clusters, sort each energy vector in ascending numerical order. Then normalize each element following the same procedure as in algorithm 1. The resulting arrays are denoted $\tilde{\mathbf{E}}^{(k)}$.
4. For each of the m clusters found in 1), cluster the configurations within each based on energy. As in the first algorithm, the tolerance, δ can either be set explicitly by the user or 'automatically' by the program. When set automatically, the tolerance is

$$\delta = |tol| \times \max_k \|\tilde{\mathbf{E}}^{(k)}\|_{L^2} \tag{2.15}$$

   where *tol* is again a parameter specified by the user (see Appendix B). All configurations within an energy cluster will be equivalent given a suitable tolerance.

As with algorithm 1, suitable tolerances are chosen through experimentation.

2.3. *The starting configurations*

The local optimization routine requires an initial configuration of particles. Previous work has focused on using many trials with random starting configurations. This quickly becomes computationally expensive as the number of local minima is believed to increase exponentially [23,32] which requires the number of random trials to increase quickly as well. A unique algorithm for generating starting configurations was developed that significantly reduces the number of required optimizations.

The following steps are performed to generate a starting configuration to use with the local optimization routine described in Section 2.1.

1. Perform Delaunay Triangulation on the particles of a known local minimum and compute the convex hull.
2. Compute the center of mass for each of the resulting triangular facets. Let there be m such centers.
3. Create m new configurations each containing the original $N-1$ particles as well as a particle at one of the m triangle middles.
4. Many of the resulting N-particle configurations will be equivalent due to the symmetry of the original $(N-1)$-particle optimal configuration. Remove the redundant configurations using one of the algorithms described in Section 2.2.

An illustration of the construction of starting configurations is shown in Fig. 1 for $N = 26$ from the $N = 25$ global minimum.

2.4. *Removal of putative saddle-point configurations*

A routine for removing non-optimal configurations is needed as the local optimization algorithm will occasionally produce saddle-point configurations. A Maple program was developed that computes the gradient and Hessian matrix of Eq. (1.1) for a given choice of the function $h(|\mathbf{x}_i - \mathbf{x}_j|)$ and given a set of coordinates, $\mathbf{x_i}$, produced by the local optimization routine.

Let ϕ_i and θ_i denote the azimuthal and polar coordinates of particles i. Upon transforming the pairwise distances between particles into spherical coordinates with the constraint $|\mathbf{x}_i| = 1$, one obtains

$$|\mathbf{x}_i - \mathbf{x}_j| = \left(2 - 2\sin\theta_i \cos\phi_i \sin\theta_j \cos\phi_j - 2\sin\theta_i \sin\theta_j \sin\phi_i \sin\phi_j - 2\cos\theta_i \cos\theta_j\right)^{1/2}, \tag{2.16}$$

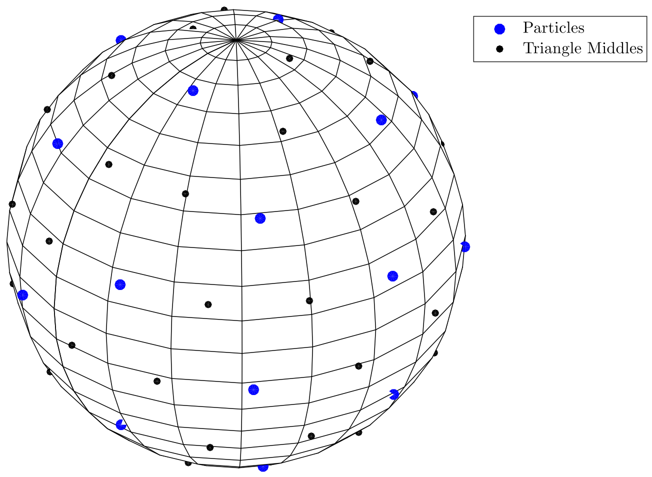


Fig. 1. Globally optimal arrangement of 25 particles for the Coulomb potential with all triangle middles shown.

which can be substituted into Eq. (1.1) to express the energy in terms of the spherical angles. Calculating the gradient and Hessian matrix is computationally easier in spherical coordinates due to the simple form of the constraint. The gradient vector, $\mathbf{f}$, and Hessian matrix, $\mathbf{M}$ are then computed with respect to the $2N-3$ degrees of freedom. The angles θ_i, ϕ_i, and θ_j for some arbitrary choice of i and j are fixed in the calculation which removes the rotation and reflection symmetries that cause the Hessian to be semi-definite. The result is evaluated using numerical values from the computed configurations. The $2N-3$ eigenvalues, λ_k, of $\mathbf{M}$ are computed. All eigenvalues must be positive for the configuration to be a local minimum. In some cases the eigenvalues are very small which can cause them to be slightly negative when computed numerically even for actual local minima. To deal with this difficulty, two different choices for the fixed angles are used. Computationally this is done by circular shifting the numerical values of θ and ϕ that are substituted into the Hessian.

A MATLAB routine then removes all configurations that fail to satisfy the following requirements:

$$\|\mathbf{f}\|_\infty < 10^{-6}, \qquad \min_k \lambda_k > -10^{-14}. \tag{2.17}$$

The first requirement may seem fairly loose but recall that the local optimization already produces configurations with very small tangential forces. The requirement is a safety that can identify when the optimization algorithm does not converge or when other unexpected errors occur.

3. Run examples

The general algorithm for finding optimal configurations on the sphere has been introduced. Now we focus on applications to three pairwise potentials: the Coulomb potential, the inverse square law potential, and the logarithmic potential given by (1.5)–(1.7).

Consider the case of $N = 4$ particles for which the optimal configuration for each potential is an inscribed tetrahedron, in fact this configuration is optimal for all completely monotonic potentials (see [33] for a discussion on universally optimal configurations). This configuration can be used as a starting point to compute local minima for higher N. A MATLAB program implementing the complete procedure that computes local minima for $N \leq 65$ has been provided. It is straightforward to modify the program to work with other pairwise potentials.

The following sequence is executed in a MATLAB script called `Opt_Procedure.m`.

1. Set $N = 4$.
2. Generate starting configurations for $N + 1$ particles using the algorithm in Section 2.3. Remove the redundant configurations using the procedure described in Section 2.2.
3. Perform local optimization on these starting configurations in parallel using the chosen pairwise potential. The optimization process is by far the main bottleneck especially for large N. Therefore it is important that they are executed in parallel and that redundant configurations are removed correctly in the previous step. Many of the resulting optimal configurations will be equivalent due to rotation and reflection symmetries. Remove the redundant configurations again using the procedure of 2.2.
4. Save the current results. Set $N \to N + 1$ and if $N = 65$ stop, otherwise go to step 2).

3.1. Modifying the program

The script is a general procedure that can be adapted to different potentials in a straight-forward manner. The main program listed in Appendix D is written for the Coulomb potential and uses algorithm 1 in Section 2.2.1. Detailed information on how to modify the program for different situations is given in the Appendix. We expect that the most frequent changes will be the maximum particle number, the flag to keep temporary files, and the weight parameter (all described in the appendices). These values can easily be changed by modifying the variables *N_END*, *keep_temp_files*, and *weight* in `Opt_procedure.m`. Adapting the program to a different potential is discussed in the appendices.

3.2. Parameter choices

There are number of tolerances which one must consider. In general, careful testing may be required to find appropriate values as the results of choosing poor values are not always apparent for small values of N which are relatively easy.

The parameter β in Eq. (2.11) is an important parameter which is chosen based on several trials performed for low numbers of particles ($N \leq 30$). Choices that are too large lead to poor results at best and non-convergent results in extreme cases. Smaller values can be chosen but too small a value is not necessarily better as it will lead to slow convergence and in some cases a lack of numerical precision during force calculations. The choices for each potential are as follows:

- Coulomb: $\beta = 0.5$
- Inverse square law: $\beta = 0.4$
- Logarithmic: $\beta = 0.5$

The tolerance in algorithms 1 and 2 must also be chosen with care. We remark that typically it is more difficult to distinguish between different starting configurations as opposed to local minima. This is due to the fact that of the N particles in a starting configuration, $N - 1$ of them have identical coordinates. These parameter choices are explored for the Coulomb potential and the choice giving the best results is used for the remaining potentials.

3.2.1. Parameter choices for Algorithm 1

Algorithm 1 in Section 2.2 is implemented within the supplied code. Several different tolerances are used for this algorithm to illustrate the importance of making a suitable choice. We remark that when the pairwise distance vectors $\tilde{\mathbf{x}}^{(k_1)}$ and $\tilde{\mathbf{x}}^{(k_2)}$ corresponding to two equivalent configurations are compared in the L^2-norm, $\|\tilde{\mathbf{x}}^{(k_1)} - \tilde{\mathbf{x}}^{(k_2)}\|_{L^2}$, the difference is typically less than 10^{-9} and appears to be roughly independent of N. A loose lower bound on the choice of *tol* is therefore 10^{-9}. Dissimilar configurations differ by at least 10^{-1} for small N but the difference slowly becomes smaller, down to about 10^{-2} around $N = 65$. Thus the first tolerance in Eq. (2.13) is a more natural choice since it decreases with N.

The program is implemented for three choices of the tolerance:

- δ_1 in Eq. (2.13) with the choice $tol = 10^{-3}$
- δ_2 in Eq. (2.13) with the choice $tol = 10^{-3}$
- A constant or 'manual' tolerance of 10^{-7}

The first choice results in tolerances on the order of 10^{-4} for small N and 10^{-5} for N closer to 65. The second choice gives tolerances of 10^{-3} for small N and 10^{-2} for $N \approx 65$.

3.2.2. Parameter choices for Algorithm 2

The example is also implemented with algorithm 2 for a single tolerance of -10^{-7}. Two values of n from Eq. (2.14) are chosen: -3 and 3. In almost every case, the energies of different local minima are very closely spaced and thus the normalization procedure described in Section 2.2.2 results in nearly identical tolerances for all N. Instructions for modifying the program are supplied in Appendix B.

3.3. Results

The following sections describe the results of local optimization for the three potentials. The discussion includes the number of local minima as a function of the number of particles, comments on scars, comments on the performance of the algorithm, numbers of starting configurations, and examples of some particular configurations in which optimization proves difficult.

3.3.1. Coulomb potential

The number of local minima is shown in Fig. 2. Prior to extraction of saddle points, both algorithms produce identical local minima. A complete list of all discovered local and global minima for the Coulomb potential, the corresponding energies, and other geometric properties is given in Table 1. A total of 116 minima and 5 saddle points were found.

In a planar geometry, particles interacting via a repulsive pairwise potential arrange themselves into a 2D hexagonal lattice in which each particle has a coordination number of $c_i = 6$ (see [16]). However, for a tessellation of the sphere the Euler theorem of topology prohibits all particles from having this property and thus a number of defects or *scars* must be present. As in [16], one can show that all possible defects that can occur on a sphere tessellated with n-sided polygons must satisfy

$$\sum_{i=1}^{N} (6 - c_i) = 12. \tag{3.18}$$

Therefore, the minimum number of defects that can occur is 12. This is the most common case among the local minima identified.

Table 1
Comprehensive list of local and global minima for the Coulomb potential. In order, the columns show the number of particles, the Coulomb energy, the number of particles with coordination numbers 3–7, and the number of iterations required to reach the given minimum averaged over the number of starting configurations.

N	Coulomb energy	$c_i = 3$	$c_i = 4$	$c_i = 5$	$c_i = 6$	$c_i = 7$	Average iterations
4	3.6742346	4	0	0	0	0	0
5	6.4746915	2	3	0	0	0	200
6	9.9852814	0	6	0	0	0	200
7	14.4529774	0	5	2	0	0	8100
8	19.6752879	0	4	4	0	0	1500
9	25.7599865	0	3	6	0	0	4700
10	32.7169495	0	2	8	0	0	2034
11	40.5964505	0	2	8	1	0	2250
12	49.1652531	0	0	12	0	0	517
13	58.8532306	0	1	10	2	0	5900
14	69.3063633	0	0	12	2	0	1886
15	80.6702441	0	0	12	3	0	4600
16	92.9116553	0	0	12	4	0	1556
	92.9203540	0	0	12	4	0	2160
17	106.0504048	0	0	12	5	0	4460
18	120.0844674	0	2	8	8	0	3156
19	135.0894676	0	0	12	7	0	93167
20	150.8815683	0	0	12	8	0	4273
21	167.6416224	0	1	10	10	0	31275
22	185.2875361	0	0	12	10	0	3767
	185.3079516	0	0	12	10	0	8800
23	203.9301907	0	0	12	11	0	2950
24	223.3470741	0	0	12	12	0	2543
25	243.8127603	0	0	12	13	0	70167
26	265.1333263	0	0	12	14	0	15812
27	287.3026150	0	0	12	15	0	3288
28	310.4915424	0	0	12	16	0	3275
29	334.6344399	0	0	12	17	0	14960
30	359.6039459	0	0	12	18	0	16412
31	385.5308381	0	0	12	19	0	1822
32	412.2612747	0	0	12	20	0	1055
	412.4683972	0	0	12	20	0	6950
33	440.2040574	0	0	13	19	1	28534
34	468.9048533	0	0	12	22	0	11075
35	498.5698725	0	0	12	23	0	29185
	498.5734540	0	0	12	23	0	70400
36	529.1224084	0	0	12	24	0	633422
37	560.6188877	0	0	12	25	0	10650
	560.6279731	0	0	12	25	0	27180
38	593.0385036	0	0	12	26	0	2850
	593.0489435	0	0	12	26	0	6222
39	626.3890090	0	0	12	27	0	2154
	626.4409584	0	0	12	27	0	55175
40	660.6752788	0	0	12	28	0	3028
	660.7253041	0	0	12	28	0	91364
	660.7412143	0	0	12	28	0	3000
41	695.9167443	0	0	12	29	0	2059
	695.9786994	0	0	12	29	0	9100
42	732.0781075	0	0	12	30	0	2443
43	769.1908465	0	0	12	31	0	85750
44	807.1742631	0	0	12	32	0	25350
45	846.1884011	0	0	12	33	0	50300
46	886.1671136	0	0	12	34	0	3000
	886.1702160	0	0	12	34	0	7260
	886.1714324	0	0	12	34	0	11200
47	927.0592707	0	0	12	35	0	8337
	927.0622697	0	0	12	35	0	30653
	927.0722246	0	0	12	35	0	103819
	927.0882335	0	0	12	35	0	13700
	927.1410883	0	0	12	35	0	12900
48	968.7134553	0	0	12	36	0	6445
49	1011.5571827	0	0	12	37	0	4460
50	1055.1823147	0	0	12	38	0	4135
51	1099.8192903	0	0	12	39	0	8134
	1099.9402311	0	0	12	39	0	5700
52	1145.4189643	0	0	12	40	0	4246
	1145.4219806	0	0	12	40	0	6245
	1145.4357090	0	0	12	40	0	8667
	1145.4375970	0	0	12	40	0	47367
53	1191.9222904	0	0	12	41	0	78006

(*continued on next page*)

Table 1 (*continued*)

N	Coulomb energy	$c_i = 3$	$c_i = 4$	$c_i = 5$	$c_i = 6$	$c_i = 7$	Average iterations
	1191.9315847	0	0	12	41	0	4431
54	1239.3614747	0	0	12	42	0	8723
	1239.3652553	0	0	12	42	0	12300
	1239.3711923	0	0	12	42	0	30850
	1239.3732007	0	0	12	42	0	7777
55	1287.7727208	0	0	12	43	0	5072
	1287.7770275	0	0	12	43	0	7993
	1287.7772608	0	0	14	39	2	15185
	1287.7887093	0	0	12	43	0	19375
	1287.7890572	0	0	12	43	0	8525
	1287.8001593	0	0	12	43	0	28900
56	1337.0949453	0	0	12	44	0	8515
	1337.0953483	0	0	12	44	0	10280
	1337.0987274	0	0	12	44	0	18371
57	1387.3832293	0	0	12	45	0	9383
	1387.4200823	0	0	13	43	1	16384
	1387.4303725	0	0	12	45	0	31867
	1387.4311301	0	0	12	45	0	10725
58	1438.6182506	0	0	12	46	0	6316
	1438.6255086	0	0	12	46	0	12780
	1438.6262899	0	0	12	46	0	11709
	1438.6272252	0	0	12	46	0	15695
	1438.6337080	0	0	12	46	0	118867
	1438.6381050	0	0	12	46	0	9355
	1438.6473598	0	0	12	46	0	15700
59	1490.7733353	0	0	14	43	2	12471
	1490.7743861	0	0	12	47	0	14356
	1490.7847558	0	0	12	47	0	56532
	1490.7907731	0	0	12	47	0	11164
60	1543.8304010	0	0	12	48	0	3873
	1543.8350996	0	0	12	48	0	4295
	1543.8415351	0	0	12	48	0	13250
	1543.9694723	0	0	12	48	0	60400
	1543.9807338	0	0	12	48	0	11000
61	1597.9418302	0	0	12	49	0	9773
	1597.9515553	0	0	12	49	0	5325
	1597.9551278	0	0	12	49	0	81158
	1597.9703606	0	0	12	49	0	17658
	1597.9726602	0	0	12	49	0	164500
	1597.9808036	0	0	12	49	0	21600
62	1652.9094099	0	0	12	50	0	4338
	1652.9285937	0	0	12	50	0	19363
	1652.9420143	0	0	12	50	0	7888
63	1708.8796815	0	0	12	51	0	9493
64	1765.8025779	0	0	12	52	0	9643
65	1823.6679603	0	0	12	53	0	22113

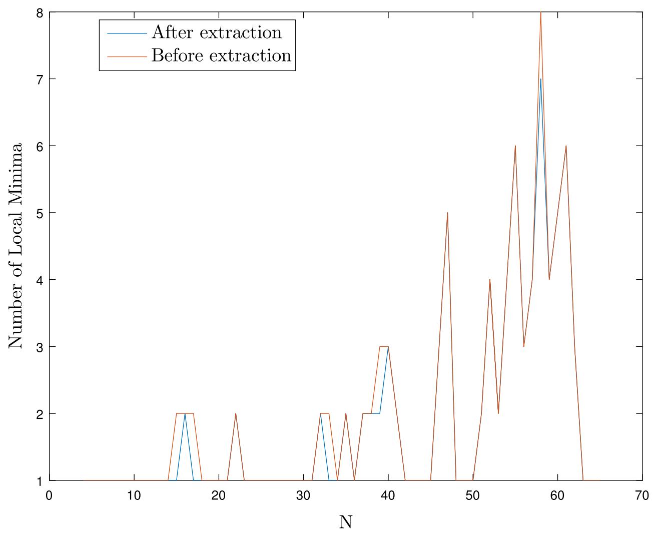


Fig. 2. Number of configurations found using the local optimization routine for the Coulomb potential. Results are shown before and after removing non-optimal configurations and are identical for both algorithms presented in Section 2.2. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

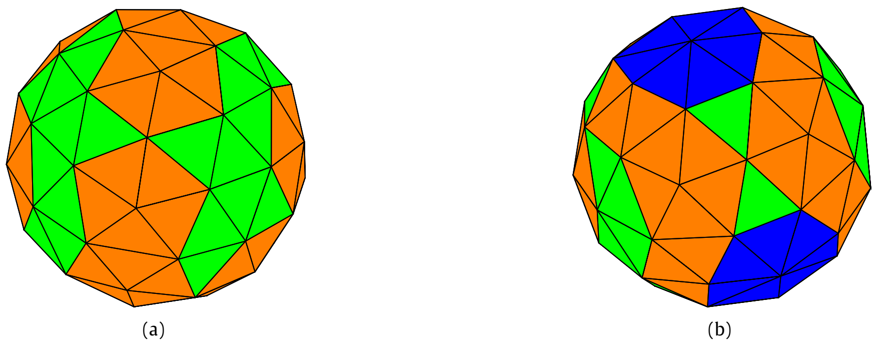


Fig. 3. Scar arrangements for the two most closely spaced Coulomb adjacent energy minima. Left: Arrangement of scars for the second lowest energy minimum for $N = 55$. Right: Arrangement of scars for the third lowest energy minimum for $N = 55$. Blue faces correspond to defects with coordination number 5 and orange faces to defects with coordination number 7. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

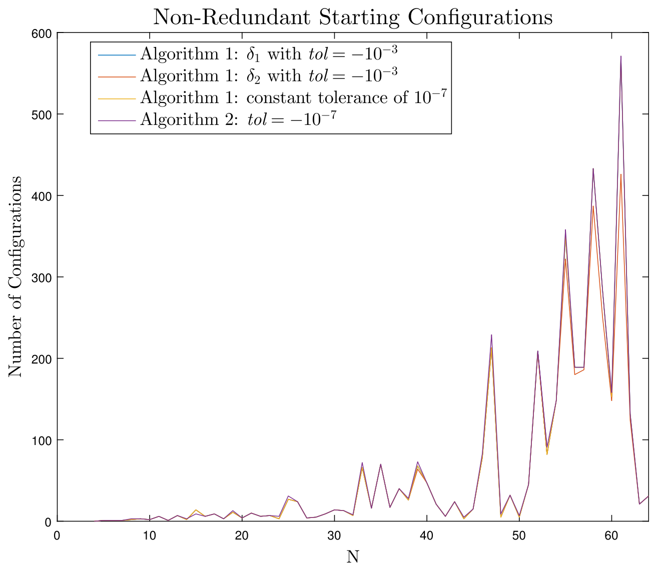


Fig. 4. Number of non-redundant starting configurations resulting from algorithms 1 and 2 for different tolerance choices. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The arrangement of scars for various local minima are quite interesting. Consider Fig. 6 which shows all the local minima found for $N = 60$. All of these minima exhibit twelve five-fold vertices and similar energies, however the defect structures are clearly different.

An energy spectrum normalized to the global energy minima is plotted in Fig. 5. Adjacent local minima are more closely spaced for higher particle numbers which illustrates the difficulty in distinguishing between geometrically different configurations using only the energy to characterize the arrangements for larger particle numbers. The two most closely spaced energies occur for the second and third smallest local minima for 55 particles, differing by only 2×10^{-5}%. Despite similar energies, the coordination numbers are different. Fig. 3 shows the arrangements of such defects for these two minima. The configuration in Fig. 3(a) has 12 particles with a coordination number of 5 while the configuration in Fig. 3(b) has 14 such particles and 2 particles with a coordination number of 7. Furthermore, comparing the pairwise distance vectors for both configurations as in 3.2.1, one finds that the difference is approximately 0.305. When equivalent configurations are compared in this manner, the result is on the order of 10^{-9} or less.

The number of starting configurations for the different algorithm and tolerance choices is shown in Fig. 4. The differences illustrate the importance of making an appropriate choice for removing redundant configurations. Using algorithm 1 with the tolerance δ_1 and the choice $tol = -10^{-3}$ in Eq. (2.13) produces the same number of starting configurations as the choice of the constant tolerance. The tolerance δ_2 produces slightly fewer starting configurations than the other choices. For every N, algorithm 2 produces the same or slightly more starting configurations. Ideally, one would find a range of tolerances spanning at least an order of magnitude that result in the same number of starting configurations. The similar results from the δ_1 tolerance and the constant tolerance indicate that the range of acceptable choices for algorithm 1 is larger than algorithm 2. However, it is relatively difficult to find a range of tolerances for algorithm 2 that produce results that agree closely with those of algorithm 1. Although the local minima in this case are identical for each choice of algorithm and tolerance, we choose to use algorithm 1 with δ_1 and $tol = 10^{-3}$ for the remaining potentials since a larger tolerance can be used to distinguish configurations. While all of the choices give the same local minima up to $N = 65$, the energies become more closely spaced as N increases and one will eventually be unable to distinguish configurations based on energy.

Some of the configurations have a negative-definite Hessian matrix despite having very small gradient components and are therefore believed to be saddle points. These configurations are not local minima although typically have energies that are very close to that of a local minimum. However the configurations are noticeably different from any of the local minima even upon visual inspection. Furthermore,

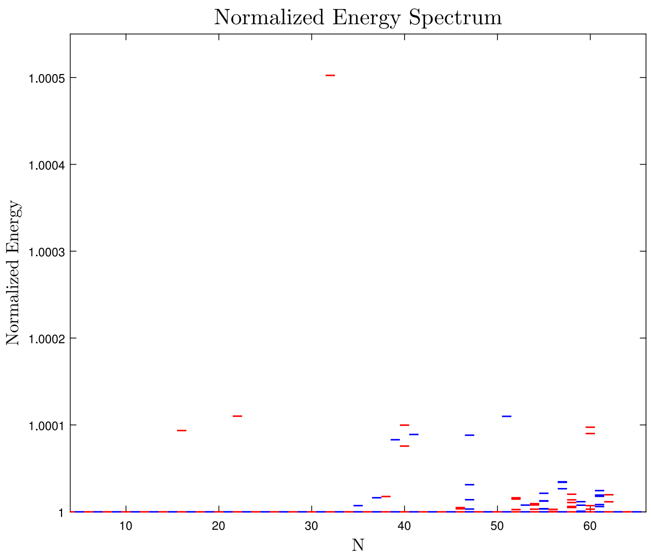


Fig. 5. Normalized Coulomb energies for locally and globally optimal configurations. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

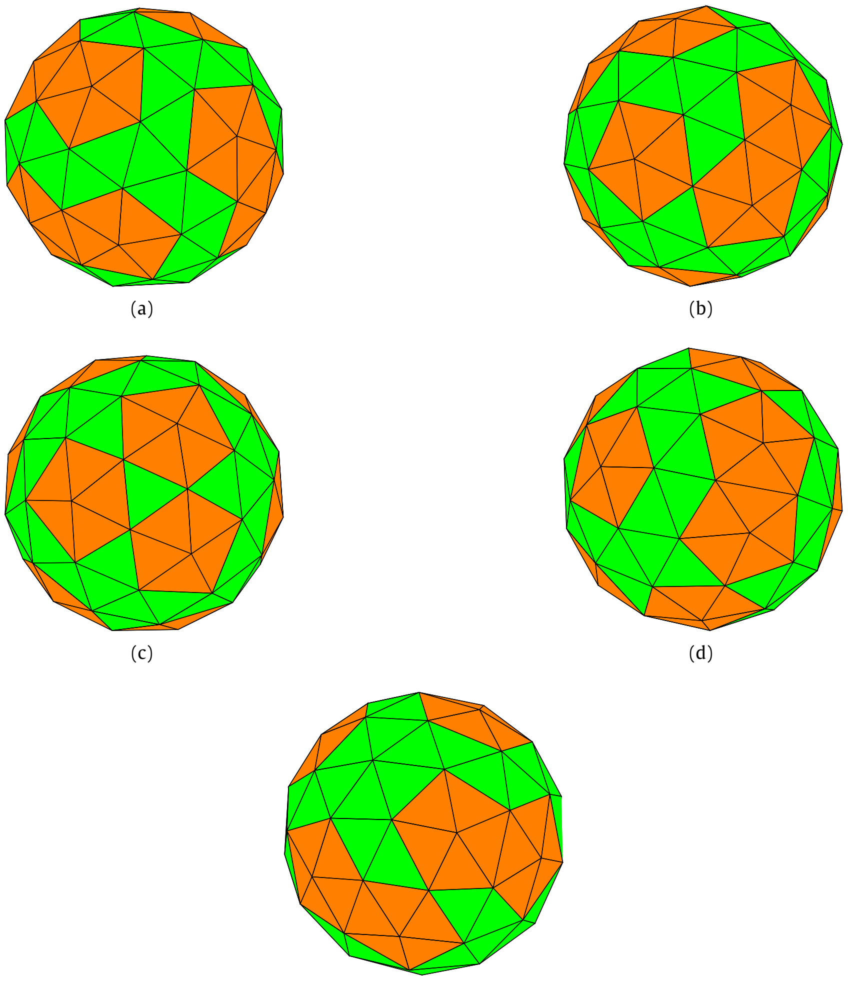
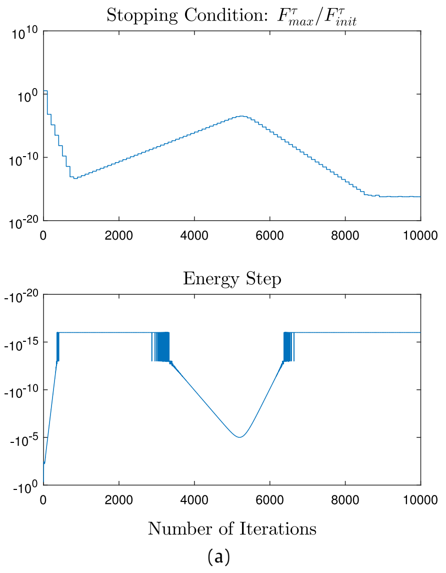


Fig. 6. Arrangements of scars for all $N = 60$ energy minima for the Coulomb potential, ordered (a–e) from lowest to highest energy.



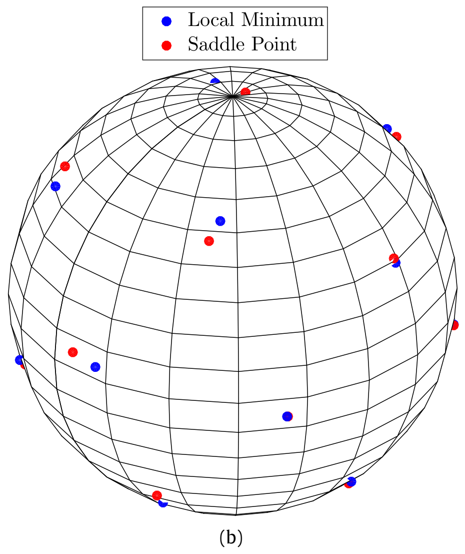


Fig. 7. Left: Evolution of the energy and the condition to terminate the optimization. Note that the energy steps are zero to numerical precision in many cases. These data points have been replaced with the value 10^{-16} so that they can be plotted on a logarithmic scale. Right: Resulting configurations after 1000 and 9000 iterations.

performing additional iterations using the optimization algorithm eventually results in the configuration falling into a local minimum. Most of these putative saddle point configurations cannot be identified with the local optimization routine since they satisfy the stopping condition and potentially require a large number of further iterations to fall into a local minimum. Further tightening of the stopping condition tolerance results in much longer runtimes as well as problems with numerical precision and is therefore impractical.

We give an example of this difficulty for the relatively simple case of $N = 17$. A fixed number of iterations (ignoring the stopping condition) were performed on a starting configuration generated using the algorithm described in Section 2.3. The starting configuration comes from the highest energy 16-particle local minimum. Fig. 7(a) shows the evolution of the Coulomb energy step and the stopping condition, $F^{\tau}_{max}/F^{\tau}_{init}$, during the optimization process (evaluated every 100 iterations). The energy step is zero to numerical precision during most of the optimization so these values are replaced with 10^{-16} for the purposes of visualization. The stopping condition is satisfied after 1000 iterations and thus the routine would normally stop at a non-optimal configuration. The configurations at 1000 and 9000 iterations are plotted in figure 7(b) and are noticeably different. The difference in Coulomb energy is about 3.3×10^{-3} which is roughly 0.003% of the optimal energy.

Saddle points appear for larger N as well and typically require many more iterations to fall into a local minimum. The evolution of $F^{\tau}_{max}/F^{\tau}_{init}$ shown in Fig. 7, which is simply a normalized tangential force, is typical for saddle points that have been found. This shows that these configurations are actually saddle points rather than points of slow convergence. It is remarkable that saddle points are computationally difficult to avoid with a steepest descent approach since in principle it should not find any.

The nonzero number of saddle points discovered suggests that the energy landscape near the saddle points has some structure that causes the optimization routine to stop at such configurations. Many of the eigenvalues of the Hessian are typically quite small ($\sim 10^{-3}$ or smaller) and therefore the energy landscape is "flat". This alone does not explain the occurrence of saddle points since one could simply take a larger descent step (by choosing a larger β). The slow convergence from a saddle point suggests that the unstable direction is very narrow such that the forces have only a very small component along this direction. A schematic of such a saddle point in two dimension is shown in Fig. 8. Eventually optimization does converge to a minimum and we believe that the algorithm escapes saddle points by 'hopping' across the unstable direction, slowly moving away from the saddle until the forces acquire a large enough component in the unstable direction to escape. These 'hops' are illustrated in Fig. 8 as jumps from point a) to b) to c). This 2-dimensional illustration is a significantly simplified view, however, it does provide some insight into the structure of the energy landscape.

An exhaustive list of local and global minima as well as corresponding properties (dipole moments, occurrence frequency, energy and angular diversity, etc.) is given by [22]. The approach uses a steepest descent optimization algorithm similar to (2.9) except the *total* force is used and the position increment is used in the limit $\gamma \to \infty$. Starting configurations are generated randomly to give a uniform spherical distribution. A comparison between the configurations identified by [22] and those obtained here provides significant insight into the properties of the configurations and the remarkably frequent occurrence of non-optimal configurations. We note two minor corrections to the configurations found by [22], specifically the highest energy 39- and 58-particle configurations, for which Hessian is negative definite. The energies were reproduced here exactly to the given precision (7 decimal places) for the 39-particle configuration and to 6-decimal places for the 58-particle configuration. A comparison between the quoted local minima (excluding the 38-particle and 59-particle saddle points) is shown in Fig. 9. All of the local minima identified here can be matched with configurations from [22], however the configurations that were not reproduced here cannot be verified since the particle coordinates themselves are not available.

It is worth noting that non-optimal configurations should not be excluded based entirely on the occurrence frequency. The non-optimal 58-particle configuration in [22] has an occurrence frequency of only 0.04%, however there are 58-particle configurations in [22] with lower occurrence frequencies that are apparently optimal. Further, the non-optimal 39-particle configuration has a large occurrence frequency of 4.55%.

Table 2
Comprehensive list of local and global minima for the inverse square law potential. In order, the columns show the number of particles, the inverse power law energy, the number of particles with coordination numbers 3–7, and the number of iterations required to reach the given minimum averaged over the number of starting configurations.

N	Inverse power law energy	$c_i = 3$	$c_i = 4$	$c_i = 5$	$c_i = 6$	$c_i = 7$	Average iterations
4	2.2500000	4	0	0	0	0	0
5	4.2500000	2	3	0	0	0	300
6	6.7500000	0	6	0	0	0	300
7	10.2500000	0	5	2	0	0	15000000
8	14.3367911	0	4	4	0	0	1300
9	19.2528688	0	3	6	0	0	3900
10	25.0413597	0	2	8	0	0	3429
11	31.8347216	0	2	8	1	0	2650
12	39.0000000	0	0	12	0	0	667
13	47.7733090	0	1	10	2	0	7600
14	57.1212086	0	0	12	2	0	2358
15	67.4861847	0	0	12	3	0	6000
16	78.7726490	0	0	12	4	0	2478
	78.8054346	0	0	12	4	0	2880
17	91.0473193	0	0	12	5	0	5180
18	104.3146853	0	2	8	8	0	4112
19	118.8255636	0	0	12	7	0	288300
20	133.9369786	0	0	12	8	0	5206
21	150.3251227	0	1	10	10	0	33650
22	167.6657856	0	0	12	10	0	4478
	167.7622724	0	0	12	10	0	11200
23	186.4037129	0	0	12	11	0	3750
24	205.6584380	0	0	12	12	0	3300
25	226.5450773	0	0	12	13	0	81700
26	248.2671389	0	0	12	14	0	20141
27	270.7984042	0	0	12	15	0	4488
28	294.8784716	0	0	12	16	0	4175
29	320.2160318	0	0	12	17	0	12960
30	346.2636306	0	0	12	18	0	20123
31	373.5808690	0	0	12	19	0	2429
32	401.5000000	0	0	12	20	0	1340
	402.4773719	0	0	12	20	0	8567
33	431.9318386	0	0	12	21	0	32986
34	462.7012364	0	0	12	22	0	13678
35	494.8164320	0	0	12	23	0	241107
36	527.9142566	0	0	12	24	0	353876
37	562.2556382	0	0	12	25	0	10450
	562.2952168	0	0	12	25	0	29277
38	597.7394531	0	0	12	26	0	3675
	597.7901988	0	0	12	26	0	7758
39	634.4153334	0	0	12	27	0	2687
	634.6932924	0	0	12	27	0	55700
40	672.3093535	0	0	12	28	0	5362
	672.5909921	0	0	12	28	0	399034
	672.6569114	0	0	12	28	0	3600
41	711.5261515	0	0	12	29	0	2571
	711.8931991	0	0	12	29	0	11067
42	751.8751968	0	0	12	30	0	2900
	752.5328625	0	0	12	30	0	1218500
43	793.5218863	0	0	12	31	0	71970
44	836.0418318	0	0	12	32	0	26059
45	880.3579693	0	0	12	33	0	97467
46	926.0623438	0	0	12	34	0	2200
	926.0698337	0	0	12	34	0	11000
	926.0707424	0	0	12	34	0	9650
	926.1074258	0	0	12	34	0	10700
47	972.8237449	0	0	12	35	0	11255
	972.8393193	0	0	12	35	0	37003
	972.8889969	0	0	12	35	0	193636
	973.0094385	0	0	12	35	0	30900
	973.3156444	0	0	12	35	0	11900
48	1019.8295806	0	0	12	36	0	8684
49	1069.5597398	0	0	12	37	0	5540
50	1119.5995065	0	0	12	38	0	5010
51	1171.3283814	0	0	12	39	0	11067
	1172.0869779	0	0	12	39	0	6250
52	1224.4784561	0	0	12	40	0	5153
	1224.4984399	0	0	12	40	0	9000
	1224.5870561	0	0	12	40	0	9900
	1224.6012659	0	0	12	40	0	46756

(*continued on next page*)

Table 2 (*continued*)

N	Inverse power law energy	$c_i = 3$	$c_i = 4$	$c_i = 5$	$c_i = 6$	$c_i = 7$	Average iterations
53	1278.6522062	0	0	12	41	0	93346
	1278.7144239	0	0	12	41	0	5675
54	1334.0848954	0	0	12	42	0	11407
	1334.1009935	0	0	12	42	0	13200
	1334.1442304	0	0	12	42	0	25250
	1334.1673488	0	0	12	42	0	10916
55	1390.9691942	0	0	12	43	0	7107
	1390.9854274	0	0	12	43	0	9832
	1390.9959619	0	0	14	39	2	21104
	1391.0779067	0	0	12	43	0	23420
	1391.0786667	0	0	12	43	0	10232
	1391.1428240	0	0	12	43	0	25800
56	1448.9542741	0	0	12	44	0	9118
	1448.9605281	0	0	12	44	0	12146
	1448.9750705	0	0	12	44	0	20715
	1449.8392941	0	0	12	44	0	51700
57	1508.3688385	0	0	12	45	0	7816
	1508.4277593	0	0	12	45	0	28550
	1508.4322230	0	0	14	41	2	14223
	1508.5853825	0	0	13	43	1	21193
	1508.6314958	0	0	12	45	0	11388
	1508.6578101	0	0	12	45	0	28150
58	1569.0699385	0	0	12	46	0	8300
	1569.1051146	0	0	12	46	0	12700
	1569.1064753	0	0	12	46	0	14957
	1569.1135768	0	0	12	46	0	29078
	1569.1243234	0	0	12	46	0	80969
	1569.1654449	0	0	12	46	0	23150
	1569.1846489	0	0	12	46	0	13329
	1569.2582432	0	0	12	46	0	15525
59	1630.9096583	0	0	12	47	0	30045
	1630.9107941	0	0	14	43	2	17790
	1630.9806275	0	0	12	47	0	64542
	1631.0331471	0	0	12	47	0	12955
60	1693.7946118	0	0	12	48	0	4880
	1693.8310725	0	0	12	48	0	5352
	1693.8823156	0	0	12	48	0	13064
	1694.6789561	0	0	12	48	0	25300
	1694.7700688	0	0	12	48	0	11900
61	1758.6971340	0	0	12	49	0	12349
	1758.7607346	0	0	12	49	0	6331
	1758.7927630	0	0	12	49	0	107310
	1758.8850228	0	0	12	49	0	18029
	1758.8982585	0	0	12	49	0	151200
	1758.9516937	0	0	12	49	0	22900
62	1824.3469264	0	0	12	50	0	5486
	1824.4602937	0	0	12	50	0	20173
	1824.5469417	0	0	12	50	0	10292
63	1891.6323308	0	0	12	51	0	11507
64	1960.2816270	0	0	12	52	0	10681
65	2030.2336785	0	0	12	53	0	28226

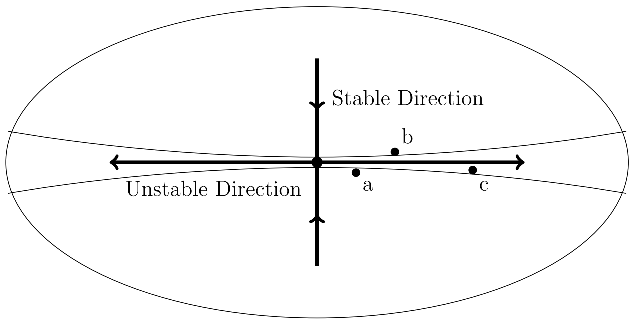


Fig. 8. Schematic illustration of an analogous saddle point in two dimensions showing a narrow unstable direction. The path a→b→c is a simplified picture of the steps that the optimization algorithm might take to escape the saddle point.

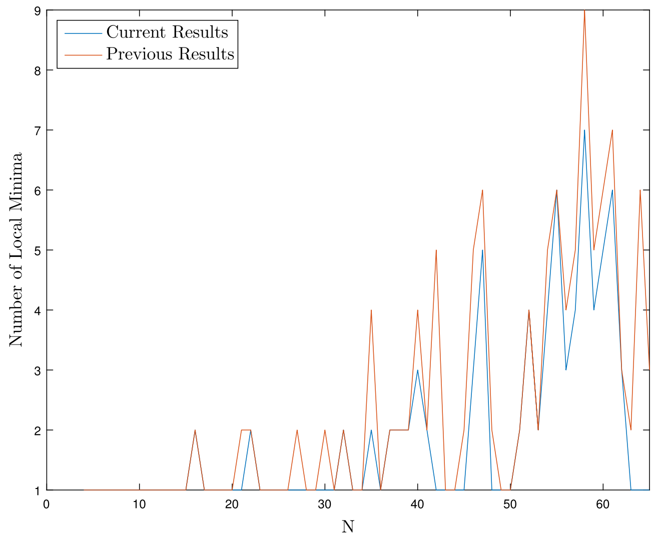


Fig. 9. Number of local minima discovered for the Coulomb potential compared with results by Erber & Hockney [22] for the Coulomb potential. Note that the two non-optimal configurations for $N = 39$ and $N = 58$ have been excluded from the previous results. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

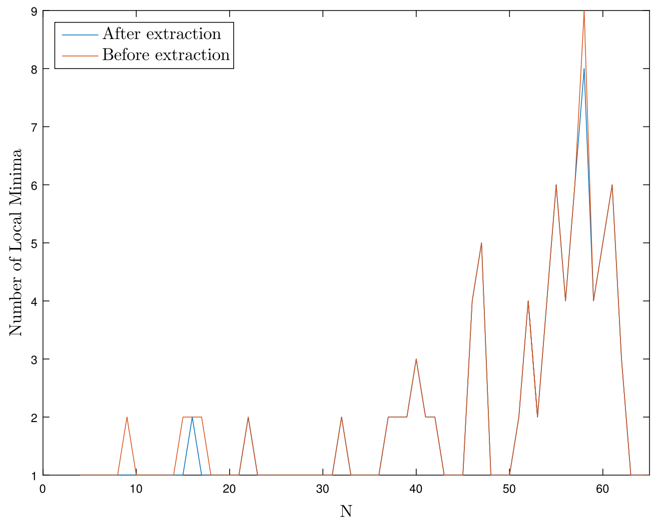


Fig. 10. Number of configurations found using the local optimization routine for the inverse square law. Results are shown before and after removing non-optimal configurations. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3.3.2. *Inverse square law* $1/r^2$

Analogous results for the inverse square law, Eq. (1.6), are now presented. As per the discussion of different tolerance choices in algorithms 1 and 2, we choose to use algorithm 1 with the tolerance δ_1 and $tol = 10^{-3}$ in Eq. (2.13).

The number of local minima is shown in Fig. 10. A total of 121 minima and 4 saddle points were found. A comprehensive list of the local and global minima, the corresponding energies, and other geometric properties is given in Table 2. At the time of writing the authors are not aware of any data for this potential and therefore no comparisons can be made. The number of starting configurations is shown in Fig. 11.

The saddle points that arise for the inverse square law potential are also computationally difficult to avoid. In fact, the energy step and the stopping condition for the 17-particle saddle point behave similar to that for the Coulomb potential shown in Fig. 7.

A normalized energy spectrum is plotted in Fig. 12. The two most closely spaced energies are the fourth and fifth lowest energies for 55 particles which differ by only 5×10^{-5}%, similar to the Coulomb case. The particles in each of the two configurations have identical coordination numbers, however the configurations can be seen to be different when one observes the scars.

3.3.3. *Logarithmic potential*

Local optimization for the logarithmic potential is more challenging than the previous potentials since the forces are long range. The number of local minima is shown in Fig. 13 and the number of starting configurations in Fig. 14. We remark that the one configuration found for $N = 33$ was found to be a saddle point and hence no minima were found for 33 particles in this run. This difficulty is due in part to the high symmetry of the 32-particle configuration which admits only 1 starting configuration. A number of attempts were made to avoid

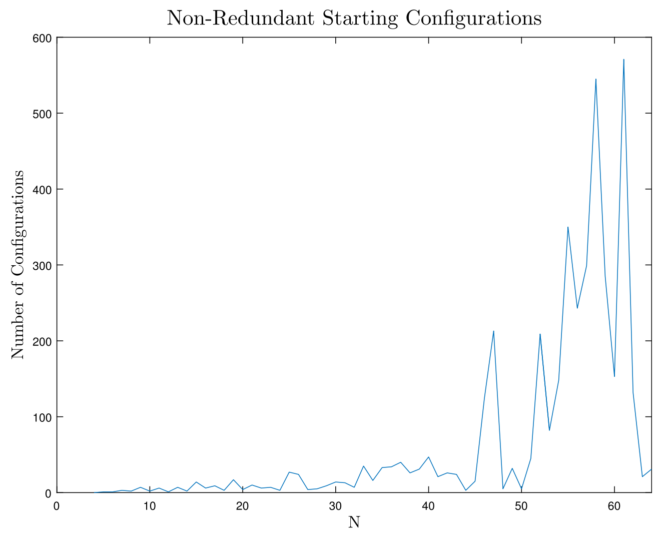


Fig. 11. Number of non-redundant starting configurations resulting from algorithm 1 with *tol* $= 10^{-3}$ for the inverse square law potential. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

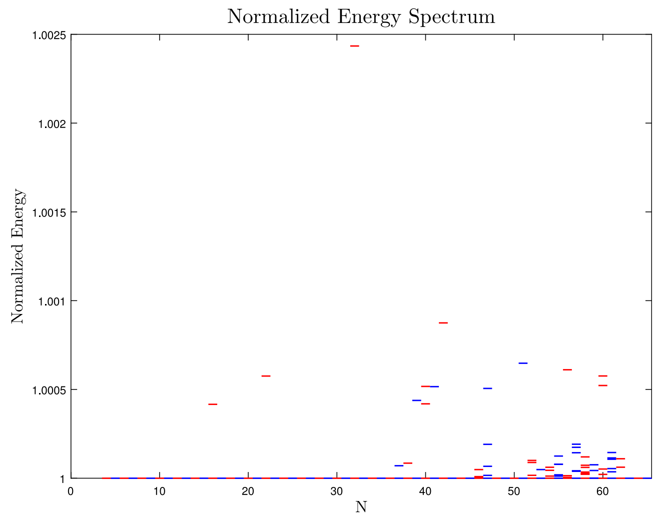


Fig. 12. Normalized inverse square law energies for locally and globally optimal configurations. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

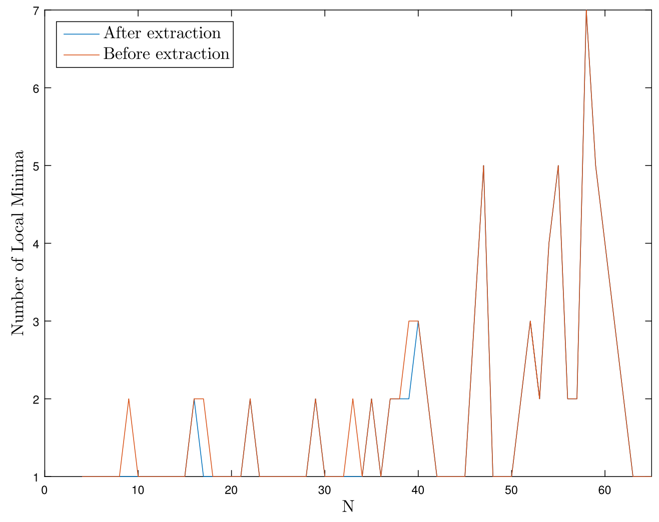


Fig. 13. Number of configurations found using the local optimization routine for the Logarithmic potential. Results are shown before and after removing non-optimal configurations. Note that the missing 33-particle configuration has been included.

Table 3
Comprehensive list of local and global minima for the logarithmic potential. In order, the columns show the number of particles, the logarithmic energy, the number of particles with coordination numbers 3–7, and the number of iterations required to reach the given minimum averaged over the number of starting configurations.

N	Logarithmic energy	$c_i = 3$	$c_i = 4$	$c_i = 5$	$c_i = 6$	$c_i = 7$	Average iterations
4	−2.9424878	4	0	0	0	0	0
5	−4.4205072	2	3	0	0	0	300
6	−6.2383246	0	6	0	0	0	200
7	−8.1824779	0	5	2	0	0	15000000
8	−10.4280178	0	4	4	0	0	900
9	−12.8877527	0	3	6	0	0	3400
10	−15.5631234	0	2	8	0	0	2815
11	−18.4204797	0	2	8	1	0	2950
12	−21.6061452	0	0	12	0	0	534
13	−24.8667219	0	1	10	2	0	6700
14	−28.4078130	0	0	12	2	0	2472
15	−32.1478763	0	0	12	3	0	5900
16	−36.1061522	0	0	12	4	0	1600
	−36.1027947	0	0	12	4	0	2600
17	−40.2730670	0	0	12	5	0	4880
18	−44.6502873	0	2	8	8	0	3400
19	−49.1998916	0	0	12	7	0	53534
20	−54.0111300	0	0	12	8	0	4137
21	−59.0009121	0	1	10	10	0	69875
22	−64.2060078	0	0	12	10	0	4523
	−64.1988790	0	0	12	10	0	9600
23	−69.5783826	0	0	12	11	0	3584
24	−75.2139848	0	0	12	12	0	2686
25	−80.9975100	0	0	12	13	0	101267
26	−87.0094231	0	0	12	14	0	18775
27	−93.2519864	0	0	12	15	0	3567
28	−99.6586094	0	0	12	16	0	3750
29	−106.2545712	0	0	12	17	0	69950
	−106.2544605	0	0	12	17	0	46000
30	−113.0892555	0	0	12	18	0	21042
31	−120.1103466	0	0	12	19	0	2015
32	−127.3788676	0	0	12	20	0	1724
33	−134.7478208	0	0	13	19	1	1000000
34	−142.3758523	0	0	12	22	0	15091
35	−150.1920585	0	0	12	23	0	15200
	−150.1916099	0	0	12	23	0	20580
36	−158.2240684	0	0	12	24	0	162559
37	−166.4506975	0	0	12	25	0	19600
	−166.4473062	0	0	12	25	0	40714
38	−174.8801972	0	0	12	26	0	3308
	−174.8761457	0	0	12	26	0	8193
39	−183.5092257	0	0	12	27	0	2607
	−183.4934373	0	0	12	27	0	63412
40	−192.3376899	0	0	12	28	0	3023
	−192.3235723	0	0	12	28	0	49974
	−192.3168507	0	0	12	28	0	4360
41	−201.3592066	0	0	12	29	0	2554
	−201.3427393	0	0	12	29	0	10725
42	−210.5845116	0	0	12	30	0	3005
43	−220.0034771	0	0	12	31	0	188667
44	−229.6418015	0	0	12	32	0	27092
45	−239.4536983	0	0	12	33	0	33867
46	−249.4558479	0	0	12	34	0	3700
	−249.4546504	0	0	12	34	0	13700
	−249.4525407	0	0	12	34	0	14200
47	−259.6617599	0	0	12	35	0	9354
	−259.6608278	0	0	12	35	0	55400
	−259.6572394	0	0	12	35	0	92347
	−259.6545257	0	0	12	35	0	9567
	−259.6389655	0	0	12	35	0	102550
48	−270.1179500	0	0	12	36	0	6714
49	−280.7019031	0	0	12	37	0	5020
50	−291.5286007	0	0	12	38	0	4913
51	−302.5336735	0	0	12	39	0	8067
	−302.5020271	0	0	14	35	2	8250
52	−313.7323719	0	0	12	40	0	5215
	−313.7315714	0	0	12	40	0	7645
	−313.7281773	0	0	12	40	0	10800
53	−325.1382347	0	0	12	41	0	126538

(*continued on next page*)

Table 3 (*continued*)

N	Logarithmic energy	$c_i = 3$	$c_i = 4$	$c_i = 5$	$c_i = 6$	$c_i = 7$	Average iterations
	−325.1361679	0	0	12	41	0	5546
54	−336.7454644	0	0	12	42	0	10520
	−336.7439517	0	0	12	42	0	22100
	−336.7429857	0	0	12	42	0	8308
	−336.7428868	0	0	12	42	0	260034
55	−348.5417963	0	0	12	43	0	5409
	−348.5405384	0	0	14	39	2	15373
	−348.5400024	0	0	12	43	0	10500
	−348.5379975	0	0	12	43	0	24216
	−348.5379778	0	0	12	43	0	11263
56	−360.5458992	0	0	12	44	0	16896
	−360.5456829	0	0	12	44	0	14628
57	−372.7412006	0	0	12	45	0	7990
	−372.7307204	0	0	13	43	1	23150
58	−385.1328298	0	0	12	46	0	7384
	−385.1306381	0	0	12	46	0	205915
	−385.1304746	0	0	12	46	0	11813
	−385.1297172	0	0	12	46	0	33560
	−385.1269588	0	0	12	46	0	10595
	−385.1265181	0	0	12	46	0	41034
	−385.1017210	0	0	14	42	2	47100
59	−397.7281497	0	0	14	43	2	11457
	−397.7274834	0	0	12	47	0	15606
	−397.7261278	0	0	12	47	0	50472
	−397.7249559	0	0	12	47	0	1805743
	−397.7242154	0	0	12	47	0	15319
60	−410.5331628	0	0	12	48	0	4896
	−410.5322566	0	0	12	48	0	5332
	−410.5311780	0	0	12	48	0	27984
	−410.4944688	0	0	12	48	0	23900
61	−423.5076360	0	0	12	49	0	12377
	−423.5051974	0	0	12	49	0	6848
	−423.5048303	0	0	12	49	0	295760
62	−436.7039792	0	0	12	50	0	5864
	−436.6952127	0	0	12	50	0	10950
63	−450.0812392	0	0	12	51	0	11782
64	−463.6544330	0	0	12	52	0	21472
65	−477.4264261	0	0	12	53	0	24052

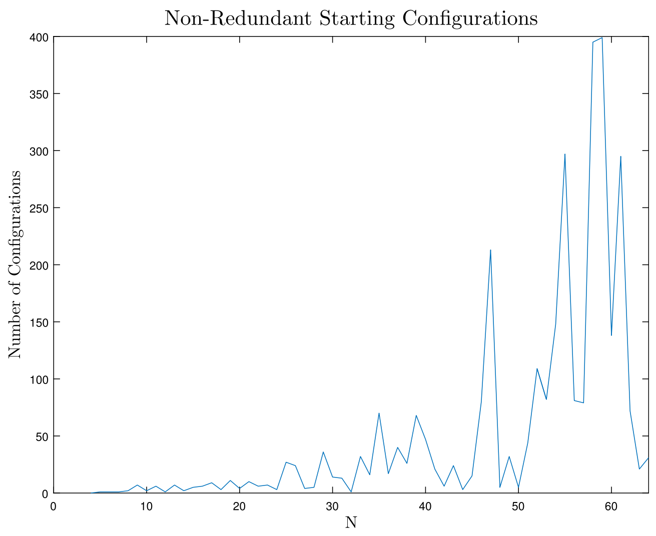


Fig. 14. Number of non-redundant starting configurations resulting from algorithm 1 with $tol = 10^{-3}$ for the Logarithmic potential. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

this saddle point by adjusting the weight parameter β, however all attempts failed. A global minimum was obtained by setting the weight parameter to 0.4 and performing 10^6 iterations on the single starting configuration (ignoring the stopping condition). None of the other parameters were changed. We remark that the saddle point is found after only 10^4 iterations but the global minimum requires around 5×10^5 iterations to reach. In the discussion and figures that follow, the 33-particle configuration refers to the configuration obtained in this way, not the saddle point. A total of 107 minima (including the 33-particle global minimum) and 4 saddle points (including the 33-particle saddle point) were found (see Table 3).

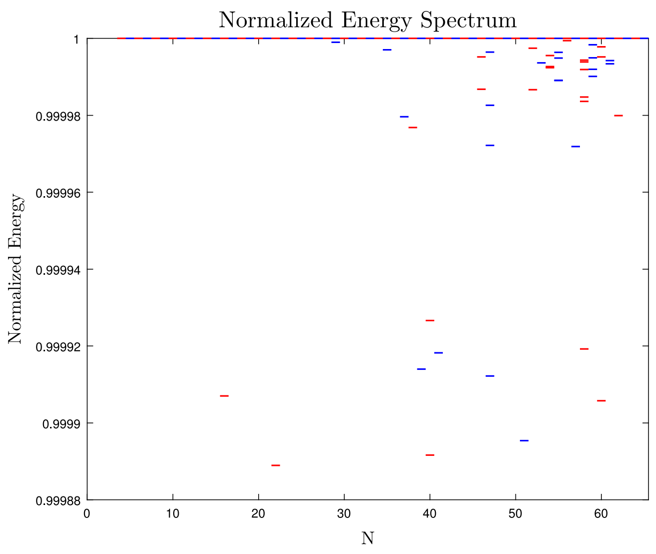


Fig. 15. Normalized inverse square law energies for locally and globally optimal configurations. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

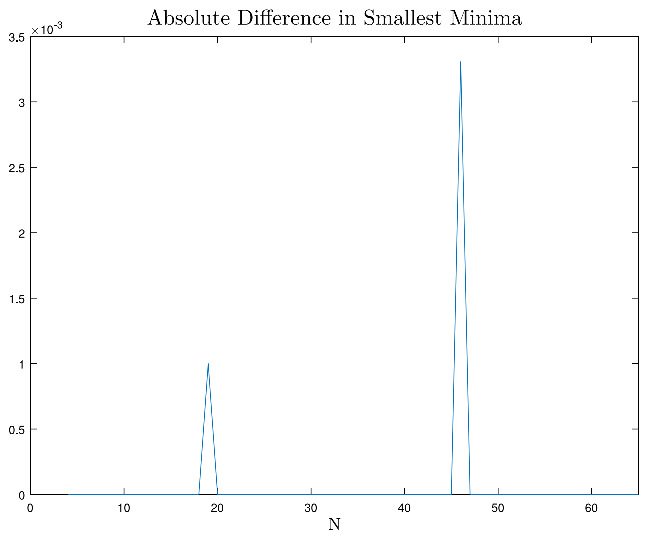


Fig. 16. Difference between the globally optimal energies obtained by [24] and those obtained here. Two new global minima have been found, one at $N = 19$ and another at $N = 46$.

Globally optimal energies have previously been found by [24] up to 9 significant digits. Fig. 16 shows the difference between these energies and those obtained here up to the same precision. We note that two new global minima have been found, one for $N = 19$ and another for $N = 46$. All 45 local minima given here are also new.

The normalized energy minima are shown in Fig. 15. Note that the logarithmic energies are negative and therefore the global minima are larger in magnitude than the local minima. The two closest energy levels again occur at $N = 55$ between the fourth and fifth lowest energies. The difference is around 6×10^{-6}% but as with the other potentials they can be easily distinguished both visually with the scar picture and with the algorithms from Section 2.2.

4. Discussion

An iterative systematic procedure to compute multiple locally and putative globally optimal spherical designs for pairwise potentials has been presented. The proposed algorithm generates N-particle starting configurations from known $(N - 1)$-particle optimal configurations and performs a modified steepest-descent optimization to compute local and global minima. Geometrically equivalent configurations are identified and excluded using pairwise distances or total energy. Saddle points are excluded using a symbolic-numerical Maple-based routine that computes the eigenvalues of the Hessian matrix.

Sample calculations were performed up to $N = 65$ for the Coulombic, logarithmic, and inverse square law potentials. The algorithm reproduces most, if not all, known putatively optimal configurations for the Coulombic and logarithmic potentials. Two new global minima and 45 new local minima were discovered for the logarithmic potential. The inverse square law has not previously been studied and thus all configurations for this potential are new. The number of starting configurations required is much fewer than is required for randomly generated configurations.

The presented computational procedure is able to handle a wide class of pairwise potentials, and can be generalized to other kinds of surfaces and interactions, in particular, cases involving interacting particles of non-equal "charges". It is planned to apply the algorithm to the cases of more complicated potential functions, such as non-monotone pairwise potentials, as well as to the cases of non-spherical domains in appropriate curvilinear coordinates (see, e.g., [34]).

Another important future work direction is the improvement of the optimization algorithm convergence near a saddle point, for example, by implementing an eigenvector-following technique (cf. [23]).

Acknowledgments

The authors are grateful to NSERC of Canada for support through a Discovery grant and a USRA fellowship.

Appendix A. Program structure: local optimization algorithm

The local optimization algorithm implements a modified steepest descent algorithm as given in Eq. (2.9) to find a local minimum from a given starting configuration. The program is implemented in C++ and consists of several files:

- `main.cpp`: Responsible for reading in the input arguments, beginning the algorithm, and writing the results to the output file. The particles' coordinates in the starting configuration are also initialized here.
- `simulation.cpp`, `simulation.h`: Defines a class whose member functions perform the iteration scheme described in Section 2.1.
- `sph_vector.cpp`, `sph_vector.h`: Defines a class containing member functions that define cartesian vectors and perform some basic operations such as scaling and adding vectors.
- `sph_vector_functions.cpp`: A namespace defining support functions that compute force, energy, distances, etc.
- `simulation.h`, `sph_vector.h`, `sph_vector_functions.h`: Header files.
- The makefile.
- The executable.

The Windows executable is named `trap_dyn_sys_xxxx.exe` where `xxxx` is used to abbreviate the pairwise potential used i.e. coul, log, etc. The program accepts three input arguments via the command line and has two command line outputs. An output file is also generated which contains the configuration after the final iteration. See Table 4 for input and output descriptions. The following command-line input executes the program for the Coulomb potential with a weight parameter of 0.5 (see Eq. (2.11)) using the starting configuration specified in *infilename* and writing the optimal configuration to *outfilename*.

`trap_dyn_sys_coul.exe` *infilename outfilename weight*

The program should rarely be executed directly from the command line except in the case where debugging is required. The MATLAB scripts deal with executing the optimization algorithm. Executing the program once will perform local optimization on the provided starting configuration and produce an output file with the resulting optimal configuration.

Table 4
Descriptions of input arguments (in order) and outputs for the program implementing the local optimization routine.

	Name	Description
Inputs	*infilename*	A string specifying the name of the input file containing the starting configuration. The format is a single column consisting of all the polar angles θ_i followed by all the azimuthal angles ϕ_i in the same column.
	outfilename	A string specifying the name of the output file to which the results are printed. The format is identical to *infilename*.
	weight	A string specifying the parameter β in Eq. (2.11).
Outputs	–	The first command-line output giving the energy after the last iteration.
	–	The final command-line output specifying the maximum tangential force (see Section 2.1)
	–	An output file containing the particle configuration after the final iteration. The file name is given by *outfilename* and the format is identical to *infilename*.

Appendix B. Program structure: the identification of different configurations

The two algorithms for identifying different configurations (see Section 2.2) are implemented in MATLAB and each requires the Statistics Toolbox. Algorithms 1 and 2 implement the routines utilizing pairwise distances and energy respectively.

Algorithm 1 consists of two files:

- `remove_redundant_dist.m`: The function implementing algorithm 1. Requires the Statistics Toolbox.
- `pairwise_distance.m`: A support function that computes every pairwise distance between the pairs of particles

The input and output parameters for `remove_redundant_dist.m` are described in Table 5. The input *cell_en_coord* is formatted into k rows and 3 or more columns where k is the number of configurations to be compared. Only the second column contains data that is relevant to the algorithm. Each cell in the second column contains an $N \times 5$ matrix. In this matrix, the first and second columns contain the polar and azimuthal angles respectively of the N particles and the final three columns contain the particles' cartesian coordinates. Note that the

Table 5
Descriptions of inputs and outputs for the functions programs implementing algorithms 1 and 2, `remove_redundant_xxxx.m`.

	Name	Description
Inputs	*cell_en_coord*	A cell array variable containing the configurations to be compared. The array can contain other information that is relevant to the configurations and will not be modified by the function.
	max_tol	The tolerance used in the clustering described in Section 2.2.
	pow_array	A vector containing the user-specified values of n in Eq. (2.14) (`remove_redundant_energy.m` only).
Output	*really_different_configs*	A cell array identical to *cell_en_coord* but with the rows corresponding to redundant configurations removed.

other cell columns can contain relevant information and will not be modified by the function with two exceptions: the column containing the number of iterations and the column containing parent numbers. In general, these columns will contain entries that are different for each configuration, even those that are geometrically equivalent. When the identical configurations are removed, the average number of iterations is computed so that the output cell array contains the average number of iterations required to obtain the corresponding local minimum. The parent numbers keep track of the $(N-1)$-particle local minima that yield a given N-particle minimum after optimization. Since multiple $(N-1)$-particle local minima can yield the same N-particle minimum, the smallest parent number is used for the output cell array.

Algorithm 2 consists of four files

- `remove_redundant_energy.m` Contains the implementation of the pairwise energy clustering.
- `remove_redundant_coord.m` Implements the clustering based on coordination number.
- `get_coordination_all.m` Computes the coordination number for each particle.
- `pairwise_energy.m` Calculates energy of a given configuration.

The input and output parameters for `remove_redundant_energy.m` and `remove_redundant_coord.m` are given in Table 5. The usage of these functions is demonstrated within the main optimization script given in Appendix D.

Appendix C. Program structure: the starting configurations

The algorithm is implemented in MATLAB but not as a standalone function. It is part of a larger routine described in Section 3 and shown in Appendix D within the main routine. There is one support function:

- `compute_triangle_middles_D.m`: Performs Delaunay Triangulation on the particle locations and computes the convex hull. Returns the center of mass of each of the triangular facets that make up the convex hull.

Appendix D. Program structure: run examples

This section contains descriptions of the main routine and support functions used to produce the results in Section 3 for the Coulomb potential. The required file structure is also described.

The following file structure is used.

- `all_loc_min.m`
- `compute_triangle_middles_D.m`
- `coordination_numbers.m`
- `coul_energy.m`
- `eval_coul_energy.m`
- `get_coordination_all.m`
- `get_coordination5.m`
- `number_loc_min.m`
- `Opt_procedure.m`
- `pairwise_distance.m`
- `pairwise_energy.m`
- `plot_configuration.m`
- `point_coordinates.m`
- `read_data.m`
- `remove_redundant_coord.m`
- `remove_redundant_dist.m`
- `remove_redundant_energy.m`
- `run_dyn_sys.m`
- `trap_dyn_sys_coul.exe`
- `write_data.m`
- *C++Source* (folder)
  - All C++ files listed in Appendix A

- *input* (folder)
  - initialConfig_N4.dat
- *Maple* (folder)
  - *configFiles* (folder)
  - *io* (folder)
  - extract_optimal_configs.m
  - MinChecker.mpl
  - MinChecker.mw
  - run_min_checker.m
  - write_maple_data.m
- *output* (folder)
- *temp* (folder)

The main script described in Section 3 is contained in Opt_procedure.m. The files under the *Maple* subfolder perform the removal of saddle point configurations. The script that executes the Maple program described in Section 2.4 is extract_optimal_configs.m.

After executing Opt_procedure.m the *output* and *temp* folders will contain some files. The new file in the *temp* folder is simply a log file containing the MATLAB command window output as a .txt file. The others will be stored in *output* and are files containing MATLAB variables storing the result of the optimization after each N. Each of these files stores the results of all previous optimization up to that N. For example a file called *N065-loc_opt.mat* will store a MATLAB variable containing all minima from $N = 4$ up to $N = 65$. There is flag called *keep_temp_files* which can be set to keep intermediate files that would normally be deleted which contain the starting configurations as well as the raw text files resulting from the optimization algorithm. The initial and final values of N can be modified as well. To change the final value of N, change the value of *N_END* to the desired value. The default starting value for N is 4 which can be modified by commenting the section of code delimited by "load to start from N=4" and uncommenting the section of code immediately after. Then the value of *N_START* should be changed to the desired value. Note that any starting value of N other than 4 requires an additional .mat file to load the local minima for the chosen value of *N_START*. This file is automatically generated by the program after optimization for each N. The option to change the starting value of N is most useful when one wishes to pause the computations and resume at a later time.

The potential can be changed by modifying the following files.

- coul_energy.m
- eval_coul_energy.m
- Opt_procedure.m
- trap_dyn_sys_coul.exe
- sph_vector_functions.cpp
- MinChecker.mpl
- MinChecker.mw

Modifying the first two functions is straightforward as they simply compute the energy for the potential of interest using (1.1). The variable *potential* in Opt_procedure.m determines the name of the executable that performs the local optimization. For example, if *potential* ='log', then the program would call trap_dyn_sys_log.exe. Within Opt_procedure.m, one must also change function calls to eval_coul_energy.m. The C++ program sph_vector_functions.cpp is adapted by modifying the methods named *force* and *energy* which is straightforward. The C++ program must then be recompiled and one should change the name of the executable in the Makefile. The calculation of the energy in MinChecker.mw must be changed and exported to .mpl format.

The script given below is set up to run the implementation algorithm 1 in Section 3. To implement algorithm 2, some simple changes must be made:

- Replace both function calls to remove_redundant_dist.m and replace with calls to remove_redundant_coord.m, include *The_comp_powers* as the second input argument.
- Make any necessary changes to the variable *the_tol*. The values used here are given in Sections 3.2.1 and 3.2.2.
- (Optional) A variable called *The_comp_powers* which is n in Eq. (2.14) can be changed, however the value in the script below is the one that was used here.

Changing between the tolerances δ_1 and δ_2 in Eq. (2.13) is done by editing remove_redundant_dist.m. There are two lines where *tol* is computed (the variable is called *the_tol*), one simply selects either δ_1 or δ_2 by commenting/uncommenting the appropriate line.

The file initialConfig_N4.dat contains the coordinates of the 4-particle global minimum as a single column with the first four entries being the θ-coordinates and the last four the ϕ-coordinates.

The following files which are also listed above are complementary functions and return useful data:

- all_loc_min.m
- coordination_numbers.m
- number_loc_min.m
- plot_configuration.m
- point_coordinates.m
- coul_energy

They are straightforward to use and return data on the local minima such as the number of local minima, cartesian and spherical coordinates, coordination numbers, and energies. The function `plot_configuration.m` plots a visual representation of the configuration which also shows the coordination numbers.

A listing of the main program for the Coulomb potential implementing algorithm 1 is given below.

```
close all
clearvars;
clc
format long

formatdate = 'yyyy_mm_dd  HH_MM';
diary(['output/_log  ', datestr(now,formatdate),'.txt']); %log output

keep_temp_files = false; %option to keep temporary input/output files for C prog. Note:
%enabling this option for large computations will generate a large number
%of files.
The_tol= 1e 3; %<0; hence do automatic in extracting redundant configs!
The_comp_powers=[ 3,3];% [ 1]; %for use with extract_really_different_configs.  the vector of powers, n, in the potential
potential = 'coul';  %potential energy function used for optimization   Note:
%this string is important for program dependencies and
%should match the name of the .exe local opt. program.
%i.e trap_dyn_sys_log_track_iter.exe,
%trap_dyn_sys_coul_track_iter.exe, etc.
%Format of Level_Current: matr; each row is {N, E ,X, prev_config, avg_iter, n5};
%       *   N= current number of traps; (Same throughout all rows of Level_Current)
%       *   E energy of current local minimum;
%       *   X=[theta phi x y z] coords of current local minimum
%       *   prev_config = number of the "parent" (N 1) trap local min that this N trap loc min is coming from (1=global min;
    2 next lowest energy;  etc.
%       *   avg_iter = average number of iterations in the local opt.
%           algorithm to achieve this local minimum from prev. N 1 minima
%       *   n5 = number of points with coord number=5 in current local minimum


% %           load to start from N=4
%
% X_START = read_data('input/initialConfig_N4.dat');
% N_START = size(X_START,1);
% N_END=16;   %number of particles to work up to.
% E_START = eval_coul_energy([X_START(:,1) X_START(:,2)]);  %evaluate starting energy
% [n5this,coord5this] = get_coordination5( X_START );
% Level_Current=[{N_START, E_START ,X_START, 1, 0, n5this}];
%
% %format of ALL_LEVELS: 1) N traps; 2) All loc opt configs; 3) M=# local minima (found from 2));
% %4) # different triangle centre configs following from the M loc min's, used to obtain the NEXT (N+1) local minima
% ALL_LEVELS={N_START,Level_Current,1,0};
% runtimes = [];   %execution times for each level.
% %format: column 1) N, column 2) execution time from N to N+1
% %to N+1
% %           load to start from N=4, end


% %           load to continue

N_START=16; %number of particles to start from
N_END=17; %number of particles to work up to

load(['output/N',num2str(N_START,'%03d'),' loc_opt.mat']);
Level_Current=Level_Next;

Lev_num_in_ALLLEV=find(cell2mat(ALL_LEVELS(:,1))==N_START);
Level_Current=ALL_LEVELS{Lev_num_in_ALLLEV,2};
% %           load to continue

%delete(gcp('nocreate'));
%parpool(4);    %change number of workers as desired

for NN=1:N_END N_START    %trap levels

    %   For ALL local minima in the current level: horiz dimension of Level_Current
    N_curr_level_cofigs=size(Level_Current,1);

    Level_Next=cell(1,3);

    N_traps_this_level=N_START+NN 1;

    str1=['***     Going from current level: N=', num2str(N_traps_this_level),' to N+1=',num2str(N_traps_this_level+1),'
        ***'];
    disp(str1);
```

```
str1=['***     (including ', num2str(N_curr_level_cofigs),' locally optimal configuration(s)).     ***'];
disp(str1);

str1=['         *  Computing triangle middles...   *   '];
disp(str1);

all_Nplus1_inserted_triang_middle_coords=cell(0,1); %has all N level configs + all their triangle middles

all_Nplus1_inserted_triang_coming_from=cell(0,1); %remember which config it is arising from!

for LevIdx=1:N_curr_level_cofigs    %locally optimal configs
    %For each locally optimal configuration in current level:
    %   proceed with triangulation

    N_curr=Level_Current{LevIdx,1}; %=N_this_level
    X_curr=Level_Current{LevIdx,3};
    TRI_curr = compute_triangle_middles_D(X_curr);
    N_TRI_curr=size(TRI_curr,1);

    %   init variables
    All_TRI_config_curr=cell(N_TRI_curr,1);

    %   put configuration #LevIdx and inserted triangle middles in a cell
    %array; throw away redundant.

    for i = 1:N_TRI_curr    %triangle middles
        All_TRI_config_curr{i}=[X_curr; TRI_curr(i,:)];
        all_Nplus1_inserted_triang_coming_from=[all_Nplus1_inserted_triang_coming_from; LevIdx];

        TPh=[All_TRI_config_curr{i}(:,1)', All_TRI_config_curr{i}(:,2)'];
    end

    all_Nplus1_inserted_triang_middle_coords = [all_Nplus1_inserted_triang_middle_coords; {All_TRI_config_curr{:}}'];
    %disp(LevIdx);
end
%routines for removing redundant confs expects a certain format and
%number of columns so dummy values must be added
All_Tri_column = 1e20*ones(size(all_Nplus1_inserted_triang_middle_coords,1),1); %dummy column

%   Now remove redundant
cell_en_coord_TRI_CurrLevel=[num2cell(All_Tri_column), all_Nplus1_inserted_triang_middle_coords, ...
    all_Nplus1_inserted_triang_coming_from, num2cell(All_Tri_column), num2cell(All_Tri_column)];

% *select algorithm for removing redundant configs*
%pairwise distances
cell_en_coord_TRI_CurrLevel_nonredundant=remove_redundant_dist( cell_en_coord_TRI_CurrLevel,The_tol);

%   optimize these non redundant configs with inserted triangle middles

tic
n_tri_nonredundant=size(cell_en_coord_TRI_CurrLevel_nonredundant,1);

str1=['           Performing optimizations for ', num2str(n_tri_nonredundant),' sub configurations (triangle middles)...'
     ];
disp(str1);

All_energies_Np1=1e20*ones(n_tri_nonredundant,1);
All_config_Np1=cell(n_tri_nonredundant,1);
All_config_coming_from=cell(n_tri_nonredundant,1);
All_config_n5=cell(n_tri_nonredundant,1);

%preallocate for storing the number of iterations
iter_this_level = cell(n_tri_nonredundant,1);
parfor i = 1:n_tri_nonredundant
    %  for i = 1:n_tri_nonredundant
    X_tmp1 = cell_en_coord_TRI_CurrLevel_nonredundant{i,2};
    inFile = ['temp/N', num2str(N_traps_this_level+1,'%03d'), ' ', num2str(i,'%04d'), ' unopt.txt'];
    outFile = ['temp/N', num2str(N_traps_this_level+1,'%03d'),' ', num2str(i,'%04d'), ' opt.txt'];
    write_data(X_tmp1, inFile);
    weight = 0.5;

    %run local optimization
    res = run_dyn_sys(inFile, outFile, weight, potential); %local optimization

    iter_this_level{i} = str2double(res);
    X_tmp2opt = read_data(outFile);
    if ~keep_temp_files
        delete(inFile,outFile);
    end
```

```
        %evaluate pairwise potential energy funtion
        newEnergy = eval_coul_energy([X_tmp2opt(:,1) X_tmp2opt(:,2)]);

        All_energies_Np1(i)=newEnergy;
        All_config_Np1{i}=X_tmp2opt;
        All_config_coming_from{i}=cell_en_coord_TRI_CurrLevel_nonredundant{i,3};

        [n5this,coord5this] = get_coordination5(X_tmp2opt);
        All_config_n5{i}=n5this;

    end
    runtimes = [runtimes; [N_curr, toc]];
    disp(strcat('Elapsed time: ',num2str(runtimes(end,2))));

    cell_en_coord_Np1=[num2cell(All_energies_Np1), ...
        All_config_Np1, All_config_coming_from, iter_this_level, All_config_n5];

    cell_en_coord_Np1_sortd=sortrows(cell_en_coord_Np1,1);

    %now remove redundant with pairwise distances
    cell_en_coord_Np1_nonredundant=remove_redundant_dist( cell_en_coord_Np1_sortd,The_tol);

    All_TRI_Ns=N_traps_this_level*ones(size(cell_en_coord_Np1_nonredundant,1),1);

    Level_Next=[num2cell(All_TRI_Ns), cell_en_coord_Np1_nonredundant];

    %sort energies
    Level_Next=sortrows(Level_Next,2);

    %Level_Next=cell_en_coord_Np1_nonredundant;

    for ii=1:size(cell_en_coord_Np1_nonredundant,1)
        Level_Next{ii,1}=N_traps_this_level+1;
    end

    ALL_LEVELS{end,4}=n_tri_nonredundant; %add info about how many diff triangle middles from Prev (N trap) config gave rise
         to this N+1 trap config

    %update results
    ALL_LEVELS=[ALL_LEVELS;{N_traps_this_level+1,Level_Next, size(cell_en_coord_Np1_nonredundant,1), 0}];

    %done for this N, save before going to next
    save_File = ['output/N', num2str(N_traps_this_level+1,'%03d'),' loc_opt.mat'];
    save(save_File,'Level_Next','ALL_LEVELS','runtimes');

    Level_Current=Level_Next;

end

diary off;
disp(['... Finished computations starting from N=',num2str(N_START),...
    ' up to N=',num2str(N_END),' for ',potential,'.']);
```

Appendix E. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cpc.2018.03.029.

References

[1] Z. Schuss, Brownian Dynamics at Boundaries and Interfaces, Springer, 2015.
[2] O. Bénichou, R. Voituriez, Phys. Rev. Lett. 100 (2008) 168105.
[3] D. Holcman, Z. Schuss, J. Stat. Phys. 117 (2004) 975.
[4] Z. Schuss, A. Singer, D. Holcman, Proc. Natl. Acad. Sci. 104 (2007) 16098.
[5] A.F. Cheviakov, M.J. Ward, R. Straube, Multiscale Model. Simul. 8 (2010) 836.
[6] A.F. Cheviakov, A.S. Reimer, M.J. Ward, Phys. Rev. E 85 (2012) 021131.
[7] S. Pillay, M.J. Ward, A. Peirce, T. Kolokolnikov, Multiscale Model. Simul. 8 (2010) 803.
[8] A.F. Cheviakov, D. Zawada, Phys. Rev. E 87 (2013) 042118.
[9] T. Lagache, D. Holcman, Phys. Rev. E 77 (2008) 030901.
[10] A.E. Lindsay, A.J. Bernoff, M.J. Ward, Multiscale Model. Simul. 15 (2017) 74.
[11] A.E. Lindsay, T. Kolokolnikov, J.C. Tzou, Phys. Rev. E 91 (2015) 032111.
[12] P.M.L. Tammes, Recueil des travaux botaniques néerlandais 27 (1930) 1.
[13] O.R. Musin, A.S. Tarasov, Experiment. Math. 24 (2015) 460.
[14] T. Erber, G.M. Hockney, J. Phys. A: Math. Gen. 24 (1991) L1369.
[15] H.W. Kroto, et al., Nature 318 (1985) 162.
[16] M.J. Bowick, L. Giomi, Adv. Phys. 58 (2009) 449.
[17] W. Guo, D. Jin, H.J. Maris, J. Phys. Conf. Ser. 150 (2009) 032027.
[18] U. Albrecht, P. Leiderer, Europhys. Lett. 3 (1987) 705.

[19] U. Albrecht, P. Leiderer, J. Low Temp. Phys. 86 (1992) 131.
[20] J. Tempere, I.F. Silvera, J. Devreese, Phys. Rev. Lett. 87 (2001) 275301.
[21] N.J. Sloane, A Library of Arrangements of Points on a Sphere with (Conjecturally) Minimal 1/R Potential, 1997.
[22] T. Erber, G.M. Hockney, Adv. Chem. Phys. 98 (1996) 495.
[23] D. Mehta, J. Chen, D.Z. Chen, H. Kusumaatmaja, D.J. Wales, Phys. Rev. Lett. 117 (2016) 028301.
[24] B. Bergersen, D. Boal, P. Palffy-Muhoray, J. Phys. A: Math. Gen. 27 (1994) 2579.
[25] P.D. Dragnev, Modern Trends in Constructive Function Theory: Conference in Honor of Ed Saff's 70th Birthday: Constructive Functions 2014, May 26–30, 2014, Vandrebilt University, Nashville, Tennessee, vol. 661, American Mathematical Soc., 2016, p. 41.
[26] M.J. Bowick, D.R. Nelson, A. Travesset, Phys. Rev. B 62 (2000) 8738.
[27] J.D. Pintér, Global Optimization in Action: Continuous and Lipschitz Optimization: Algorithms, Implementations and Applications, vol. 6, Springer Science & Business Media, 2013.
[28] R. Horst, P.M. Pardalos, HandBook of Global Optimization, vol. 2, Springer Science & Business Media, 2013.
[29] L. Davis, Handbook of Genetic Algorithms, Van Nostrand Reinhold, 1991.
[30] L. Ingber, B. Rosen, Math. Comput. Model. 16 (1992) 87.
[31] J. Nocedal, Math. Comp. 35 (1980) 773.
[32] T. Erber, G.M. Hockney, Phys. Rev. Lett. 74 (1995) 1482.
[33] H. Cohn, A. Kumar, J. Amer. Math. Soc. 20 (2007) 99.
[34] D. Gomez, A.F. Cheviakov, Phys. Rev. E 91 (2015) 012137.