Acta Crystallographica Section A
Foundations of Crystallography
ISSN 0108-7673

Received 12 December 2007
Accepted 28 August 2008

The Liga algorithm for *ab initio* determination of nanostructure

P. Juhás,[a]*‡ L. Granlund,[a] P. M. Duxbury,[a] W. F. Punch[b] and S. J. L. Billinge[a]‡

[a]Department of Physics and Astronomy, Michigan State University, East Lansing, Michigan 48824, USA, and [b]Department of Computer Science and Engineering, Michigan State University, East Lansing, Michigan 48824, USA. Correspondence e-mail: pj2192@columbia.edu

Computational techniques for nanostructure determination of substances that resist standard crystallographic methods are often laborious processes starting from initial guess solutions not derived from experimental data. The Liga algorithm can create nanostructures using only lists of lengths or distances between atom pairs, providing an experimental basis for starting structures. These distance lists may be extracted from a variety of experimental probes and we illustrate the procedure with distances determined from the pair distribution function. Candidate subclusters that are a subset of a structure's atoms compete based on adherence to the length list. Atoms are added to well performing candidates and removed from poor ones, until a complete structure with sufficient agreement to the length list emerges. The Liga algorithm is shown to reliably recreate Lennard–Jones clusters from ideal length lists and the C_{60} structure from neutron-scattering data. The correct fullerene structure was obtained with experimental data which missed several distances and had loosened constraints on distance multiplicity. This suggests that the Liga algorithm may have robust applicability for a wide range of nanostructures even in the absence of ideal data.

1. Introduction

The nanostructure problem, that of determining the location (and species) of atoms in materials on nanoscale dimensions (Billinge & Levin, 2007), is often resistant to standard crystallographic techniques because of a lack of long-range periodicity in such structures. If the nanoparticles can be coaxed into an orientationally ordered crystal the structure can be solved crystallographically (Jadzinsky *et al.*, 2007), but in general this is very difficult. Obtaining the structure solution using disordered ensembles of nanoparticles is very appealing, but remains a severe challenge in both data collection and its analysis. Multidimensional NMR measurements can be used to solve the structure of small proteins (Crippen & Havel, 1988; Brünger *et al.*, 1998; Herrmann *et al.*, 2002), though a great deal of prior information is used and the final structures do not have high precision. High-quality images of individual nanoparticles with atomic resolution can be obtained using transmission electron microscopy (Wang, 2000; McBride *et al.*, 2004), diffraction imaging with electrons (Zuo *et al.*, 2003) and scanning tunneling microscopy (Helveg *et al.*, 2000). It is not, however, straightforward to go from real-space images of nanoparticles to quantitative structures. In this paper we describe in detail an algorithm that successfully determines quantitative three-dimensional atomic arrangements from the one-dimensional information in the atomic pair distribution function (PDF) (Egami & Billinge, 2003). The PDF is obtained experimentally from neutron or X-ray powder diffraction data from a disordered ensemble of nanoparticles. A preliminary description of this algorithm has already appeared (Juhás *et al.*, 2006).

The PDF has peaks at distances corresponding to the separation between pairs of atoms in the material (Warren, 1990; Egami & Billinge, 2003). The multiplicities of these separations can be found by integrating the areas of the peaks. It is straightforward to calculate the PDF given a three-dimensional structural model, and programs exist for refining structural models given a good initial guess of the solution (Farrow *et al.*, 2007; Tucker *et al.*, 2001). For highly disordered materials such as glasses, unbiased Monte Carlo methods yield structural solutions consistent with the PDF data (McGreevy & Pusztai, 1988). We tried to adopt reverse Monte Carlo (RMC) for structure determination of small well defined structures by fitting to synthetic PDF curves, but this was prohibitively slow for any clusters larger than eight atoms. For example, we were not able to reconstruct a 12-point ($2 \times 2 \times 3$) doubled cube. The RMC procedure was somewhat more successful after we simplified the cost function to fit a discrete set of distances instead of a continuous PDF curve and included a downhill refinement of atom position after each RMC step. These modifications allowed reconstruction of all Platonic solids and 20-atom Lennard–Jones clusters, but only

‡ Current address: Applied Physics & Applied Mathematics, Columbia University, New York, NY 10027, USA

with excessive computational time, and the program remained unable to converge for prism-like shapes.

Owing to these limitations, we have explored different methods to optimize structure shapes with respect to distance data available in the PDF. The best performance so far in terms of speed and convergence rate has been observed with a new method, known as the Liga algorithm because of its use of competition with promotion and relegation similar to European soccer leagues (Juhás *et al.*, 2006). The Liga algorithm can reconstruct clusters of up to ~150 atoms using just the atomic pair distance information available in a one-dimensional PDF measurement. We provide a complete description of the Liga algorithm, and an analysis of its performance with both ideal and experimental data, and discuss some possible extensions.

2. Liga algorithm concepts

The Liga algorithm searches for a structure of N atoms consistent with the supplied list of pair distances. The input distances can be obtained by fitting peaks to the experimental PDF (Egami & Billinge, 2003) or simply calculated in the case of known test structures. The Liga procedure maintains a pool of candidate subclusters (or simply candidates) of each possible size less than or equal to N, separated into divisions labeled $n = 1, 2, \ldots, N$, equal to the subcluster size. These candidates compete with others in the same division *via* a cost function that measures deviation from the length list. Each division contains a fixed number of candidates, typically ten in our trials.

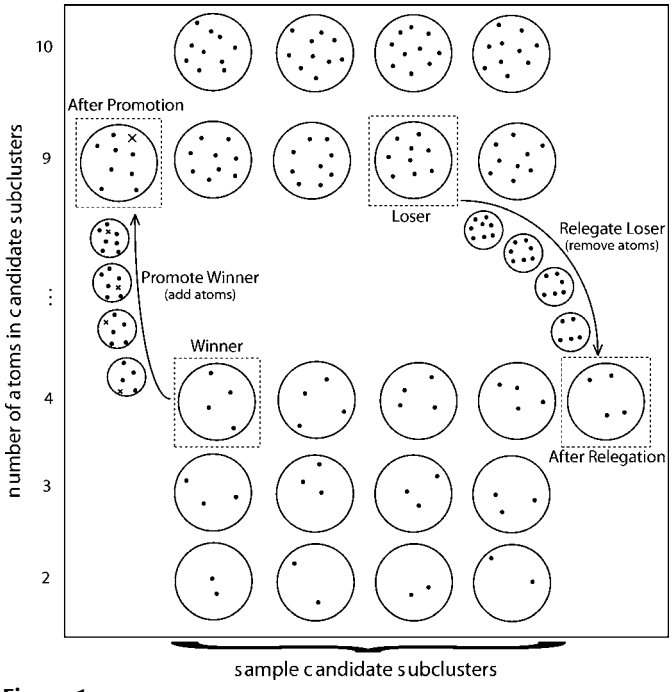

Figure 1
A schematic of promotion and relegation operations in a system with ten divisions, each with four competing clusters, while at the $n = 4$ division. The winning candidate is randomly selected, favoring those with low cost, and it attempts to add as many good atoms as possible (denoted '×'). Assuming the winner is able to add five atoms before exceeding a cost tolerance, it will get promoted to division $n = 9$. A losing candidate is then randomly selected from the $n = 9$ division and five of its atoms are removed, moving it to the winner's original division. The losing candidate and the atoms removed are randomly selected, but favor those with high cost.

The following describes the basic operation of the Liga algorithm. Exceptions and special cases that alter this simple description, such as initialization procedures, are examined in detail in §4.

The Liga algorithm performs iterations called 'seasons', each of which runs through all divisions starting at the lowest level $n_{\min}$ up to the full-sized division N. Often $n_{\min} = 1$, though tests with selected starting subclusters with $n_{\min} > 1$ have also been investigated. At each division, n, a winning candidate subcluster is selected randomly, with relative probability equal to the reciprocal of its cost, from all those in its division. This winning subcluster is promoted by adding one or more atoms to the subcluster. Atoms to be added are selected from a large pool of 'good' test atoms with relative probability equal to the reciprocal of their contribution to the cost of the winning subcluster. The creation of this pool of 'good' test atoms is described below in §3. Atoms from this pool are added until the winning subcluster reaches the maximum size N or the winning subcluster's new cost exceeds a certain tolerance. The number of atoms in the subcluster after the promotion indicates the division that it joins. A losing candidate subcluster is then randomly selected, based on its cost, from the pre-existing candidates at this higher division. The losing structure undergoes a relegation process by removing exactly the number of atoms the promoted subcluster had gained. The removed atoms are selected stochastically with a relative probability equal to their contribution to the loser's total cost. In this way the winning and losing subclusters swap divisions, as illustrated in Fig. 1.

Once the promotion and relegation procedures have been carried out at division n, the algorithm repeats the process at the $(n + 1)$th division, continuing until it eventually reaches division N. Candidate subclusters in the highest division cannot add more atoms, and so the algorithm evaluates whether or not a solution has been found. A candidate at full-size N is declared the solution if its cost is below a user-defined tolerance. If a solution is not found, a new season begins from the lowest division once again.

3. Liga's functions in detail

3.1. Cost function

All competition in the Liga algorithm is resolved probabilistically through weights derived from the cost function. The cost of a candidate subcluster quantizes the degree to which it deviates from the length list. The algorithm also tracks the individual contribution of every atom to the total cost of a subcluster, thus measuring how 'poor' its placement is with respect to other atoms and allowing the prioritization of which

positions should be changed (Bortz *et al.*, 1975; Greene & Supowit, 1986; Altschuler *et al.*, 1994).

The target distance list t_k contains P length values, where $P = N(N-1)/2$ is the number of pairs in an N-atom structure. Distances often appear several times in the list owing to degeneracies in the bond-length values. The cost of a full-sized structure is calculated as

$$C = (1/P)\sum_{k=1}^{P}(t_k - d_k)^2, \quad (1)$$

where t_k and d_k are sorted lists of target and actual distances. The summation over sorted distance sets results in minimum cost C. This can be shown by calculating the cost difference ΔC for swapping the assignment of two model distances,

$$\begin{aligned}\Delta C &= [(t_k - d_l)^2 + (t_l - d_k)^2] - [(t_k - d_k)^2 + (t_l - d_l)^2] \\ &= 2(t_k - t_l)(d_k - d_l). \end{aligned} \quad (2)$$

When target distances are sorted, $t_k > t_l$, and swapping of any unsorted model distances $d_k < d_l$ would result in a lower cost. Therefore the assignment of sorted model to sorted target distances must be the one with the minimum cost C.

In some cases, see §4.6 below, the target distance list may be extended by enlarging the multiplicities of its unique distances so that it contains more lengths than present in a full-sized structure. The cost is then calculated as

$$C = (1/P)\sum_{k=1}^{P}(t^{\rm ext}_{l(k)} - d_k)^2, \quad (3)$$

where $t^{\rm ext}_{l(k)}$ is the extended target distance list and $l(k)$ is an assignment of model distances to the nearest target lengths.

The cost of candidate subclusters with fewer than N atoms is obtained as

$$C = (1/p)\sum_{k=1}^{p}\left(t_{l(k)} - d_k\right)^2, \quad (4)$$

where $p = n(n-1)/2$ is the number of pairs in the candidate. The assignment $l(k)$ maps the model length d_k to the nearest unused target distance $l(k)$. This amounts to labeling each edge in the model subcluster with the length it 'should' have. Because the assignment $l(k)$ is constructed gradually from a subset of distances, it may not be optimal once the structure reaches its full size. Thus to ensure an accurate cost value, the model and target distances are sorted and reassigned every time a candidate cluster grows to the full size.

The individual cost contribution of atom i is expressed through the partial sum

$$c_i = (1/2p)\sum_{j\neq i}^{n}\left(t_{l(ij)} - d_{ij}\right)^2 \quad (5)$$

such that

$$C = \sum_{i=1}^{n} c_i. \quad (6)$$

The fitness of a candidate subcluster, F, and of an atom, f_i, are defined to be the reciprocal of C and c_i, respectively. If the cost value is 0, the fitness is made significantly larger (by a factor of 10) than that of its next-best peer in the same Liga division (for clusters) or cluster (for atoms). If all peers have cost 0, the fitness for each is set to 1.

When selecting a subcluster or atom we refer to 'winners' as those chosen randomly with probability proportional to fitness, and 'losers' as those chosen with probability proportional to cost (Dall & Sibani, 2001; Boettcher & Sibani, 2005).

3.2. Promotion

Promotion is the process of changing the division to which a candidate belongs by adding at least one atom, and more if it appears favorable. The first addition will create n new atomic pairs and uses the n nearest lengths from the target distance table. The distances which are available are only those in the target set t_k that have *not* already been used in the existing candidate subcluster (but see §4.6). The Liga algorithm generates possible positions for new atoms using three different methods:

(1) *Line trials*. This method places new sites in-line with two existing atoms in the cluster. Frequently the existing structures have three or more aligned atoms, and the technique attempts to capture such structure motifs. Two atoms in a subcluster are randomly chosen with a probability based on their fitness, and one unused distance is picked from the target table t_k. The first atom is used as an anchor for the distance, while the second defines the direction towards the new site. When there is only one atom in the structure, the direction is set along the z axis. Two new positions are generated for both orientations of the distance vector.

(2) *Planar trials*. This method accounts for occurrence of atom planes in existing structures. Three atoms are randomly chosen based on their fitness. Two of them are used as the triangle base, while the third one defines the plane of a triangle. When there are fewer than three atoms (or when the chosen atoms form a line) the triangle plane is generated such that the base line defines its angle to the nearest Cartesian plane. Two distances are selected from the available target lengths and are used to construct a triangle vertex. In general four vertexes are possible within a given plane, so this method adds four candidate positions.

(3) *Pyramid trials*. Three atoms in the subcluster are randomly selected based on their fitness to form a base for a pyramid of four atoms. The remaining vertex is constructed using three randomly chosen lengths from the list of target distances. As there are 3! ways of assigning three lengths to three atoms, and because a pyramid vertex can be placed above or below the base plane, this method generates 12 candidate positions.

Each of these methods is repeated many times (typically 10 000 in our trials) to provide a large pool of possible positions for the new atom. For each of the generated sites Liga calculates the associated cost increase for the enlarged candidate and it filters the 'good' positions with the new cost in a cost window $\langle C_{\rm min},\ C_{\rm min} + \varepsilon C_{\rm tol}\rangle$. Here $C_{\rm min}$ is the new candidate cost for the best position found, ε is a user-defined selection fraction ($\varepsilon = 0.1$) and $C_{\rm tol}$ is the allowed cost toler-

ance for the solution (see program parameters *promotefrac* and *tolcost* in the supplemental code description). The atom positions outside the cost window are discarded, ensuring that the better selection probability of a 'good' site is not overwhelmed by a large quantity of poor positions. A winner atom is randomly selected from the remaining set of good atoms. The winner atom is added to the candidate, and it uses n lengths from the target list. The costs of other atoms in the pool are recalculated with respect to the new candidate subcluster and the shortened distance table. If the candidate has fewer than N atoms and there are any atoms inside the cost window, a new winner is selected and added. This can lead to an avalanche of added atoms, potentially reducing the long-term overhead associated with generating larger high-quality candidates.

Every division keeps a record of how many test positions were generated and how many accepted for each of the three placement techniques. These data are used to estimate the success rate of each technique, and the more successful methods get proportionally preferred when placing new atoms at a particular size. We assume the number of accepted positions generated by method m has a binomial distribution with probability p_m. If method m has created n_m positions, of which s_m have been selected, the probability p_m is known with the beta distribution $B(\alpha, \beta)$, where $\alpha = s_m + 1$ and $\beta = n_m - s_m + 1$. The beta distribution is conjugate prior to the binomial distribution, and so the program estimates the success rate p_m by generating a random number from the appropriate beta distribution. Given total number of test positions t to allocate for an enlarged cluster, the method m will be used $t \times (p_m / \sum_i p_i)$ times.

3.3. Relegation

The relegation operation reassigns a candidate subcluster to a lower division $n' = n - m$ by removing m loser atoms. The loser atoms are randomly selected according to their cost from all the atoms in the cluster. After their removal the target lengths associated with the destroyed atom pairs are returned to the target list of distances.

4. Special cases in the Liga algorithm

Several special cases can alter the simple behavior of the Liga algorithm. In order to keep the description in §2 focused on the fundamental concepts, we have delayed their discussion until now.

4.1. Initial conditions

The algorithm may be initialized with a single copy of a trivial one-atom structure in the lowest division or a non-trivial cluster of larger size, $n_{\min} > 1$. Any existing lower divisions (*i.e.*, $n_{\min} > n > 1$) are initialized with relegated copies of the initial structure. A non-trivial starting structure can be particularly useful for cases in which a quality guess or known substructure already exists.

Each division is set to contain a fixed number of candidates, but at the beginning they are completely or partially empty. When a winner for promotion is selected from a division that is not full it adds a copy of itself to that division in addition to being promoted. Similarly, when a loser is selected for relegation from a division that is not full it adds a copy of itself to that division before being relegated. Finally, after a winner is promoted it checks to see if there are any empty divisions below its new division. If this is the case then it adds an appropriately relegated clone of itself to those empty divisions.

4.2. Fixed subclusters

By default the Liga algorithm creates divisions of size 1 to N and runs through each. However, if the unknown structure contains a known subcluster of $n_{\min}$ atoms, it is neither necessary nor meaningful to consider candidates of this or smaller sizes. A user may specify some or all of the atoms in an initial structure as fixed. The Liga competitions are then performed only for divisions $n_{\min}$ to N, and the $n_{\min}$ fixed atoms are kept in all subclusters. Fixed subclusters can be advantageous for systems containing rigid chemical subunits such as aromatic rings or fourfold silicon sites.

4.3. Very good promotions

A winning candidate's total cost after promotion is often greater than that of the losing candidate which is to be relegated. The Liga algorithm attempts to retain well performing candidates so when a winner's cost is lower, the loser is replaced with a clone of the promoted winner. The loser is discarded from the collection of candidate subclusters. This operation has an analogy in sports competitions, where a poorly performing team may adopt the 'game' of its better opponents.

4.4. Promotion of best candidate

Promotion often removes the fittest candidate from a division, which can be quite detrimental to the division's overall fitness if the best remaining candidate is extremely costly by comparison. Before promoting a winning candidate the Liga algorithm determines whether it has the lowest cost in its division. If this is so, after promotion and relegation have been completed it compares this cost to that of the new best candidate in the division. If that candidate has a significantly higher cost (*e.g.*, by a factor of 10) then the division selects a loser candidate and replaces it with a copy of the original (pre-promotion) winner. The loser is then completely removed from the collection of candidate subclusters.

4.5. Skipping unfavorable promotions

Some candidates have such high cost that it is extremely unlikely that promoting them will introduce better structures that can be retained. This is particularly true early on, when few if any good candidates populate the upper divisions. Since promotion is a computationally expensive procedure, it may be better to not promote such candidates even if they have

won the division. Thus, after a winning candidate has been selected, but before the promotion process begins its cost is compared to a user-defined maximum cost. If it exceeds this limit the algorithm skips to the next division without performing promotion or relegation.

4.6. Loose multiplicity constraints

When target distance lists are obtained from experimental PDF data, they may contain significant errors in the multiplicity of unique distances. This is a problem especially because underestimated multiplicities may greatly increase the cost value of the correct structure. Under such circumstances it is advantageous to relax adherence to distance multiplicities or even ignore multiplicities altogether, as done in §§5.2 and 5.3 below. In the first case, the target distance table $t_l^{\rm ext}$ is constructed with distance multiplicities increased by a fixed percentage, and thus it contains more lengths than actually present in the searched structure. In the second case, the program allows any target distance to be compared with the model structure an arbitrary number of times. This is in effect the same as setting infinite multiplicities for all lengths in the target set t_k.

5. Results

Algorithm performance was benchmarked for several structures using both experimental and ideal distance lists. All data were obtained on a high-performance cluster running 2.3 GHz quad-core Intel Xeon processors. The supplementary material includes C++ source code, length lists for all the structures reported below and the algorithm parameter values used to generate these results.[1]

Owing to the stochastic nature of the Liga algorithm, the CPU times required for convergence to a given cost tolerance can vary by as much as an order of magnitude. To obtain a meaningful assessment of the algorithm speed, each benchmark has been repeated at least 100 times using the same algorithm parameters but different seeds of the random-number generator. This reduced the standard deviation of the average run time by a factor of 10 and thus allowed determination of the effects of algorithm parameters on the solution time. The Liga algorithm uses the MT19937 random-number generator (Matsumoto & Nishimura, 1998), which is the default algorithm provided by the GNU Scientific Library (GSL) (Galassi *et al.*, 2006). Fig. 2 shows a histogram of structure solution times from 900 benchmark runs on ideal distances from a 50-atom Lennard–Jones (LJ-50) cluster. The distribution has an asymmetric shape with a slowly decaying tail for long solution times. To account for this, the standard deviation of run times was calculated separately for those times lower and higher than the average, and both low and high standard deviations are reported. Fig. 2 indicates the mean solution time and its low and high deviations in the histogram of solution times for LJ-50 clusters.

5.1. Lennard–Jones clusters

Lennard–Jones clusters are minimum-energy configurations of N atoms that interact *via* the Lennard–Jones (LJ) pair potential. These structures are often used for benchmarking energy-minimization algorithms (Deaven *et al.*, 1996; Wales & Scheraga, 1999; Cai & Shao, 2002). In our study we used the published coordinates of LJ clusters (Wales & Doye, 1997; Hartke, 1999) to calculate ideal distance lists and feed them back into the Liga algorithm to reconstruct the original structure.

In the first step we have run the code on a few clusters at several different sizes up to 150 atoms. All of these runs converged successfully to a structure with final cost below 0.0001 Å^2 and with the same atomic coordination number for each atom as the correct solution. The atomic coordination properties of a cluster were assessed by calculating the coordination histogram, which counts the number of atoms in the cluster that have a given coordination number.

In the following studies we evaluated the effects of algorithm parameters on the time required to solve LJ clusters. While the optimum parameters may vary widely for different structures, these studies allow us to assess the general impact of each parameter on performance and tune the algorithm's parameters toward improved performance.

The first parameter studied was *seasontrials*, the number of atom positions generated in a complete season. Since generation of atom positions and evaluation of their costs is, computationally, the most expensive operation in the Liga algorithm, *seasontrials* also provides a measure of CPU time required for one season of competition. The placement trials are shared among all divisions in Liga, and large values of *seasontrials* allow for a thorough search for low-cost atoms to add during the subcluster promotion procedure. When, however, *seasontrials* is too large it can slow down the

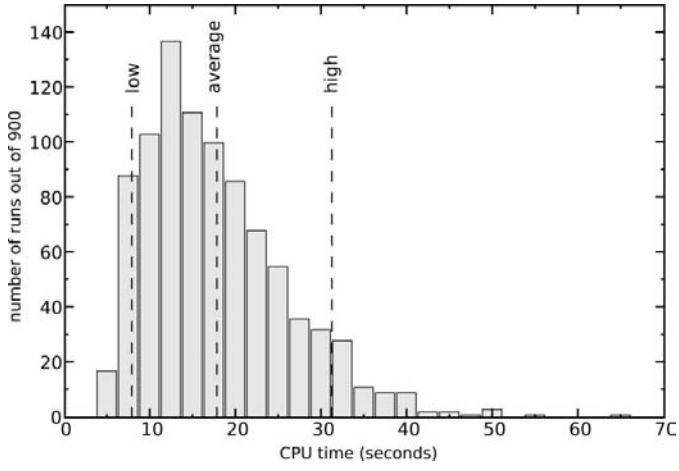

Figure 2
Histogram of run times required for reconstruction of the LJ-50 cluster from ideal distances. All runs used equivalent settings with the exception of the random-number-generator seed.

[1] Supplementary data for this paper are available from the IUCr electronic archives (Reference: MK5018). Services for accessing these data are described at the back of the journal.

Table 1
Optimum values of *seasontrials*.

These values were obtained from a parameter study and their values were modeled by equation (7) with $A = 20$.

	seasontrials	
Cluster	Ideal	Modeled
LJ-1	1	0
LJ-30	16000	8700
LJ-50	20000	24500
LJ-70	50000	48300

performance, because a more limited number of Liga seasons can then be carried out. On the contrary, when *seasontrials* is small, the number of played seasons becomes large, but the promoted subclusters have lower quality because the search for good configurations is too superficial.

The impact of *seasontrials* was measured for Lennard–Jones clusters of 30, 50 and 70 atoms, as shown in Fig. 3. Each of these clusters was solved for a set of *seasontrials* values ranging from 2×10^3 to 10^7. All three curves display a minimum at an optimal value of *seasontrials*, and this value increases with the size of the cluster. We have fitted the optimum values of *seasontrials* to the empirical formula

$$seasontrials = AP = AN(N-1)/2. \qquad (7)$$

This formula has been chosen because P, the number of pairs in the structure, is also the number of constraints the solution needs to satisfy and it makes sense to increase the times for promotion search at the same rate. Table 1 lists the observed optimum *seasontrials* together with the values modeled by equation (7), where the constant A was optimized to $A = 20$. This equation provides a useful estimate of the location of the minimum, but should not be considered fully quantitative.

In the next study we have evaluated the effect of the *ligasize* parameter, which specifies the size of a Liga division (*i.e.*, the number of competitors at each possible cluster size). The benchmarks were performed for the LJ-70 cluster with the value of *seasontrials* set to its near-optimum value of 50 000. Fig. 4 compares the run times for the values of *ligasize* ranging from 1 to 10. The observed run times remained essentially the same as the number of competitors was decreased from 10 to 3. For two competitors the average solution time was increased by 50% and for *ligasize* = 1 it was inflated by three orders of magnitude to more than 2 days. The average solution time shows a broad minimum value at around five competitors.

The sharp decline in performance for small *ligasize* was expected, because with a tiny population of candidates the algorithm cannot distinguish good candidates from faulty ones. Another consequence of this limit is that the program stores fewer cost points in the phase space of all possible structures. For the values of *ligasize* exceeding 5, however, the average run times also slightly increased. This may be caused by the fact that the total winning probability of the best candidate decreases with the number of competitors. In other words, if a Liga division contains one excellent candidate, a large number of average competitors make it less likely for this candidate to be selected for promotion. However, this effect is very moderate at the tested values of *ligasize* and only distinguishable in the average times. The standard deviations of observed times are large, and for practical purposes the expected run times are very similar for all *ligasize* values ranging from 3 to 10.

In the third study we compared the performance for LJ clusters sized 10, 20, 30, ..., 100. These runs were performed using optimum values of *seasontrials* as modeled from Table 1. Fig. 5(*a*) displays run times *versus* the size of the modeled clusters. The solution times increase monotonically with the

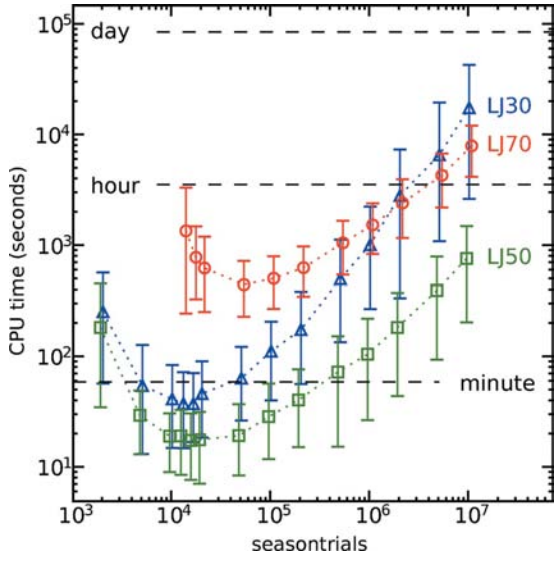

Figure 3
Liga run time for LJ clusters of size 30, 50 and 70 as a function of *seasontrials*, the number of atom positions generated in a complete season. The positions of error bars are slightly shifted to reduce their overlap.

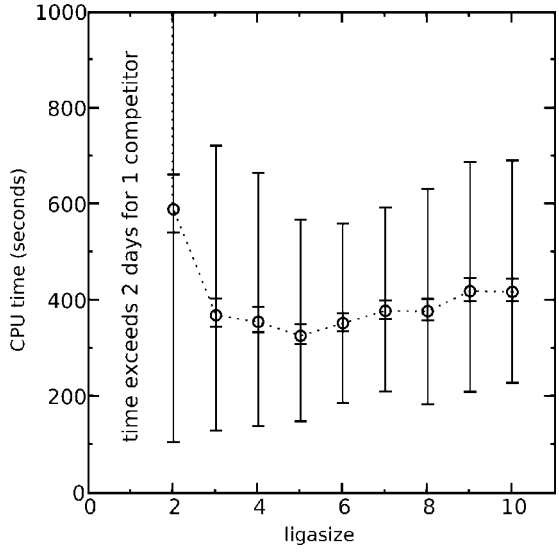

Figure 4
Solution times for the LJ-70 cluster *versus* the number of competitors at the same size (parameter *ligasize*). Each data point comes from 100 repeated runs with different random-number-generator seeds. The tighter error bars denote standard deviations of the average times.

exception of large spikes for LJ-30 and LJ-80. The origin of this trend can be better understood by introducing the concept of the per-distance entropy of the input distances lists, Fig. 5(*b*). The entropy of the distance data was calculated from the number of possible assignments of P distances to P atom pairs by using

$$S_{\rm d} = \frac{1}{P}\log\frac{P!}{m_1!m_2!\ldots m_l!} \simeq \frac{1}{P}\left[P\log(P) - \sum_i m_i \log m_i\right], \tag{8}$$

where m_i is the multiplicity of each unique distance. The multiplicities m_i were evaluated by re-binning the distance list with a step of 0.02 Å and counting distances in each bin. The value of the bin size was chosen according to the target simulation cost, which had to decrease below 0.0001 Å^2; this value corresponds to a standard deviation between data and model distances of 0.01 Å. This analysis suggests that distance lists with many unique distances are harder for Liga to solve, while structures with short lists of unique distances are solved more efficiently.

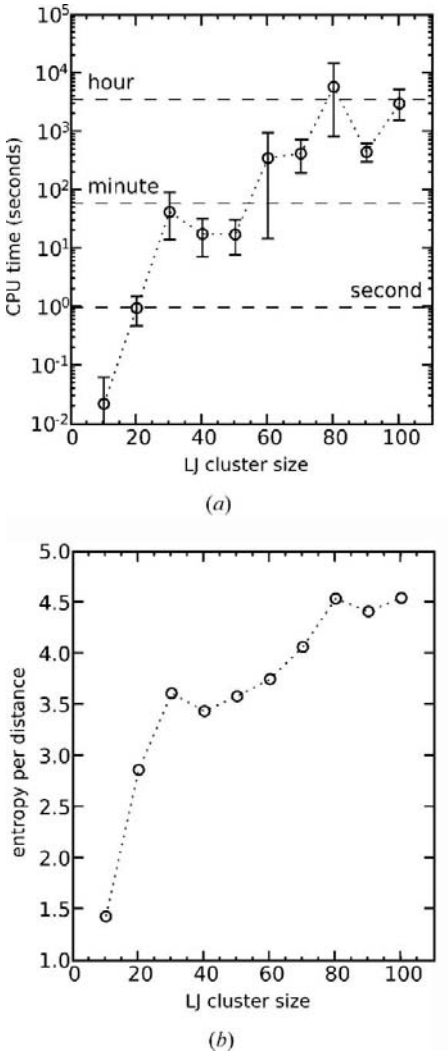

Figure 5
(*a*) Average run time of LJ-*n* solution as a function of structure size *n*. The low-value error is in fact smaller than the high-value deviation, but displays longer owing to the logarithmic scale. Simulations were performed 100 times at each structure size. (*b*) Entropy per distance of input distance lists.

Table 2
Liga performance for C_{60} reconstruction from ideal and experimentally measured distance data.

The required solution cost was 0.0001 Å^2 for ideal and 0.00345 Å^2 for neutron PDF distances. $t_{\rm CPU}$ is the average time taken to reach convergence and σ_t gives standard uncertainties for run times below and above $t_{\rm CPU}$.

Data type	Trials	$t_{\rm CPU}$ (s)	σ_t (s)
Ideal	225	1.6	−1.1, +1.7
Neutron PDF	225	1100	−650, +930

5.2. C_{60} nanostructure

The Liga algorithm successfully converges to the C_{60} structure using ideal distances as well as those derived from neutron PDF data (Juhás *et al.*, 2006). A summary of these results is given in Table 2, with 100% of the attempted trials resulting in convergence to a truncated icosahedral shape.

For angle-averaged neutron and X-ray scattering data, the pair distances and their multiplicities can be extracted by fitting Gaussian profiles to the experimental radial distribution function (RDF), $R(r)$ (Egami & Billinge, 2003). The $R(r)$ function is obtained from $G(r)$, the measured PDF, after background subtraction as

$$R(r) = r[G(r) - G_{\rm bg}]. \tag{9}$$

In the case of C_{60}, the background $G_{\rm bg}$ was modeled by a piece-wise linear function with a single break. This is a correct particle–particle correlation function for randomly oriented spherical shells, as shown in Thorpe *et al.* (2002). The pair distances and multiplicities were obtained using a simplified approach, where the distances were assigned to peak maxima and shoulders on the leading or trailing edges of peaks of $R(r)$, Fig. 6(*a*). The shoulder positions were set at the inflection points on the peak slopes, where the first derivative has either a positive local minimum (leading edge) or a negative local maximum (trailing edge). The multiplicities of the corresponding distances were estimated by numerical integration of the RDF, where integration limits were set either at the peak foot or, in the case of a shoulder position, at midpoints defined by the ratio of RDF amplitudes. The length list extracted by this procedure had 14 unique lengths rather than the 21 of the ideal structure, Fig. 6(*b*). Despite these deviations in the extracted distance list, a correct structure solution was obtained without the use of any prior structure information.

The extracted multiplicities of C_{60} distances contain errors due to unresolved peak overlap and noise in the data. The target table contained too few distances at some values while others had excessive multiplicities. These errors in multiplicity cause convergence to a faulty structure when run with a *tight* distance list of 1770 lengths (the number of pairs in a 60-atom cluster). In fact, the cost of the faulty structure with respect to

the *tight* data was lower than for the correct solution (Juhás *et al.*, 2006). To overcome such errors, the multiplicity of every unique distance has been increased by 10%, and for such *loose* distance lists the algorithm converged to the correct structure. The effect of distance multiplicities on the solution is discussed in detail in §5.3 below.

Fig. 7 shows the development of the eventual solution (winner) in a typical C_{60} simulation from neutron PDF data. The solution cluster moves between sizes 45 to 55 as its cost gradually decreases. Adjacent seasons where the cost changes but the size does not are indicative of the very good promotions described in §4.3. At size 55, there is significant backtracking to a smaller size of about 40 atoms. This backtracking is through the relegation process and is essential to finding the solution.

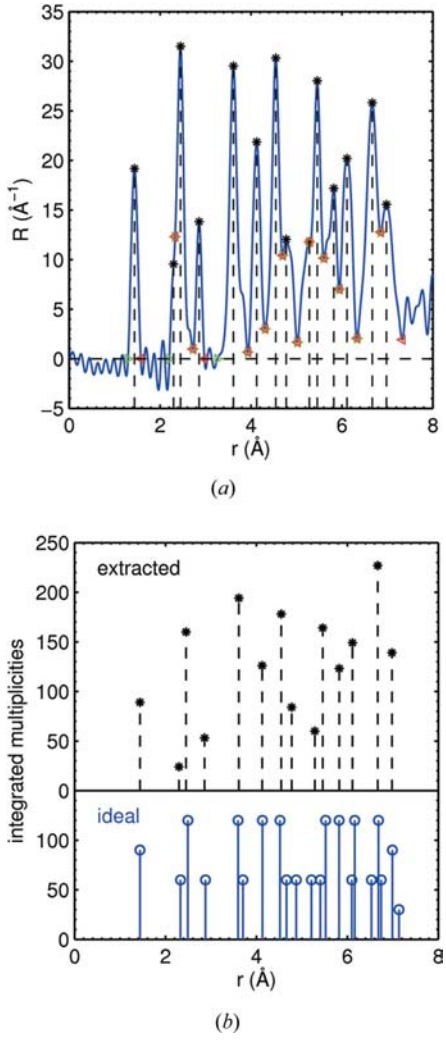

Figure 6
(*a*) Extraction of target distance table from neutron RDF data for C_{60}. The experimental distances were obtained from peak and shoulder positions (asterisks) and their multiplicities by integrating peak areas. The integration limits marked by red and green triangles were set at peak feet or proportionally to the RDF amplitude for the overlapping peaks. Adapted from Juhás *et al.* (2006) Fig. 2(*b*) using an updated model for interparticle correlations. (*b*) Comparison of extracted target distances with the ideal distance list.

5.3. Effect of loosening multiplicity constraints

For a successful solution of C_{60} from neutron PDF data the extracted distance list had to be loosened by increasing all distance multiplicities by 10%. This 'relaxation' of the distance constraints allowed the program to overcome errors from incorrectly estimated multiplicities, which would otherwise prevent convergence to the correct answer. In this case it was more important to fit the high-precision distance values than to satisfy their high-error multiplicities. In fact, for several shapes the multiplicities can be ignored (made arbitrarily large), and the correct structures are obtained from the set of the unique distance values alone. The simplest example of this effect is a tetrahedron, which is the largest three-dimensional shape containing just one pair-distance value; any placement of a separate fifth atom would create a different distance. A similar effect occurs with the ideal buckyball. Fig. 8 shows the cost–size dependence for the best clusters found using only the

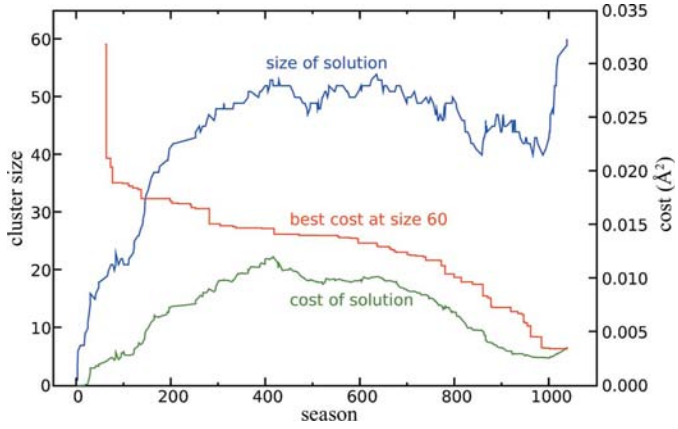

Figure 7
The size and cost of the eventual solution as it converges to the C_{60} nanostructure. The cost of the best structure of size 60 found at each season is included for comparison.

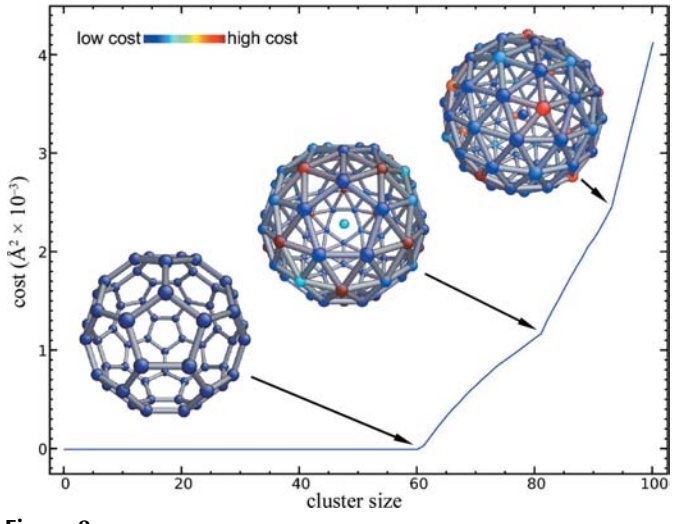

Figure 8
Minimal cost of structures found at various structure sizes using the ideal C_{60} distances and unconstrained multiplicity. In addition to the buckyball, the insets show examples of such structures for 81 and 93 atoms.

21 unique distances in the ideal buckyball. The best cost remains at 0 up to size 60, and thus the buckyball seems to be the largest shape consistent with exactly those distances. Addition of an atom to make a structure of size 61 incurs a non-zero cost and the cost continues to increase with further atom addition. This intriguing behavior presents an interesting mathematical puzzle which is beyond the scope of this study.

6. Discussion and conclusions

Although the Liga algorithm uses some strategies characteristic of other optimization algorithms, it modifies them in unique ways in order to solve the nanostructure problem. Liga shares aspects with techniques such as reverse Monte Carlo (McGreevy & Pusztai, 1988), and a variety of other strategies based on simulated annealing (Bortz *et al.*, 1975; Kirkpatrick *et al.*, 1983; Dall & Sibani, 2001) and evolutionary algorithms (EAs) (Goldberg, 1989; Deaven *et al.*, 1996; Hartke, 1999), which calculate the energy or cost associated with changing the position of an atom in a candidate structure. The Liga algorithm, however, is unique in that it keeps a pool of 'good' atom positions for the promotion process and can use a string of promotions to rapidly generate larger high-quality structures.

Liga maintains multiple competitors of each subcluster size, which is typical of some EA solvers and genetic algorithms (GAs) that modify the problem being solved into levels of difficulty (Cantu-Paz, 2001); for example the injection architecture (Lin *et al.*, 1994) creates multiple levels in a problem. Liga is unique, however, in that it not only promotes candidate clusters to higher levels (closer to the final solution), but also relegates solutions from a higher level to a lower level. The relegation strategy is useful in that it keeps the best of both worlds: promoted candidates provide an improved starting point for finding better, more complete, subclusters while relegated candidates are disassembled in a way that maintains improved subclusters as competitors in later Liga seasons.

Perhaps the most remarkable aspect of the Liga algorithm is that it finds nanostructures without the use of any information other than the distance list. This can be compared to the NMR structure solution of soluble proteins, where a great deal of prior knowledge and, most importantly, assignment is carried out prior to reconstruction (Brünger *et al.*, 1998; Herrmann *et al.*, 2002). It is not obvious, *a priori*, that sufficient information exists in the distance list to carry out a reconstruction, since the problem we seek to solve amounts to inverting scattering data to find a structure. The distance-list formulation of this problem, as described here, removes the traditional 'phase' problem, but introduces its own computational complexities *via* the large number of ways of placing $N(N-1)/2$ distances between N atoms. This problem is not without mathematical precedents (Hendrickson, 1995), but there are no mathematical theorems that guarantee that a unique nanostructure can be found from its distance list. In contrast, some simple structures, such as the hexagon, are known to be degenerate with other structures which have the same distance lists; however, adding even a single additional atom usually removes this degeneracy.

At a more practical level, the promise of Liga can only be fully realized if it is robust to experimental uncertainties and if it can be generalized to find the correct chemical placements of atoms in binary and more complex nanostructures. Liga has already demonstrated some degree of robustness by reconstructing the correct structure of fullerenes from noisy neutron-scattering data. The addition of prior constraints should be very helpful in reconstructing more difficult structures, or those with data suffering from increased noise. Moreover, we have carried out preliminary studies of the generalization of Liga to placement of chemical species in binary and more complex systems. One approach, divide and conquer, follows straightforwardly from the results presented in this paper. In this approach we use the fact that the geometry of candidate subclusters may be generated using only the unique distance list while allowing each distance to occur an arbitrary number of times in the structure. Although we cannot expect this procedure to always identify the correct geometry, we do expect it to create a set of low-cost candidates for the second stage of the procedure, the placement of chemical species. The second stage of the divide and conquer procedure assigns chemical species to the known atom sites and finds the cost of each placement. At this stage information about viable chemical structures and coordinations can be used. This information can be provided by techniques such as EXAFS and solid state NMR, as well as prior knowledge. Thus,
both the robustness and chemical-placement issues appear to have viable solution paths, suggesting that the Liga algorithm can form the basis of broad and robust methods for generating high-quality candidates for nanostructure refinement.

An area where Liga performs more poorly is when the distance list contains a very large number of unique distances. We have further tested this behavior using completely random point sets which have $N(N-1)/2$ unique distances, confirming that Liga performs poorly for these rather unphysical cases. Nevertheless, the physical origin of this reduced performance is the fact that these random structures have a very large number of local minima which compete with the true minimum of the reconstruction. Liga has difficulty navigating this phase space as there are a very large number of 'good' atom placements that do not appear in the global minimum. This problem can be expected to occur in very low symmetry nanostructures and remains an important additional challenge for the future. Nevertheless, the algorithm was still successful for medium-symmetry structures, such as the 112-atom unit cell of the Peierls charge-density-wave material $CeTe_3$ (Kim *et al.*, 2006) or the 71-atom functionalized C_{60}-*N*-methylpyrrolidine (Sun *et al.*, 1997), which were solved from generated distance data. The symmetry of these structures is much lower than for C_{60}, as reflected in their distance entropies S_d, equation (8), which are 4.2 and 5.7, compared to S_d = 3.0 for C_{60}. Another area where we expect the procedure to be applicable is the determination of local structure deviations

from an average crystallographic lattice, such as those that occur in ferroelectric perovskites, (Dmowski *et al.*, 2000; Juhás *et al.*, 2004).

In conclusion, the detailed benchmarking and tests we have presented indicate that the Liga procedure can reconstruct noncrystalline motifs from high-quality scattering data alone. Generalization of the current Liga strategies to binary and higher-order systems and low-symmetry structures, and the need for a robust response to experimental uncertainties remain a challenge, though viable approaches for resolving these issues do not appear intractable.

We gratefully acknowledge Drs A. P. Ramirez and R. C. Haddon for supplying the C_{60} sample. PJ appreciates useful discussions with Drs J. Bloch and E. S. Božin. PMD and LG were supported by the MSU Center for Nanomaterials Design and Assembly (CNDA). SJLB and PJ acknowledge support from the Distributed Analysis of Neutron Scattering Data (DANSE) project, which is funded by the US National Science Foundation (NSF) under DMR-0520547. Neutron data were collected at the Intense Pulsed Neutron Source (IPNS) at Argonne National Laboratory, which was supported by the US Department of Energy, Office of Science, Office of Basic Energy Sciences, under contract No. DE-AC02-06CH11357.

References

Altschuler, E. L., Williams, T. J., Ratner, E. R., Dowla, F. & Wooten, F. (1994). *Phys. Rev. Lett.* **72**, 2671–2674.
Billinge, S. J. L. & Levin, I. (2007). *Science*, **316**, 561–565.
Boettcher, S. & Sibani, P. (2005). *Eur. Phys. J. B*, **44**, 317–326.
Bortz, A., Kalos, M. & Lebowitz, J. (1975). *J. Comput. Phys.* **17**, 10–18.
Brünger, A. T., Adams, P. D., Clore, G. M., DeLano, W. L., Gros, P., Grosse-Kunstleve, R. W., Jiang, J.-S., Kuszewski, J., Nilges, M., Pannu, N. S., Read, R. J., Rice, L. M., Simonson, T. & Warren, G. L. (1998). *Acta Cryst.* D**54**, 905–921.
Cai, W. S. & Shao, X. G. (2002). *J. Comput. Chem.* **23**, 427–435.
Cantu-Paz, E. (2001). *J. Heuristics*, **7**, 311–334.
Crippen, G. M. & Havel, T. F. (1988). *Distance Geometry and Molecular Conformation.* New York: Wiley and Sons.
Dall, J. & Sibani, P. (2001). *Comput. Phys. Commun.* **141**, 260–267.
Deaven, D., Tit, N., Morris, J. & Ho, K. (1996). *Chem. Phys. Lett.* **256**, 195–200.
Dmowski, W., Akbas, M. A., Davies, P. K. & Egami, T. (2000). *J. Phys. Chem. Solids*, **61**, 229–237.
Egami, T. & Billinge, S. J. L. (2003). *Underneath the Bragg Peaks: Structural Analysis of Complex Materials.* Oxford: Pergamon Press/Elsevier.
Farrow, C. L., Juhás, P., Liu, J. W., Bryndin, D., Božin, E. S., Bloch, J., Proffen, T. & Billinge, S. J. L. (2007). *J. Phys. Condens. Matter*, **19**, 335219.
Galassi, M., Davies, J., Theiler, J., Gough, B., Jungman, G., Booth, M. & Rossi, F. (2006). *GNU Scientific Library Reference Manual*, 2nd ed. Bristol: Network Theory Ltd. http://www.gnu.org/software/gsl/.
Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. Boston: Addison-Wesley.
Greene, J. & Supowit, K. (1986). *IEEE Trans. Comput. Aid. Des.* **5**, 221–228.
Hartke, B. (1999). *J. Comput. Chem.* **20**, 1752–1759.
Helveg, S., Lauritsen, J. V., Laegsgaard, E., Stensgaard, I., Norskov, J. K., Clausen, B. S., Topsoe, H. & Besenbacher, F. (2000). *Phys. Rev. Lett.* **84**, 951–954.
Hendrickson, B. (1995). *SIAM J. Optim.* **5**, 835–857.
Herrmann, T., Güntert, P. & Wüthrich, K. (2002). *J. Mol. Biol.* **319**, 209–227.
Jadzinsky, P. D., Calero, G., Ackerson, C. J., Bushnell, D. A. & Kornberg, R. D. (2007). *Science*, **318**, 430–433.
Juhás, P., Cherba, D. M., Duxbury, P. M., Punch, W. F. & Billinge, S. J. L. (2006). *Nature (London)*, **440**, 655–658.
Juhás, P., Grinberg, I., Rappe, A., Dmowski, W., Egami, T. & Davies, P. (2004). *Phys. Rev. B*, **69**, 214101.
Kim, H. J., Malliakas, C. D., Tomic, A., Tessmer, S. H., Kanatzidis, M. G. & Billinge, S. J. L. (2006). *Phys. Rev. Lett.* **96**, 226401.
Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. (1983). *Science*, **220**, 671–680.
Lin, S. C., Punch, W. & Goodman, E. (1994). *Sixth IEEE Symposium on Parallel and Distributed Processing*, pp. 28–37. Dallas: IEEE Computer Society Press.
McBride, J., Kippeny, T., Pennycook, S. & Rosenthal, S. (2004). *Nano Lett.* **4**, 1279–1283.
McGreevy, R. L. & Pusztai, L. (1988). *Mol. Simul.* **1**, 359–367.
Matsumoto, M. & Nishimura, T. (1998). *ACM Trans. Model. Comput. Simul.* **8**, 3–30.
Sun, Y. P., Drovetskaya, T., Bolskar, R. D., Bau, R., Boyd, P. D. & Reed, C. A. (1997). *J. Org. Chem.* **62**, 3642–3649.
Thorpe, M. F., Levashov, V. A., Lei, M. & Billinge, S. J. L. (2002). *From Semiconductors to Proteins: Beyond the Average Structure*, edited by S. J. L. Billinge and M. F. Thorpe, pp. 105–128. New York: Kluwer/Plenum.
Tucker, M. G., Dove, M. T. & Keen, D. A. (2001). *J. Appl. Cryst.* **34**, 630–638.
Wales, D. J. & Doye, J. P. K. (1997). *J. Phys. Chem. A*, **101**, 5111–5116.
Wales, D. J. & Scheraga, H. A. (1999). *Science*, **285**, 1368–1372.
Wang, Z. L. (2000). *J. Phys. Chem. B*, **104**, 1153–1175.
Warren, B. E. (1990). *X-ray Diffraction.* New York: Dover.
Zuo, J. M., Vartanyants, I., Gao, M., Zhang, R. & Nagahara, L. A. (2003). *Science*, **300**, 1419–1421.