Global Cluster Geometry Optimization by a Phenotype Algorithm with Niches: Location of Elusive Minima, and Low-Order Scaling with Cluster Size

BERND HARTKE

Institut für Theoretische Chemie, Universität Stuttgart, Pfaffenwaldring 55, D-70569 Stuttgart, Germany

Received 12 July 1999; accepted 18 August 1999

ABSTRACT: The problem of global geometry optimization of clusters is addressed with a phenotype variant of the method of genetic algorithms, with several novel performance enhancements. The resulting algorithm is applied to Lennard–Jones clusters as benchmark system, with up to 150 atoms. The well-known, difficult cases involving nonicosahedral global minima can be treated reliably using the concept of niches. The scaling of computer time with cluster size is approximately cubic, which is crucial for future applications to much larger clusters. © 1999 John Wiley & Sons, Inc. J Comput Chem 20: 1752–1759, 1999

Keywords: global geometry optimization; clusters; genetic algorithms; Lennard–Jones; size scaling

Introduction

Clusters continue to challenge experimentalists and theoreticians alike.[1–3] Experiments focus on basic questions like changes of properties with cluster size and magic numbers, but they also link clusters to areas as disparate as chemical vapor deposition,[4] climatology,[5–7] and nanotechnology.[8–11] Before theory can address such issues, it has to solve simpler problems, starting with the structure of clusters. Naively, one may assume that the cluster geometry most likely to be found experimentally correponds to the global minimum of the potential energy surface in the configuration space of the cluster. Actually, in many experimental situations, it is likely that a mixture of thermodynamic and kinetic factors determines cluster formation and structure, as in the delicate case of the water hexamer.[12–15] Nevertheless, cluster geometries corresponding to the global energy minimum or to low-lying local minima are not only likely candidates for the most probable

Correspondence to: B. Hartke; e-mail: hartke@theochem.uni-stuttgart.de

Contract/grant sponsors: Deutsche Forschungsgemeinschaft (DFG) and Fonds der Chemischen Industrie (FCI)

structure, but they are also indispensable for our understanding of the features of the potential energy surface.[16,17] Furthermore, locating these minima should be less demanding than a microscopic simulation of possible cluster formation processes.

It has been shown that a discretized version of the problem of finding the global minimum geometry of clusters belongs to the class of NP-hard problems.[18,19] This suggests that the (computer) time necessary for an (exact) solution will increase exponentially with cluster size. As discussed in ref. 20, this does not imply, however, that certain special cases of the general problem also need exponential time, for example, particular cluster sizes or particular potentials used to describe interparticle forces in the cluster. Therefore, the search for an efficient global geometry optimization method for clusters remains a valid research goal.

A suitable benchmark system for method development in this area is that of Lennard–Jones clusters, where interparticle forces are solely described by the Lennard–Jones pair potential. This system has been studied for decades;[21–25] a list of global minimum energies and structures is available on the internet.[26] A large number of global optimization methods has been applied to Lennard–Jones clusters (see the recent review ref. 20), with the following main contenders.

Potential energy surface deformation methods reduce the number of local minima until the problem becomes trivial, and then strive to map the solution of the simplified problem back to the real problem. Applications to Lennard–Jones clusters have had limited success;[27,28] it appears that the method tends to fail for certain cluster sizes (not necessarily the most difficult ones), for reasons that are not yet fully understood.

The "basin-hopping" variant[29] of simulated annealing[30] has been applied very successfully to Lennard–Jones clusters by Wales and Doye:[25] they were able to find all global minima up to 110 atoms, with impressively small computer time requirements for the smaller clusters.[28] However, the size scaling of their method appears to be exponential;[28] hence, it comes as no surprise that applications to larger clusters are lacking.

Genetic algorithms (GA)[31–34] are an attempt to solve global optimization problems using simplified models of natural evolution. They have been used for the first time by the present author[35] and then also by other groups[20,36] for the global optimization of all degrees of freedom in atomic and molecular clusters. Already in this first article[35] the present author emphasized the importance of establishing a suitable representation of the cluster problem to the genetic algorithm. This point went unnoticed for some time, and GA optimization of clusters performed not as well as expected, despite several improvements in the algorithm.[37] A decisive step forward was made by Deaven and Ho;[38,39] they recognized that the representation problem can be avoided very simply if the typical genetic operators do not act on a string representation of the problem but directly on the clusters themselves in configuration space. Because this approach eliminates the genetic string space entirely, the label "genetic algorithm" is misleading, and the present author prefers to call this a "phenotype algorithm." Again, this was not noticed as a breakthrough for some time, with very few exceptions,[40] until this approach was adopted by several groups very recently,[41–44] and tested again for Lennard–Jones clusters by Wolf and Landman.[45] Both in the original work by Deaven and Ho[39] and in the work by Wolf and Landman[45] the phenotype algorithm finds most of the global Lennard–Jones cluster minima up to $n = 110$, but fails for some special cluster sizes were the usual dominance of the icosahedral structure type is broken by fcc ($n = 38$) or decahedral ($n = 75$–77, 102–104) geometries. The difficult shape of the potential energy surface for these special cluster sizes was investigated thoroughly for $n = 38$[17] and $n = 75$;[46] it turns out that for these cases the potential energy surface is also very much dominated by icosahedral species, and the basin of attraction of the nonicosahedral minimum is a fairly isolated region. Wolf and Landman managed to overcome this situation by seeding their method with the desired nonicosahedral species, but this amounts to using external prior knowledge about the solution and is not feasible for other systems where the solution is as yet unknown.

In this article, a new version of a phenotype algorithm is presented. Its basic ingredients are similar to those used by Deaven and Ho, but several new operations are added to eliminate the most glaring deficiencies. This algorithm finds all the minima listed in ref. 26 exactly, without exception, and without using any external knowledge. Furthermore, its scaling with cluster size is only cubic, which is a clear improvement over the exponential scaling of basin hopping[28] or the quartic-to-quintic scalings of other improved GAs.[37]

Method

The algorithm is initialized with locally optimized random cluster geometries. For the size-scaling calculations reported below, a size-independent population of 20 individuals is chosen. Tests indicate that a set of 5–15 geometries suffices, depending on cluster size ($n = 10$–150); nevertheless, a larger fixed value is taken to show the actual size scaling of the algorithm, independent of external knowledge. From this zeroeth generation, all unique pairs of geometries are formed. To each pair, a crossover operator is applied: each geometry is cut in halves by a plane through its center of mass (adjusted such that the correct total number of Lennard–Jones atoms is retained after crossover), and one of the halves is swapped with one of the halves of the other geometry in the pair. In the algorithm proposed by Deaven and Ho the cutting plane is randomly oriented. We have found that the overall algorithm performance is improved by 10–20% if the cutting plane has a definite position in 50% of the cases, such that it separates the "best" half from the "worst" half of the geometry (a similar operation was already used in ref. 43). To determine the quality of cluster parts, each atom is assigned an individual contribution to the total potential energy, which is simply half of the sum of the pair potential terms for this atom. Then the worst half of one geometry replaces the best half of the other; from the resulting children geometries, one has usually a lower energy than both parents, while the other one has a higher energy. With random orientation of the cutting plane, the energies of the children are usually in the range of the energies of the parents. Therefore, swapping of random halves has more of an explorative character, while swapping of optimized halves puts more emphasis on improving existing geometries. Hence, it is not surprising that both should be used to keep the balance between local optimization and exploration.

With a probability of 15%, each resulting geometry is randomly mutated. The mutation operator moves a randomly determined number of atoms (between 1 atom and 40% of the number of atoms in the cluster) away from their original positions, by a distance that is randomly determined to be between the nearest neighboring atom and the approximate radius of the cluster. Each geometry is then locally optimized. For local optimizations, the limited-memory quasi-Newton method of Liu and Nocedal[47] is used. Its memory requirements and iteration time scale linearly with the dimension of the search space. Additionally, it turns out to be more robust towards numerical difficulties originating from accidentally close atoms, compared to standard implementations of conjugate gradients.

To the resulting enlarged set of new geometries, all parent geometries are added to ensure that good solutions already found do not get lost again (a strong form of elitism[32]). From this intermediate set of geometries, the ones with the lowest energies are chosen for the actual next generation. To maintain diversity, however, the geometries have to maintain a minimum energy difference to each other. For the present calculations, this minimum difference is set to 0.1 (in the usual reduced units, with the minimum of the Lennard–Jones pair potential at -1.0). In addition, a minimum degree of exploration is ensured by enforcing 20–30% of the selected geometries to be mutants. The resulting next generation is then again subjected to the reproductive cycle described above.

As already noted by several other researchers, the algorithm described so far will converge fairly quickly to geometries that differ from the global minimum geometry (or a low-lying local minimum geometry) only by a few misplaced atoms and the corresponding vacancies elsewhere in the cluster. Repair of these final faults takes many generations with the operators described so far. This final repair process can be speeded up greatly by a "directed mutation" step: this new operator is invoked whenever there is no change in energy in any one member of the population from one generation to the next. Using again the distribution of the total potential energy into individual atomic contributions as described above, the "worst" atom and the "best" vacancy in the cluster are located, and the worst atom is simply moved to the best vacancy position. In particular for larger clusters ($n > 100$), the resulting overall speedup can be so large that it makes all the difference between an efficient solution and impractically long computation times. This directed mutation operator bears some faint resemblance to the "add-and-etch" processes used by Wolf and Landman,[45] but it is better tailored to the actual needs of the optimization process and presumably also less expensive (because it does not involve several local optimizations).

In this form, the algorithm reliably finds all published global minima[26] up to and including $n = 150$, including the difficult cases $n = 38$ and $n = 102$–104. For $n = 75$–75, however, it takes about 20–40 times longer than for the neighboring simple icosahedral cases $n = 74$ and $n = 78$. This can be dramatically improved by borrowing the concept of niches from the GA literature.[32] To distribute different types of clusters into niches, a measure is needed to differentiate between clusters of varying geometric characteristics. A first, simple approach to such a measure is the observation that two-dimensional projections of particularly oriented icosahedral, decahedral, and fcc clusters not only show characteristic geometric patterns (cf. Fig. 1), but also distinctly different degrees of coverage of the projection plane: most strikingly, in fcc clusters, an orientation can be found where the atoms are "lined up" in only very few positions on the projection plane, producing a very sparse pattern. Hence, a simple approximation g to the amount of projection plane coverage is calculated for each geometry in the intermediate set, as follows: the cluster is rotated about its center of mass into an orientation where a maximum number of atoms conincides in any one point of the projection of the atomic coordinates into the xy-plane. A grid with 40 points in each direction is put down in the xy-plane, covering an area corresponding to the maximum cluster diameter. The number g of grid squares containing at least one projected atom is counted. Figure 1 shows the value of g for typical cases, namely the lowest local icosahedral minima of $n = 38$ and $n = 104$, and the corresponding fcc and decahedral global minima, respectively (note that g depends by definition on cluster size; hence, only g values for the same value of n are comparable). Clearly, the g-values for these geometries differ significantly; tests show that they also differ to a similar extent for local minima higher in energy that bear some resemblance to the pure icosahedral, decahedral, or fcc types. Therefore, this crude geometry measure g can be used to establish niches. In the selection step that extracts the next generation geometries from the intermediate set, only a fixed number m of geometries are accepted that have g-values closer than Δg to a geometry already selected (in addition to the energy difference and mutation criteria explained above). This allows fcc and decahedral geometries to survive even in the presence of many icosahedral geometries of lower energy; the latter cannot take over the whole population. For populations of 20 geometries, it is found that setting $m = 10$ and $\Delta g = 20$ lowers the computer time needed for $n = 75$–77 to about 5–10 times that of $n = 74$ or $n = 78$. If one puts the icosahedral geometries to a further disadvantage by not allowing any geometry with $g > 36$ to be selected into the next generation (with the exception of mutants), solution times for $n = 75$–77 drop down still further, to about one to five times that of $n = 74$ or $n = 78$. Interestingly, restricting also the mutants to $g \leq 36$ is no improvement anymore; apparently, this restricts exploration and diversity too much.

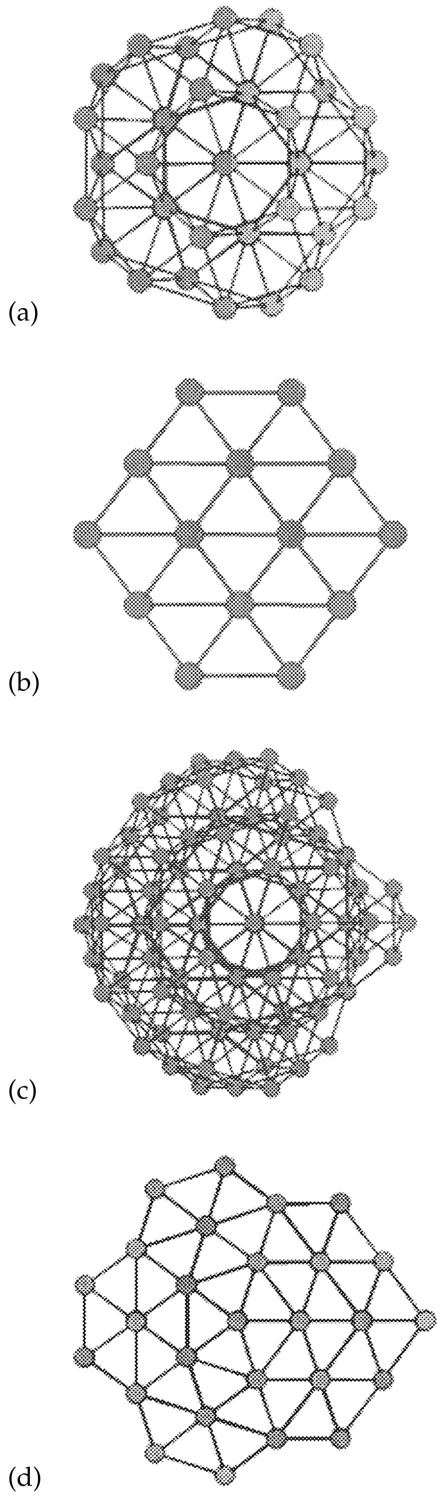

FIGURE 1. Two-dimensional projections of the lowest local icosahedral minimum geometries of $n = 38$ (a) and $n = 104$ (c), of the global fcc minimum geometry of $n = 38$ (b), and of the global decahedral minimum geometry of $n = 104$ (d); the corresponding values of the geometrical measure g (see text) are: (a) $g = 29$, (b) $g = 14$, (c) $g = 75$, (d) $g = 27$.

It is important to note that no external knowledge about the true solutions is used in this niching concept. Only m and Δg are input; the actual g-value ranges of the niches are established by the geometries that happen to be accepted first during the selection step. In fact, the actual niches are free to vary from one generation to the next, with only m and Δg kept fixed. The additional restriction of $g \leq 36$ also does not need to be imposed from outside; it is perfectly possible to let the algorithm converge towards the dominating icosahedral geometries first and then to check for other geometry types by barring the so far most successful g-range from further proliferation; ideally, this would require only twice the computer time of a normal run.

Also, the niches exist only temporarily, in the selection step; in contrast to some stricter definitions of niches,[32] there is constant unrestricted interbreeding between all niches. Doll et al.[41] have used a structural diversity algorithm that is similar in this respect to the niching used here. The present method, however, is more flexible (it allows not just one geometry of each type, but groups of geometries of controlled size and similarity) and computationally cheaper: Doll et al. need $n(n-1)(n-2)$ superposition attempts and a complete set of atomic distance determinations for each attempt, and they need to do this for each pair of clusters anew—whereas, the determination of g only needs an (almost) n-independent number of rotations to determine the maximum-coincidence orientation and a simple scan over a grid, with the grid size scaling only like the maximum area of the cluster projection, and this has to be done only once per cluster.

One may argue that g may happen to provide a useful niche definition only for Lennard–Jones clusters, and that this amounts to the use of external knowledge. This seems hard to refute, but actually the first half of this argument is dubious, and the second half is wrong or at least misleading. Even for Lennard–Jones clusters, g can presumably be replaced by other measures: As is obvious from Figure 1, one could equally well test for the presence of (approximate, local) fivefold symmetry: (near-)icosahedral geometries will always show several axes of this type, decahedral geometries only one, and fcc geometries none. Clearly, some measures are more general than others; but this is not really important. The central issue is that the usefulness of g (or any other measure) can already be deduced from high-lying local minimum geometries; one does not need to know the true global or low-lying local minimum structures in advance. Therefore, ideally, g (or some other measure) should be extracted from early stages of the algorithm and then used lateron. In this sense, this is not "external" knowledge but extraction of information from the problem to be solved. In fact, striving for a totally problem-independent algorithm is nonsense; every reader who is not convinced of this is urged to study the literature on the "no free lunch" theorem,[48] in particular, Culberson's brilliant article "On the futiliy of blind search."[49]

Benchmark Application

The algorithm described in the last section finds all the geometries listed for $n = 2$–150 in ref. 26, without exception, and with energies agreeing to all nine decimal places given there. Note that this is a pure feasibility test, that is, the algorithm was stopped as soon as the given final energy for each cluster size was reached; no attempt was made to find possible lower lying minima.

The computer times needed (on a single 400-MHz Pentium-II processor, using the Portland Group[50] Fortran compiler PGF77, release 3.0, with "-O" optimization, under linux) are displayed vs. cluster size n in Figure 2. Because a fair amount of random steps is involved in each optimization run, there is considerable variation in the lengths of different runs for the same cluster size. Instead of taking averages over a statistically significant number of runs for selected cluster sizes, the lengths of single runs for each cluster size between $n = 10$ and $n = 150$ are displayed. To reduce the scatter of the

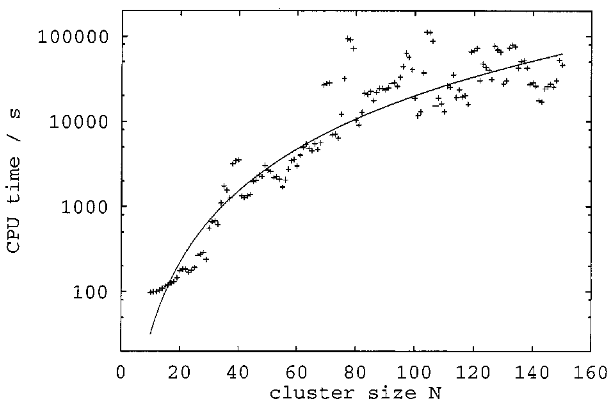

FIGURE 2. CPU times needed to reach the known global minimum geometry for each Lennard–Jones cluster size between $n = 10$ and $n = 150$, and fit of an approximately cubic polynomial to these data (note the logarithmic scale; see text for further explanations).

data, an ad hoc averaging over three neighboring cluster sizes is performed, purely as a guide to the eye. In addition, a least-squares fit of the functional form $f(n) = a \times n^b$ to the data has been performed, resulting in $a = 0.05 \pm 0.02$ and $b = 2.8 \pm 0.1$ (the dependence of this result on the degree of data averaging is insignificant); the curve $0.05 \times n^{2.8}$ is also displayed in Figure 2. Of course, the present data could also be fitt with an exponential form; this would result in a straight line in Figure 2, in obvious contrast to the actual trend of the data.

Note that the phenotype program used is a development version; no attempt was made at performance optimization. Also, all parameters were held fixed at their values given in the last section; no attempt was made at further parameter optimization, in particular, the parameters do not change with cluster size. Therefore, there is a considerable computational overkill for the smaller cluster sizes; in fact, the times reported for an application of "basin-hopping" to cluster sizes $n \leq 50$ in ref. 28 are one to two orders of magnitude smaller—but those times were taken on a different processor, with a highly optimized program, and with parameters that were meticulously optimized for each cluster size.

To put the given CPU times in perspective, other timing details should be mentioned. One single local geometry optimization takes typically 0.1 CPU seconds for the smallest clusters ($n = 20$–30), 1.5–2 CPU seconds for medium sizes ($n = 70$–80), and 2–8 CPU seconds for the largest ones ($n = 140$–150). The number of local optimizations per generation is almost fixed, because each member of the intermediate population is optimized once, leading to $20(20 - 1) = 380$ local optimizations per generation (plus about 0–20 more, depending on the necessary amount of postimprovements, as described in the last section). The number of generations rises from 1 for the smallest clusters via approximately 2–7 for the medium ones to about 7–15 for the largest clusters, with large variations due to the smallness of these numbers and the random nature of the algorithm. Therefore, none of these numbers gives a clearer picture of algorithm performance than the overall timings. The amount of time spent in different parts of the algorithm also varies with cluster size (and randomly): For the smallest clusters, the generation of the intermediate population via crossover and local optimizations takes approximately 15% of the total time (or, equivalently, the time per generation); the geometry analysis to calculate g-values and establish niches takes 80% (documenting some of the above-mentioned overkill for these small sizes); and the final improvement steps (including directed mutation) take 5%. For medium sizes, the corresponding partitioning of time varies from (61%, 28%, 11%) during the first generations to (30%, 12%, 58%) during the final generations (with necessarily more emphasis on final improvements). For the largest sizes, the time partitioning varies from (70%, 14%, 16%) to (64%, 12%, 24%).

Actually, the point to be made here is neither raw algorithm performance nor a detailed timing analysis of the algorithm, but overall size scaling. A variety of approaches can solve small Lennard–Jones clusters within practically reasonable times (see, e.g., ref. 37); nevertheless, it appears likely that the highly optimized basin hopping results for small clusters are very hard to beat. But, as already mentioned in the Introduction, from the data given in ref. 28, the basin hopping algorithm appears to scale exponentially with cluster size. This rules out any applications of this algorithm to larger clusters. In ref. 37, a size scaling of approximately $n^{4.5}$ was reported for a standard-GA enhanced with local optimizations and an additional minimum-gradient criterion; this was still considered prohibitive, and clusters larger than $n = 31$ were not even tried. Therefore, the present scaling of approximately n^3 is a significant improvement. Extrapolating the fitted curve in Figure 2, cluster sizes up to $n = 250$ appear to be within reach; and optimization of the program and the parameters will surely push this limit even further up.

Conclusions

A new version of a phenotype algorithm was presented, including several new features that enhance performance, most notably a directed mutation operator that greatly reduces the problem of repairing isolated faults in the final phase of the algorithm, and dynamically defined niches that prevent a single geometry type from dominating the whole population, thus enabling the algorithm to locate the notoriously difficult nonicosahedral minima efficiently and reliably. Additionally, already the development version of this program scales only cubically with cluster size, which is a marked improvement over other scalings published in the literature.

Obviously, a thorough optimization of the program and the parameters are necessary before the performance data for smaller clusters can compete with that of the basin-hopping algorithm. Most importantly, the amount of local optimizations used should be reduced, particularly for cases where the potential is not analytically differentiable (as in more elaborate empirical potentials involving iterative loops[15]). Ultimately, thanks to the flexibility of the genetic algorithm approach, it is even possible to include the whole basin hopping algorithm as an additional mutation operator, arriving at a method that combines the efficiency of basin hopping for small clusters with the favorable size scaling of the phenotype scheme for large clusters.

The program described in this article has already been adapted to molecular clusters and tested with pure water clusters, using the approach combining more elaborate empirical potentials with ab initio calculations as described in ref. 15, and with Mg^{2+}–water clusters.[51] Remarkably, it turns out that standard-GA approaches, as used in ref. 15, already fail to locate the global cubic minimum of $(H_2O)_8$; the algorithm presented here finds this geometry reliably within only 1 generation.

Future research will apply the method presented here to larger Lennard–Jones clusters, in an attempt to get closer to the size region $n = 800$–1500 where a transition from the dominance of the icosahedral motif to fcc structures is conjectured to occur.[52–55] To actually treat clusters on the order of 1000 atoms, it will presumably be necessary to introduce further algorithmic modifications to reduce the size scaling; an interesting candidate is cluster partitioning, as introduced in Michaelian's "symbiotic GA."[56]

Furthermore, it seems likely that the method development knowledge gathered here can be transfered at least to some extent to the related areas of molecular docking[57, 58] and protein folding,[59–63] where GA-style applications still seem to be closer to the old standard-GA paradigm, leaving much room for method development.

References

1. Moskovits, M. Annu Rev Phys Chem 1991, 42, 465.
2. Berry, R. S. J Phys Chem 1994, 28, 6910.
3. Martin, T. P. Phys Rep 1996, 273, 199.
4. Swihart, M. T.; Girshick, S. L. J Phys Chem B 1994, 103, 64.
5. Yu, F. Q.; Turco, R. P.; Karcher, B. J Geophys Res Atmos 1999, 104, 4079.
6. Clement, C. F.; Ford, I. J. Atmos Environ 1999, 33, 489.
7. Bandy, A. R.; Ianni, J. C. J Phys Chem A 1998, 102, 6533.
8. Siegel, R. W. Nanostruct Mater 1993, 3, 1.
9. Datta, S.; Janes, D. B.; Andres, R. P.; Kubiak, C. P.; Reifenberger, R. G. Semicond Sci Technol 1998, 13, 1347.
10. Tenne, R.; Homyonfer, M.; Feldman, Y. Chem Mater 1998, 10, 3225.
11. Subramoney, S. Adv Mater 1998, 10, 1157.
12. Liu, K.; Brown, M. G.; Saykally, R. J. J Phys Chem 1997, 101, 8995.
13. Liu, K.; Brown, M. G.; Carter, C.; Saykally, R. J.; Gregory, J. K.; Clary, D. C. Nature 1996, 381, 501; Gregory, J. K.; Clary, D. C. J Phys Chem 1996, 100, 18014.
14. Kim, J.; Kim, K. S. J Chem Phys 1998, 109, 5886.
15. Hartke, B.; Schütz, M.; Werner, H.-J. Chem Phys 1998, 239, 561.
16. Wales, D. J.; Miller, M. A.; Walsh, T. R. Nature 1998, 394, 758.
17. Doye, J. P. K.; Miller, M. A.; Wales, D. J. J Chem Phys 1999, 110, 6896.
18. Ville, L. T.; Vennik, J. J Phys A: Math Gen 1985, 18, L419.
19. Greenwood, G. W. Z Phys Chem 1999, 211, 105.
20. Ville, L. T. Annu Rev Comp Phys 1999, submitted.
21. Hoare, M. R.; Pal, P. Adv Phys 1971, 20, 161; Hoare, M. R. Adv Chem Phys 1979, 40, 49.
22. Hoare, M. R.; McInnes, J. Adv Phys 1983, 32, 791.
23. Northby, J. A. J Chem Phys 1987, 87, 6166.
24. Xue, G. L. J Glob Optim 1994, 4, 425.
25. Wales, D. J.; Doye, J. P. K. J Phys Chem A 1997, 101, 5111.
26. Cambridge Cluster Database: http://brian.ch.cam.ac.uk/CCD.html.
27. Pillardy, J.; Piela, L. J Phys Chem 1995, 99, 11805; J Comput Chem 1997, 18, 2040.
28. Wales, D. J.; Scheraga, H. A. Science 1999, submitted.
29. Li, Z.; Scheraga, H. A. Proc Natl Acad Sci USA 1987, 84, 6611.
30. Kirkpatrick, S.; Gellat, C. D., Jr.; Vecchi, M. P. Science 1983, 220, 671.
31. Holland, J. H. Adaption in Natural and Artificial Systems; University of Michigan Press: Ann Arbor, 1975.
32. Goldberg, D. E. Genetic Algorithms in Search, Optimization, and Machine Learning; Addison–Wesley: Reading, 1989.
33. Mitchell, M. An Introduction to Genetic Algorithms; MIT Press: Boston, 1996.
34. Judson, R. S. Rev Comput Chem 1997, 10, 1.
35. Hartke, B. J Phys Chem 1993, 97, 9973.
36. Xiao, Y.; Williams, D. E. Chem Phys Lett 1993, 215, 17.
37. Gregurick, S. K.; Alexander, M. H.; Hartke, B. J Chem Phys 1996, 104, 2684.
38. Deaven, D. M.; Ho, K. M. Phys Rev Lett 1995, 75, 288.
39. Deaven, D. M.; Tit, N.; Morris, J. R.; Ho, K. M. Chem Phys Lett 1996, 256, 195.
40. Pullan, W. J. Comput Phys Commun 1997, 107, 137.
41. Curotto, E.; Matro, A.; Freeman, D. L.; Doll, J. D. J Chem Phys 1998, 108, 729.
42. Zacharias, C. R.; Lemes, M. R.; DalPino, A., Jr. J Mol Struct (Theochem) 1998, 430, 29.
43. Hobday, S.; Smith, R. J Chem Soc Faraday Trans 1997, 93, 3919.
44. Ho, K.-M.; Shvartsburg, A. A.; Pan, B.; Lu, Z.-Y.; Wang, C.-Z.; Wacker, J. G.; Fye, J. L.; Jarrold, M. F. Nature 1998, 392, 582.
45. Wolf, M. D.; Landman, U. J Phys Chem A 1998, 102, 6129.
46. Doye, J. P. K.; Miller, M. A.; Wales, D. J. Preprint, cond-mat/9903305, March 19, 1999.
47. Liu, D. C.; Nocedal, J. Math Program 1989, 45, 503.

48. Wolpert, D. H.; Macready, W. G. IEEE Trans Evol Comput 1997, 1, 67.
49. Culberson, J. C. Evol Comput J 1998, 6, 109.
50. PGF77 is a trademark of the Portland Group, Inc.; http://www.pgroup.com.
51. Schumann, U. Diploma Thesis, University of Stuttgart (1999).
52. van de Waal, B. W. J Chem Phys 1989, 90, 3407; 1993, 98, 4909.
53. van de Waal, B. W. Phys Rev Lett 1996, 76, 1083.
54. Farges, J.; de Feraudy, M. F.; Raoult, B.; Torchet, G. J Chem Phys 1983, 78, 5067; 1986, 84, 3491.
55. Swope, W. C.; Andersen, H. C. Phys Rev B 1990, 41, 7042.
56. Michaelian, K. Chem Phys Lett 1998, 293, 202.
57. Vieth, M.; Hirst, J. D.; Dominy, B. N.; Daigler, H.; Brooks, C. L., III J Comp Chem 1998, 19, 1623.
58. Morris, G. M.; Goodsell, D. S.; Halliday, R. S.; Huey, R.; Hart, W. E.; Belew, R. K.; Olson, A. J. J Comp Chem 1998, 19, 1639.
59. Neumaier, A. SIAM Rev 1997, 39, 407.
60. Onuchic, J. N.; Luthey–Schulten, Z.; Wolynes, P. G. Annu Rev Phys Chem 1997, 48, 545.
61. Durup, J. J Mol Struct (Theochem) 1998, 424, 157.
62. Dobson, C. M.; Šali, A.; Karplus, M. Angew Chem 1998, 110, 908.
63. Hansmann, U. H. E.; Okamoto, Y. In Annual Review of Computational Physics VI, Stauffer, D., Ed.; World Scientific: Singapore, 1999.