Application of artificial intelligence to search ground-state geometry of clusters

Maurício Ruv Lemes, L. R. Marim, and A. Dal Pino, Jr.
Department of Physics, Instituto Tecnológico de Aeronáutica, Pca. Marechal Eduardo Gomes, 50 São José dos Campos, Sao Paulo, Brazil 12228-900

(Received 26 December 2001; published 19 August 2002)

We introduce a global optimization procedure, the neural-assisted genetic algorithm (NAGA). It combines the power of an artificial neural network (ANN) with the versatility of the genetic algorithm. This method is suitable to solve optimization problems that depend on some kind of heuristics to limit the search space. If a reasonable amount of data is available, the ANN can ''understand'' the problem and provide the genetic algorithm with a selected population of elements that will speed up the search for the optimum solution. We tested the method in a search for the ground-state geometry of silicon clusters. We trained the ANN with information about the geometry and energetics of small silicon clusters. Next, the ANN learned how to restrict the configurational space for larger silicon clusters. For Si_{10} and Si_{20}, we noticed that the NAGA is at least three times faster than the ''pure'' genetic algorithm. As the size of the cluster increases, it is expected that the gain in terms of time will increase as well.

DOI: 10.1103/PhysRevA.66.023203 PACS number(s): 36.40.−c, 31.15.Ct, 02.60.Pn, 31.15.Pf

I. INTRODUCTION

Many areas of science are suitable for the application of artificial intelligence (AI) methods. AI emulates human thought and reasoning. A good problem for AI is one which is so complex that it cannot be solved using conventional algorithms. Problems that conventional computational methods fail to solve fall into two categories: they may be computationally intractable or conceptually obscure. The obscure problems are those that are poorly understood and, so far, neither conventional nor AI methods have had much of an impact on them. On the other hand, computationally intractable problems are quite different. These are exactly the ones that AI methods thrive on.

A familiar chemical example is the growth in the number of isomers in a homologous series. For instance, there are more than 4×10^9 distinct isomers of $C_{30}H_{62}$. It is clearly intractable to calculate every single molecular geometry to find the geometry of the ground state. To solve this kind of problem the investigator uses shortcuts, hunches, and rules of thumb. Experience gives some guidance about how to solve a particular problem. It is this kind of specialized knowledge that makes an expert so valuable. This is exactly the approach that has been used in the search for the ground-state geometry of atomic clusters.

Structural characterization of silicon clusters is currently one of the most active frontiers in physics and chemistry [1,2]. Theoretical predictions for Si_n ($n>7$) are contradictory [2–9] which makes the problem even more interesting to investigate. A recent and comprehensive study of the stable structures for Si_{20} clusters can be found in Ref. [10]. Specifically, Kaxiras and Jackson [5,6] limited their search space by using the well-known terminations of silicon surfaces to guide them through the maze of candidates for the ground-state geometry of Si_{20}. Grossman and Mitas [7,8] developed their own heuristics. They chose to describe silicon clusters (Si_n, $n<40$) as a stack of triangular layers of atoms. The most important issue discussed in this article may be put in a rather provocative way: Can an automatic system, based on AI methods, replace human experience in this field? We believe that our results indicate that the answer is yes.

The search for the ground-state configuration of a collection of atoms belongs to a class named NP-hard [11–13] optimization problems. This means that the computer time necessary to find the exact solution will increase exponentially with cluster size. Methods engineered to solve this type of problem require some kind of initial candidate solution, i.e., a starting point. From this initial guess the optimizer generates other candidate solutions until some objective criterion is satisfied. The choice of the starting point is of paramount importance to the success of the optimization [14]. If an inconvenient starting point is used the algorithm may take a lot of time to find the global minimum or, even worse, may get stuck in a local minimum. Therefore, an efficient method of obtaining these starting points is highly desirable. Usually the expertise of the investigator may help to find convenient starting points. In this paper, we are going to show that a technique based on AI (the artificial neural network) can be useful to yield an optimization procedure (genetic algorithm) with good starting points.

All total-energy calculations presented in this article were performed in the framework of the tight-binding (TB) model described in Ref. [15]. Consequently, our calculations are limited by the accuracy of the TB method. Recently, Li and Cao [10], using full-potential linear-muffin-tin-orbital molecular dynamics, found a candidate for the ground-state geometry of Si_{20} that differs from the TB prediction by less than 0.05 eV per atom. Nevertheless, the approach that we are introducing could, equally well, be combined with any other, more powerful, total-energy method.

The sequence of this paper is arranged as follows. The next section briefly introduces artificial neural networks; Sec. III deals with the mapping problem, i.e., it shows how a molecular geometry can be cast into a form that can be fed to an artificial neural network (ANN); Sec. IV shows the actual ANN architectures used in this article. Sections V and VI present our results and conclusions, respectively.

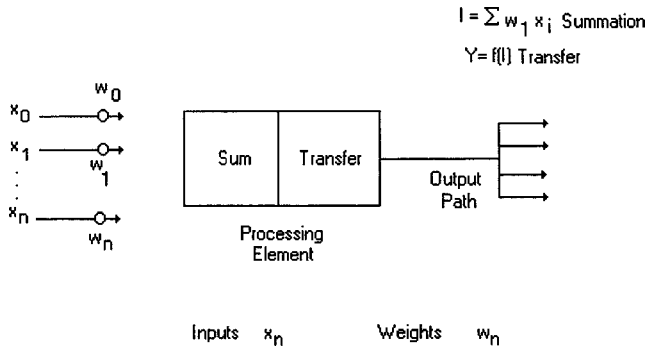

FIG. 1. A simple artificial neuron representation.

II. ARTIFICIAL NEURAL NETWORKS

A crucial feature of the behavior of the brain is that it need not to be taught how to learn. This is a characteristic that one hopes to include in any AI method. In particular, artificial neural networks are computer programs based on a simplified model of the brain. They do not attempt to copy the fine detail of how the brain works, but try to reproduce its logical operation using a collection of neuronlike entities to perform processing.

Any ANN starts from a position of ignorance, so a training period is required before it can tackle real problems. During training, the ANN is presented with examples of what it may learn to interpret; these examples constitute the training set. The training set is divided into two parts: the training stimuli, i.e., a collection of inputs to the ANN, and the associated training target, which is the desired output (the right answer) for each stimulus.

Some of the strong points of ANN's are as follows. (i) Neural networks are fault tolerant, i.e., it is able to cope with ''fuzzy data'' without modification. (ii) Once trained, an ANN can deal with previously unseen data. (iii) ANN's have the capability for parallel operation built in. (iv) ANN's operate by discovering new relationships among their input data.

ANN's excel in applications that (i) present incomplete or unreliable data, (ii) have numerous training examples available, (iii) have unknown rules that lead from a given input to a particular output, and (iv) in which the explanation of how any decision is reached is not required.

Figure 1 shows a fundamental representation of an artificial neuron. It shows various inputs x_n to it, each of which is multiplied by a connection weight represented by w_n. These products are summed and fed to a transfer function that generates a result (output). A typical neural network is obtained by grouping these neurons into layers. The connections between these layers, the summation and transfer functions, comprise a functioning ANN. Most applications require ANN's that contain three types of layers: input, hidden, and output (see Fig. 2). The input layer receives data from an input file. The output layer sends information to the researcher and, between these layers, there can be many hidden layers. In most ANN's each neuron in a hidden layer receives signals from all neurons in a layer above it. After a neuron performs its task it passes its output to all neurons in the layer below it. Such ANN's are called feedforward neural networks. The number of neurons in each layer and the transfer functions characterize the architecture of the ANN.

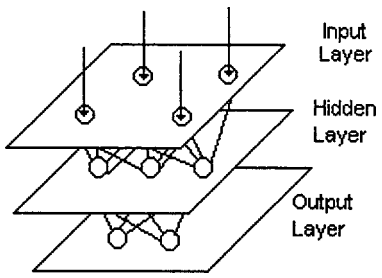

FIG. 2. A simple neural network diagram.

III. THE MAPPING PROBLEM

It must be possible to cast the solutions to a problem in a form that an ANN can manipulate. To achieve a formulation that could be used in the framework of ANN's, we have described the geometry of a cluster as a stack of planar collections of atoms.

The geometry of an atomic cluster is usually expressed in terms of Cartesian coordinates or internal coordinates (distances, bond angles, and dihedral angles). Unfortunately, these sets of variables are not suitable to train an ANN efficiently. Thus, it is necessary to map the coordinates that define the geometry of any cluster into a set of inputs appropriate to ANN computation.

Here, we have used the same kind of approach that has been used for many years to describe crystals and polytypes, i.e., we have described the spatial geometry of each cluster as a stack of planar arrangements of atoms. We have chosen, arbitrarily, the set of planar elements presented in Fig. 3. In other words, this is a basis set and any ''cluster'' is a particular composition of these elements.

Next, we must generate the training set, a convenient combination of stimuli and targets. This means that the ANN is presented with a set of clusters (stimuli), described according to our basis set, whose total energy per atom (target) is known. Here, we have used the results of total-energy optimization of small silicon clusters (Si_n, $n \leqslant 9$) previously calculated. We have used 110 different cluster descriptions to-

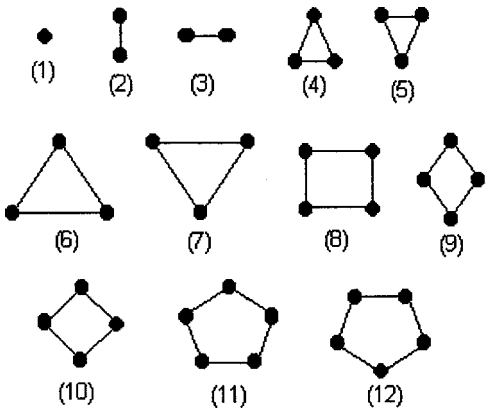

FIG. 3. Basis set of planar structures. These elements may be stacked to generate any cluster.

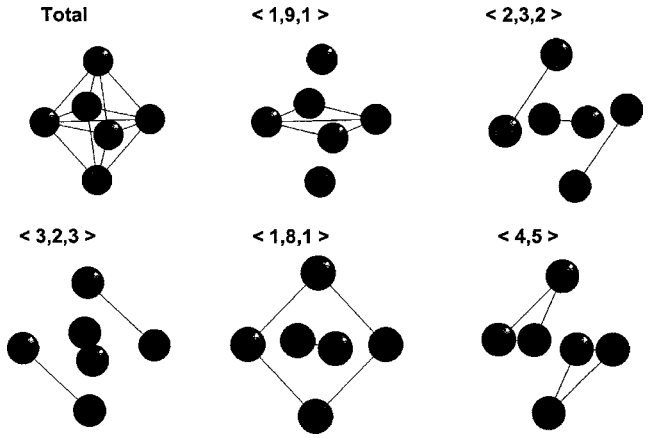

FIG. 4. Sample structures for Si_6.

gether with their calculated energy per atom. It is worth mentioning that the training set must have a mixture of good and bad geometries, i.e., local, global, and other structures whose energy per atom makes them very unlikely to occur. Figure 4 shows an example of a few clusters, their description in terms of our basis set, and their calculated energy per atom.

We have already stated that ANN's perform their best when trained with numerous sample cases. However, we have only a limited amount of data to feed the ANN. Thus, one should not expect that the ANN, trained with a small amount of information, would correctly predict the total energy of a polyatomic system. Nevertheless, the ANN does a fine job in selecting a convenient starting point for an optimization procedure.

IV. ANN ARCHITECTURE AND TRAINING PROCEDURE

We used a standard feedforward ANN consisting of an input layer, one intermediate layer of neurons, and an output layer. The input layer has 11 neurons and the input data is the sequence of elements of the basis set necessary to describe the clusters geometry. Three ANN's (11-p-2, with p equal to 3 or 6 or 9) were tested; they differ only by the number of neurons, p, in the hidden layer. Two output neurons were used for each ANN. They mainly separate the clusters into two classes: reasonable candidates for solutions and eliminated ones.

The weights that connect neurons from different layers are determined during the training procedure. Here, we used 110 different geometries of silicon clusters (Si_n, $n<9$) whose descriptions in terms of our basis set and their respective energy per atom were previously calculated. Clusters whose TB binding energy is larger than 3.0 eV per atom are classified as candidate solutions whereas those with lower values of binding energy are supposed to be eliminated. Each ANN was trained by a standard backpropagation [16] method.

The training procedure can be performed very quickly because the ANN's are only supposed to select potentially good candidates. Anyway, too much training may deteriorate the net's capability to make reasonable predictions, a feature known as overfitting. Once the training procedure is completed, the weights are found and the net is ready to make predictions. A totally new cluster geometry may be presented to the ANN and it will be classified as either a good or bad candidate solution. Since we have trained the ANN's with information related to clusters with nine or fewer Si atoms, the ANN's are in a position to make predictions for larger clusters (Si_n, $n>9$). As we will discuss in the next section, we have applied this method to Si_{10} and Si_{20}.

V. APPLICATION TO SILICON CLUSTERS

As we previously stated, any global optimization requires some kind of starting point (or points) to initiate its search procedure. The genetic algorithm (GA) [17–20] is an attempt to solve global optimization problems using simplified models of natural evolution. An initial population of candidate solutions must be introduced to the GA. Next, the fitness of each element of the population is calculated. Fitness determines the probability of the element producing offspring. Thus, less fit elements have a low probability of producing children while the opposite occurs with the fittest elements. Occasionally, a mutation takes place, i.e., a random modification of one element may occur. After a number of generations it is expected that the population will evolve to produce the desired optimum solution.

Deaven and Ho [21,22] introduced an efficient way to use the GA to search for the ground-state geometry of clusters. Some of us [23] have already made use of the GA with considerable success. Thus, it was quite reasonable to couple our ANN's with the GA.

Our procedure to search for the ground-state geometry of Si_n consists of the following steps. (i) Select n, the size of the cluster. (ii) Generate all possible structures (geometries) according to our stacking approach. (iii) Present each cluster geometry to the ANN. It will classify the geometry as a possible candidate or it will eliminate the geometry. (iv) Generate Cartesian (or internal) coordinates for the candidates. (v) Randomly select N_p candidate solutions to act as the initial population. (vi) Use the other candidate solutions as mutations according to some prescribed rule. (vii) Run a GA for a number of generations. We have named our procedure the neural-assisted genetic algorithm (NAGA).

To test the NAGA's efficiency we have compared it to the ''pure'' GA. Initially, we performed the search for Si_{10}. There are several thousand different stacking sequences of the elements of our basis set that generate a possible geometry for Si_{10}. From all possibilities, each ANN selects about 600 as possible candidates. A number $N_p=10$ of them become the initial population of the GA; the rest are saved to be used as mutations. We have tested three different architectures: ANN3, ANN6, and ANN9 (the numbers 3, 6, and 9 correspond to the number of neurons in the hidden layer). We will only show results from ANN6 because results for the other two ANN's are very similar.

Figure 5 shows the evolution of the opposite of the binding energy per atom for Si_{10} according to the number of generations. As we have stated before some operators in the GA make use of random numbers. Thus, in order to make

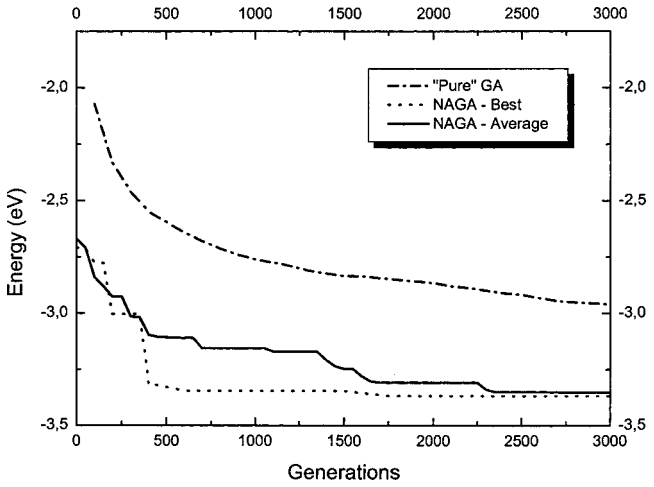

FIG. 5. Evolution of opposite of the binding energy per atom of Si_{10} as a function of the number of generations.

our results more reliable, each line corresponds to the average of ten different calculations. The best result obtained by the NAGA is also presented. It is clear that there is a significant improvement in search speed. The fact that the initial energy for the NAGA is much smaller than the corresponding initial energy for the pure GA is already an indication of the success of the ANN's. Another good sign is the fact that it takes only 300 generations for the NAGA result to reach the -3.0 eV level while a pure GA needs 4000 generations to get to the same level.

Next, we made another test. We restricted the population to those elements simultaneously selected by all three ANN's (selected NAGA) and we compared them to those that were rejected by them. The evolution of the opposite of the binding energy per atom is shown in Fig. 6. We have noticed that a GA run that employed only rejected elements (rejected NAGA) is even slower than ''pure'' GA. This proves the ability of the ANN to identify good candidates. It has learned

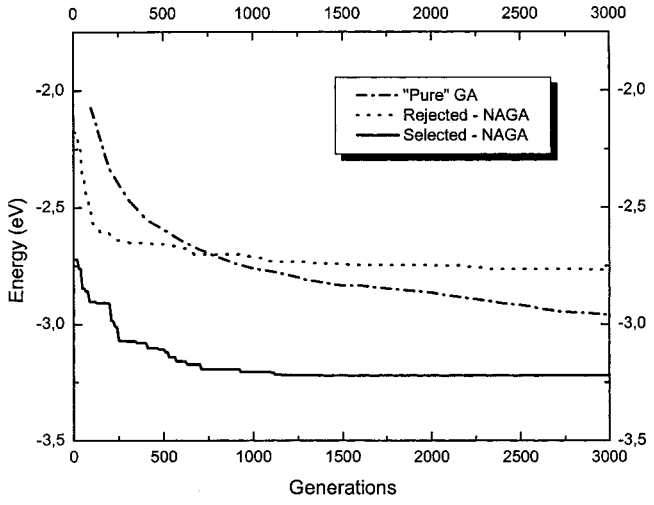

FIG. 6. Comparison of the evolution of opposite of the binding energy per atom for ''pure'' GA (dashed line), rejected NAGA (dotted line), and selected NAGA (solid line), as function of the number of generations.

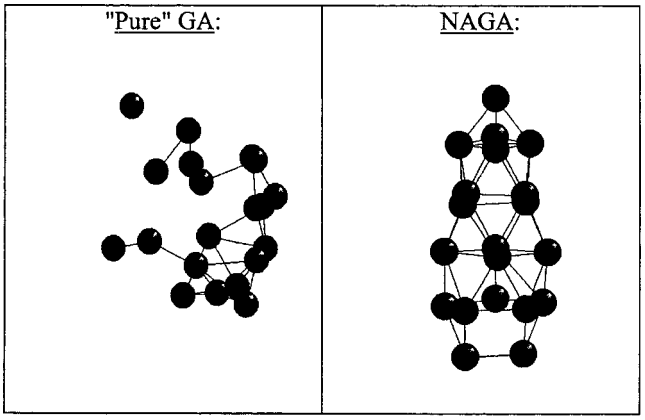

FIG. 7. Structures for Si_{20}.

from the data extracted from small silicon clusters how a larger cluster with a binding energy per atom sufficiently high is formed. In some sense, one may say that the ANN understands the energetics of the clusters. Basically, this is the same procedure an investigator follows, prior to using knowledge to restrict the search space.

Slow convergence becomes an issue after a few hundred generations. This is a usual feature of the GA and it is a consequence of poor genetic variety. Such lack of diversity is a crucial issue that deteriorates the performance of the GA. Hartke [11] has shown how to deal with this feature.

The geometry of the clusters obtained by the NAGA after only 1000 generations is very similar to the ground-state geometry. After 3000 generations the ''pure'' GA remains unable to reach candidates whose binding energy per atom is close to the 3.0 eV level.

Next, we have used the same ANN's to restrict the search space in the case of Si_{20}. This is a very unfavorable condition, because only information related to Si_n ($n \leqslant 9$) is included in the ANN's data basis. Clearly, inclusion of more information from Si_n ($10 \leqslant n \leqslant 19$) would make the NAGA's performance much better.

Figure 7 shows the ''best'' clusters obtained by the ''pure'' GA and NAGA after ten runs of 100 and 40 generations, respectively. After 100 generations, the pure GA was not able to produce a stable cluster. On the other hand, the NAGA produced a structure that already presents most features that the ground-state geometry of Si_{20} is expected to have. It resembles structures proposed by Li and Cao [10] as the most stable geometries for Si_{20}.

VI. CONCLUSIONS

We have introduced a global optimization procedure, the neural-assisted genetic algorithm. It combines an artificial neural network with the genetic algorithm. This procedure is suitable for solving optimization problems that need some kind of heuristics to limit the search space. If a reasonable amount of data is available, the ANN can ''understand'' the problem and it can provide the genetic algorithm with a selected population of elements that will speed up the search for the optimum solution.

We have tested the method to search for the ground-state

geometry of silicon clusters. This choice is relevant because the theoretical determination of these geometries remains an open issue and because it is an NP-hard optimization problem. We trained the ANN with information of geometry and energetics of small silicon clusters ($Si \leqslant 9$). Next, the ANN learned how to restrict the configurational space for larger silicon clusters. In the case of Si_{10} and Si_{20}, we noticed that the NAGA is at least three times faster than the ''pure'' genetic algorithm. As the size of the cluster increases it is expected that the gain in terms of time will increase as well.

This work reinforces other evidence [24] that artificial intelligence algorithms may become an important tool to solve problems that are not suitable to conventional computational methods. We believe that it is fair to say that AI methods may eventually replace human experience in cases such as NP-hard optimization problems.

ACKNOWLEDGMENTS

This work is partially supported by CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) and Fapesp (Fundação de Amparo à Pesquisa do Estado de São Paulo).

[1] M. F. Jarrold, Science **252**, 1085 (1991).
[2] K. M. Ho, A. A. Shvartsburg, B. Pan, Z. Y. Lu, C. Z. Wang, J. G. Wacker, J. L. Fye, and M. F. Jarrold, Nature (London) **392**, 582 (1998).
[3] B. Liu, Z. Y. Lu, B. Pan, C. Z. Wang, K. M. Ho, A. A. Shvartsburg, and M. F. Jarrold, J. Chem. Phys. **109**, 9401 (1998).
[4] C. Miller, Science **252**, 1092 (1991).
[5] E. Kaxiras and K. Jackson, Phys. Rev. Lett. **71**, 727 (1993).
[6] E. Kaxiras and K. A. Jackson, Z. Phys. D: At., Mol. Clusters **26**, 346 (1993).
[7] J. C. Grossman and L. Mitas, Phys. Rev. B **52**, 16 735 (1995).
[8] J. C. Grossman and L. Mitas, Phys. Rev. Lett. **74**, 1323 (1995).
[9] M. R. Lemes, C. R. Zacharias, and A. Dal Pino, Jr., Phys. Rev. B **56**, 9279 (1997).
[10] B. X. Li and P. L. Cao, Phys. Rev. A **62**, 023201 (2000).
[11] B. Hartke, J. Comput. Chem. **20**, 1752 (1999).
[12] L. T. Ville and J. Vennik, J. Phys. A **18**, L419 (1985).
[13] G. W. Greenwood, Z. Phys. Chem. (Munich) **105**, 211 (1999).
[14] T. R. Cundari and E. W. Moody, J. Chem. Inf. Comput. Sci. **37**, 871 (1997).
[15] K. Laasonen and M. Nieminen, J. Phys. C **2**, 1509 (1990).
[16] J. Hertz and R. G. Palmer, *Introduction to the Theory of Neural Computation* (Addison-Wesley, Reading, MA, 1991).
[17] J. H. Holland, *Adaption in Natural and Artificial Systems* (University of Michigan Press, Ann Arbor, 1975).
[18] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning* (Addison-Wesley, Reading, MA, 1989).
[19] M. Mitchell, *An Introduction to Genetic Algorithms* (MIT Press, Boston, 1996).
[20] R. S. Judson, Rev. Comput. Chem. **10**, 1 (1997).
[21] D. M. Deaven and K. M. Ho, Phys. Rev. Lett. **75**, 288 (1995).
[22] D. M. Deaven, N. Tit, J. R. Morris, and K. M. Ho, Chem. Phys. Lett. **256**, 195 (1996).
[23] C. R. Zacharias, M. R. Lemes, and A. Dal Pino, Jr., J. Mol. Struct.: THEOCHEM **430**, 29 (1998).
[24] J. J. Hopfield and D. W. Tank, Science **233**, 625 (1986).