Determination of bound-state wave functions by a genetic algorithm

Christian Winkler* and Hartmut M. Hofmann
Institut für Theoretische Physik III, Staudtstraße 7, D-91058 Erlangen, Germany

(Received 12 June 1996)

We present a stochastic method of minimizing the ground state energy in variational calculations of light nuclei using the refined resonating group model. The method utilizes a bit representation of the width parameters to be varied. To find the best possible set of width parameters we use strategies familiar from biological evolution. Very complicated problems can be solved in this way because the method is intrinsically parallel. The algorithm can be used on parallel computers with any number of processors without any change. As an example we give the results of a simple model calculation of ^{7}Li. [S0556-2813(97)06102-5]

PACS number(s): 21.60.−n, 02.60.Pn, 02.70.Rw, 21.10.Dr

I. INTRODUCTION

In the framework of the refined resonating group model [1] bound states and scattering observables of quite complicated nuclei have been calculated successfully [2]. In this model a nucleus is decomposed into clusters. For details see [1]. Here we give only the essentials of the wave functions used. The basic orbital wave function of a cluster is determined by a Gaussian function. The wave function of the relative motion again consists of a Gaussian multiplied by solid spherical harmonics. The total wave function is constructed as a linear superposition of antisymmetrized products of orbital and single particle spin functions. It might be necessary to allow for more than one width parameter per cluster and to include different decompositions into clusters, e.g., $^7\mathrm{Li}={}^4\mathrm{He}-{}^3\mathrm{H}$ and $^6\mathrm{Li}-n$.

However, as soon as the nuclei get too complex and many width parameters are involved it is very difficult to find minima in the space of the highly nonlinear width parameters. Calculations are very time intensive since a single evaluation of the binding energy for a fixed set of width parameters can already take a considerable amount of CPU time.

Traditional methods for finding minima in high dimensional spaces usually require lots of different function evaluations. Additionally it can happen that these methods stop in a local minimum and miss the global one. A further disadvantage is the fact that all function evaluations have to be done sequentially because each new test point in the parameter space depends on the function's value of the previous points. For this reason the search for the optimal width parameters takes an enormous amount of time.

One approach to finding a smaller binding energy would be simply to extend the number of linear combinations taken into account [3]. However, this leads to a very unphysical model where it is hard to interpret the contribution of each configuration. In addition to that, scattering calculations become almost impossible. Therefore it would be nice to have a method that is able to find a minimum of the binding energies in the space of a fixed (but smallest possible) number of width parameters. Ideally all this should be done as fast as possible in order to be able to calculate complicated nuclei as well.

Using our method the general procedure for studying ground and scattering states of complicated nuclei starts from the ground states of light nuclei. These structures are then incorporated into the heavier nuclei. This can be done many times and so the states of heavier and heavier nuclei can be calculated step by step. For the helium isotopes this has been carried out by [7].

For this purpose we need analytical expressions for the wave functions in a clearly defined model space which can be handled by more complex problems. This has to be regarded in contrast to other approaches like GFMC, which are more interested in exact (numerical) results on a kind of lattice with a small but finite spacing.

II. OUTLINE OF THE GENETIC ALGORITHM

Here we briefly summarize the essential points taken from [4]. Suppose we want to find a minimum in a space spanned by p parameters. The basic idea of the method is then to have bit representations (called *genes*) of all p optimization parameters which is called an *individual*. Therefore an individual consists of p genes. Initially we have to create a certain number n of individuals (a *population* of size n) with random genes. This is called the first generation. In an evolutionlike method we get from one generation to the next by selecting two ''good'' individuals. These will now have the chance to produce two *offsprings* with a crossoverlike mechanism acting on the bit representations.

As we would like to minimize our binding energies with respect to the width parameters, we chose a binary representation of 16 bit length (i.e., a 16 bit number) for each parameter. This is sufficiently precise. The genes are mapped linearly onto the width parameters but that can easily be changed by using different mappings (e.g., to enhance the resolution in certain regimes of the parameter space).

The algorithm itself consists of five steps

(1) *Initialization*. Create an initial population by randomly choosing uniformly distributed 16 bit numbers. Calculate the binding energies of all members and sort them in ascending order.

(2) *Selection*. Choose two members of the population according to their binding energy. Choosing members with

*Electronic address: cwinkler@theorie3.physik.uni-erlangen.de

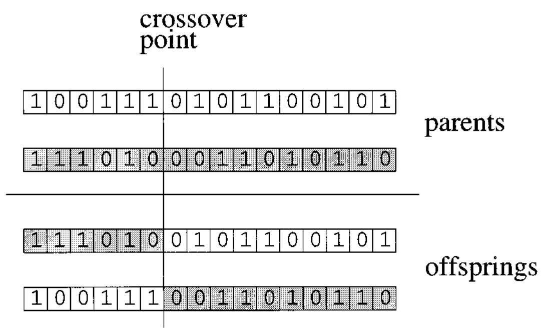

FIG. 1. Example of a one point crossover operation between two genes.

lower binding energy should be more probable. This can be implemented using strategies like roulette wheel or rank selection [4].

(3) *Crossover*. Use the bit representations of the two selected members for a crossover to produce two new offsprings [4]. Crossover can be done in many different ways, the most famous of which is one point crossover (see Fig. 1). However, in most practical applications, as in our case, uniform crossover (a generalized one point crossover = many point crossover) is used [4].

(4) *Mutation*. Mutate (i.e., invert) each bit in the offsprings with a given probability. This is done to ensure that the population does not become degenerate (and hence get stuck in a local minimum) if all bit representations are similar.

(5) *Insertion*. Calculate the binding energies of the two offsprings and insert them into the population. The ''worst'' individuals are thrown out to have a constant population size. Return to step (2).

One cycle from step (2) to step (5) is called one *generation*. If the algorithm converges, e.g., after 100 generations with a population size of 50 one will need $50+100\times 2=250$ function evaluations.

The algorithm can terminate after, e.g., a fixed number of generations has been calculated or the mean binding energy of the whole population is sufficiently close to the lowest binding energy of the population so that no drastic further change should be expected.

We have to emphasize that the most important process which leads eventually to convergence is the crossover operation. The mutation is only done to ensure diversity in the population. This can be compared to the process of biological evolution where it is also thought that crossover is the most important step.

Usually the most time consuming task is to evaluate the binding energy for the new parameters in step (5). All other tasks are more or less just bookkeeping. Therefore we note that the whole algorithm is perfectly well suited for massively parallel computation: each evaluation of the binding energy can be done on a single processor.

To achieve maximal performance on any parallel computer system our algorithm proceeds as follows: all available processors are used for calculating the binding energies of the initial population. In the selection process two individuals are selected and their offsprings' binding energies are evaluated as long as free processors are available. If no further processor is available the program waits for the binding energy returned by one of the processors which are sorted into the current population. Then two new individuals are selected and their offsprings are sent to the free processors. In this way we try to assure maximum parallelism.

The time needed for computing a fixed number of generations is therefore almost *inversely proportional* to the number of processors available. Hence the power of the algorithm grows automatically with the number of processors available.

At this point one might argue that all stochastic methods are trivially parallel. This is of course true but population based approaches exhibit even another degree of parallelism as they permit communication between the processes by means of the population as a kind of ''pool.'' This leads eventually to faster results (in real time). This is true for all population based methods, be it a genetic algorithm or, e.g., a GFMC calculation. However, not all these methods are suitable for minimization of functions and that is why we have chosen the genetic algorithm. We have tried different strategies for updating the population like stochastic averages of the width parameters. These proved however to be not as stable against being stuck in local minima and did not converge as rapidly as the genetic algorithm.

Comparing with traditional (nonpopulation based) stochastic techniques, this has considerable advantages regarding computing time. If, e.g., simulated annealing converges after 200 steps and each function evaluation takes one hour the earliest results can be obtained in 200 hours irrespective of the number of processors available. On the contrary if the genetic algorithm takes 500 function evaluations to converge the computation time can be made shorter by simply providing enough processors (e.g., with 50 processors it will be only 10 hours). We still can make stochastic statements about the quality of the solution by taking into account the whole population (average, variation, etc.).

III. SIMPLE EXAMPLE

We tried to apply all our considerations to a simple model problem to see how well it works. For this end we chose the ^{7}Li example. This is not too simple as five different width parameters are used. On the other hand, calculations do not take too long so that it is easy to compare the results by the genetic algorithm with those from a deterministic (quasi-Newton) search.

This section is just meant as a demonstration for the method itself, we wanted to see how well it performs on a problem with known exact solution to be able to compare it to other methods.

As a first test we applied the genetic algorithm several times to see if the method converges on the average. In all calculations we used a population size of 50, a mutation rate of 0.001 and the number of total generations was fixed to be 500. The results displayed in Fig. 2 show a reasonable convergence. Of course one has to do several runs to find reliable results but this is no difference to the deterministic method where it is possible to become stuck in a local minimum.

Note that even when the genetic calculation has stabilized basically still the whole space defined by the mapping of the genes to the width parameters is used for finding better val-

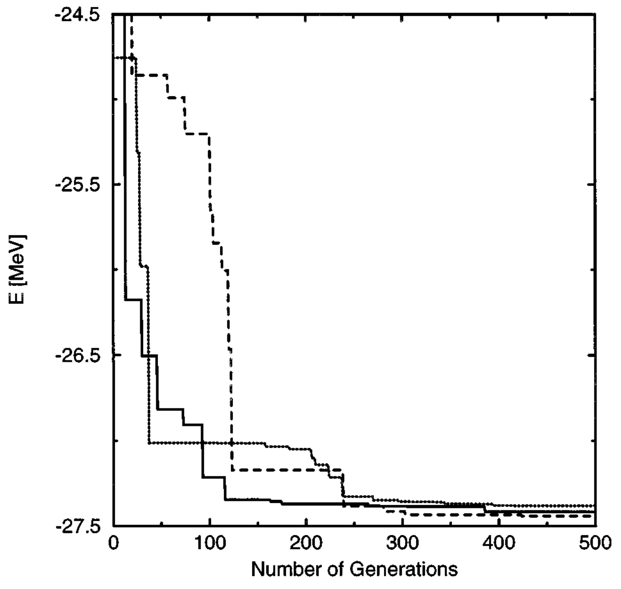

FIG. 2. Convergence of three different runs of the genetic algorithm trying to find the ground state of ^{7}Li (only the binding energy of the best individual of the generation is shown).

ues for the binding energy. At this point it is worthwhile stopping the algorithm and starting it again with a new mapping which takes account of the width parameters just found. This can be done several times to enhance the resolution of the method and to be sure that the global minimum is really found.

The rate of convergence is of course independent of the number of processors used because all ''administrative'' tasks are done in the main program which does not depend on the number of processors.

We used the results from the solid line in Fig. 2 again to estimate how well the genetic algorithm performs compared to a deterministic search method. Therefore we applied the deterministic search algorithm from the NAGLIB [5] (a quasi-Newton algorithm) and plotted in Fig. 3 the number of necessary steps together with the results from the genetic algorithm. Note, however, that the only physical input to the genetic algorithm was the *range* of physically sensible width parameters whereas the deterministic search already needs good *starting values* to yield a reasonable performance. This can be seen by the much better first energy value in the deterministic method.

However, it must be emphasized that the actual time used for getting the results is the CPU time divided by the number of processors used in the parallel implementation (except for bookkeeping tasks which can be totally neglected if the function evaluation takes most of the CPU time). Therefore it is only of minor interest that the genetic algorithm needs about twice as many steps (twice the CPU time) as a deterministic sequential method. This drawback is easily compensated by the number of available processors.

To get a feeling about the time needed for complicated calculations in both methods we compare the *real time* in Fig. 4. The genetic algorithm of course starts later since the initial population has to be calculated first. After that the genetic algorithm running on 50 processors converges dramatically faster than the deterministic method.

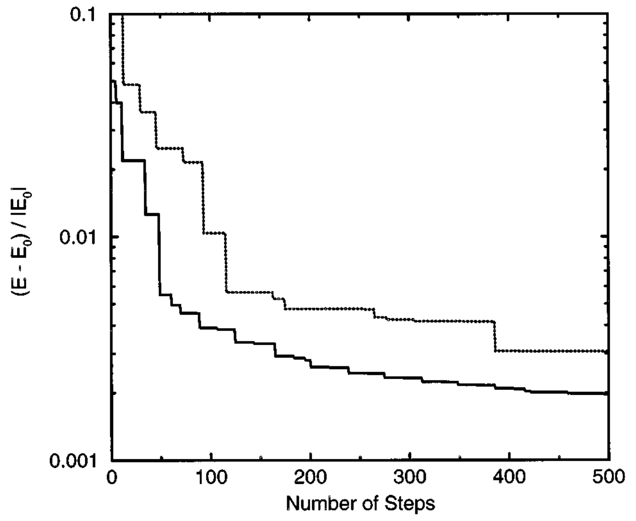

FIG. 3. Convergence of the binding energy as a function of the number of steps used in a genetic algorithm (dotted line) and a deterministic type sequential method (in the deterministic method only the minimal energy found so far is plotted).

To summarize this section we would like to point out that calculations of nuclear ground states using parallel genetic algorithms seem to be very fast and should be preferred against sequential methods. Minimizing the ground state energy of complicated nuclei might become possible in much shorter (real) time. The algorithm is very flexible because parameters like the size of the population, mutation rate, selection scheme, etc. can easily be adjusted to suit the problem [6].

IV. APPLICATION AND OUTLOOK

After having found that the genetic algorithm works in the expected way for a problem with a known solution we ap-

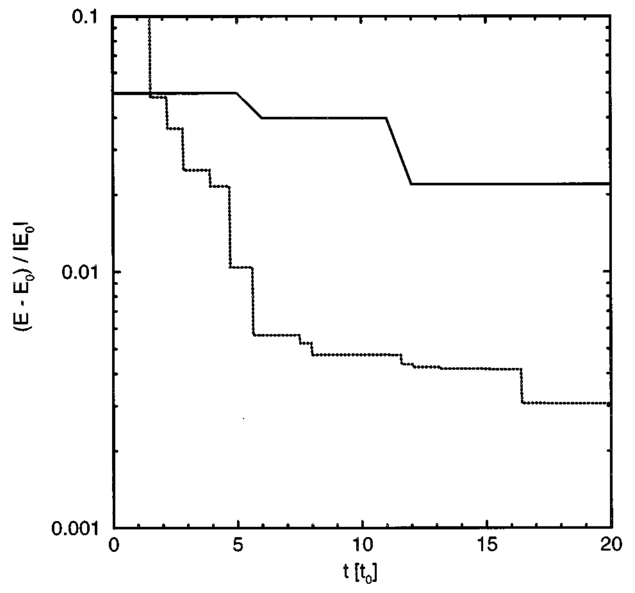

FIG. 4. Comparison of the real time used to find the minimum of binding energies using a genetic algorithm running on 50 processors (dotted line) and a sequential deterministic type search.

plied it to a far more complex nuclear physics problem, the calculation of the properties of the helium isotopes, especially ^{8}He. As we want to separate the description of our new method from the discussion of the physical input (model space, potentials, etc.), the results are presented in a separate paper [7]. The calculations would not have been possible without the new method because of the total CPU time used, the actual calculation has been performed on a Convex SPP computer using 32 processors.

We have shown that a very easy algorithm ‘‘copied’’ from nature can be used to calculate binding energies and wave functions of rather complicated nuclei. As this can be done in a highly parallel manner and is fully scalable many new problems can be solved in this way. The algorithm is extremely simple and can be generalized to almost any kind of problem where the determination of an extremum of a function is involved.

Of course all these computational methods are not restricted to nuclear physics. Indeed the first test trying to estimate the convergence of the method was done by calculating the ground state wave function of the hydrogen atom using the Ritz variational method. So there should not be any difficulty in using the same methods for quite complicated quantum chemistry calculations and atomic cluster calculations.

One possible extension of the algorithm is some kind of self-adaptive behavior, i.e., changing the mapping of the genes to the width parameters dynamically. This is currently under development.

ACKNOWLEDGMENTS

This work was supported by DFG and BMBF.

[1] H. M. Hofmann, in *Models and Methods in Few Body Physics*, edited by L. S. Ferreira, Lecture Notes in Physics Vol. 273 (Springer, Berlin, 1987).
[2] H. M. Hofmann, Nucl. Phys. **A416**, 363c (1984).
[3] K. Varga, Y. Suzuki, and R. G. Lovas, Nucl. Phys. **A571**, 447 (1994).
[4] Zbigniew Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs* (Springer-Verlag, Berlin, 1994).
[5] Numerical Algorithm Group (Oxford), NAGLIB, Fortran Library.
[6] S. W. Mahfoud and D. E. Goldberg, in *Parallel Problem Solving from Nature*, edited by R. Manner and B. Manderick (Elsevier, Amsterdam, 1989).
[7] J. Wurzer and H. M. Hofmann, following paper, Phys. Rev. C **55**, 688 (1997).