Direct search for wave operator by A Genetic Algorithm (GA): Route to few eigenvalues of a Hamiltonian

Rahul Sharma and S. P. Bhattacharyya

Abstract—A Genetic Algorithm is invoked to search out the wave operator leading to the determination of a few eigenvalues and eigenvectors of a specially designed real symmetric matrix (Durand matrix) that simulates a Hamiltonian supporting bound states coupled to continuum.

The performance is compared with that of a standard iterative method for different partition sizes, and parallelizability of the GA-based approach is tested.

In many cases the GA-based approach smoothly converges while the standard iterative schemes diverge.

I. INTRODUCTION

The calculation of the energy levels of atoms and molecules are routinely done now-a-days by using large but finite basis sets to represent the Hamiltonian and diagonalizing it numerically in the N-dimensional space spanned by the basis. Frequently, one requires information about only a few of the lowest energy levels \((m < N)\). In such situation it has been considered prudent to work with a modified Hamiltonian matrix in the \(m\)-dimensional space which generates the \(m\) exact eigenvalues and the projection of the \(m\)-exact eigenvectors on this lower dimensional space. The eigenvectors in the full space are then recovered by forming the appropriate wave operator and solving the non-linear equation it satisfies. One of the ways of finding the wave-operator is through an energy-dependent partitioning of the matrix eigenvalue problem [1 – 4] while an equivalent energy-independent partitioning method also exists [5 – 7]. The latter leads to a non-linear matrix equation which must be solved to obtain the wave operator and the eigenvalues and eigenvectors.

The present communication suggests a direct search method for finding the wave operator by Genetic Algorithms [8 – 9]. Stochastic Diagonalization by Monte Carlo [10 – 11] or by Genetic Algorithms has received some attention in the recent times [12 – 13] although the deterministic methods are still the methods of choice for large scale eigen problems [14 – 17]. However, if the reduced problem has a built-in non-linearity, the stochastic methods could be of use. Since finding the wave-operator involves solution of a non-linear matrix equation, the Genetic or Evolutionary algorithms are expected to be useful for finding the solution matrix. The main objective of this communication is to demonstrate that non-deterministic methods like the GA can explore and exploit the information present in the search space under extreme situations where other methods could fail to find out the eigenvalues through the wave operator route and home on to the eigenvalue and the corresponding eigenvector, without any preconditioning of the initial population. The technique used is easily parallelized and scalable to higher dimensions.

II. THE METHOD

Let \(H\) be the real-symmetric Hamiltonian matrix \((N \times N)\) which is transformed into a partitioned matrix-eigenvalue problem in \(n_a\) and \(n_b\) dimensional subspaces, respectively, leading to two sets of equations:

\[
H_{aa} C_a + H_{ab} C_b = E C_a
\]

\[
H_{ba} C_a + H_{bb} C_b = E C_b
\]

The energy-dependent effective Hamiltonian \((H_{eff}^a)\) that generates \(n_a\) exact eigenvalues can be obtained by eliminating \(C_b\) from the two equations by using [1], [2], [3], [4]

\[
C_b = \{(E \mathbb{1}_b - H_{bb})^{-1} H_{ba}\} C_a = W C_a
\]

where \(W\) is the wave operator (here energy-dependent)

Using equation(3) in equation(1) we get

\[
H_{eff}^a C_a = (H_{aa} + H_{ab}(E \mathbb{1}_b - H_{bb})^{-1} H_{ba}) C_a = EC_a
\]

The energy-dependence can be removed by setting \(W = Z_{ba}\), whence

\[
H_{eff}^a C_a = (H_{aa} + H_{ab} Z_{ba}) C_a = EC_a
\]

and finding out the non-linear equation that \(Z_{ba}\) satisfies. It is straightforward to show that \(Z_{ba}\) satisfies equation (6)

\[
H_{eff}^a C_a = (H_{aa} + H_{ab} Z_{ba}) C_a = EC_a
\]

which in turn demands that \(Z_{ba}\) be so chosen that [1], [5], [6]

\[
X = H_{ba} + H_{bb} Z_{ba} - Z_{ba}(H_{aa} + H_{ab} Z_{ba}) = 0
\]

We propose that equation (7) (non-linear matrix equation) can be solved by GA directly, by allowing it to search out the appropriate \(Z_{ba}\) matrix. Once such a \(Z_{ba}\) has been found, we may go back to the effective Hamiltonian of equation (5) and diagonalize it to obtain the \(n_a\) desired eigenvalues and the \(a\)-space projections (\(C_a\)) of the corresponding eigenvectors in the \(N\)-dimensional space. The \(b\)-space projections of the eigenvectors are found by using equation (8)

\[
C_b = Z_{ba} C_a
\]
If the $a$-space is chosen to be one-dimensional, $C_a$ can be determined directly through equations (8) ($C_a = 1$), once $Z_{ba}$ has been found.

The application of a GA to find solutions of the non-linear matrix equation (7) is straightforward. The population of potential solution matrices are represented as $N_{pop}$ strings, containing $N (= n_b \times n_a$) floating point numbers, each representing one element of the $Z_{ba}$ matrix. The fitness of each string is defined as follows [12 – 13]:

$$f_i = e^{-\lambda \sigma} (0 < f_i \leq 1),$$

where

$$\sigma_i = \frac{1}{\sum_{k=1}^{n_b} \sum_{l=1}^{n_a} X_{kl}^2}$$

(10)

$\lambda$ is a user defined parameter to take care of exponential overflow/underflow. The population of initial solution strings are allowed to undergo a Roulette-wheel selection process (fitness proportional selection) to form the mating pool and randomly chosen strings from the mating pool are allowed to undergo an arithmetic crossover process with a fixed crossover probability $p_c$. If the $i$th and the $j$th strings are selected for crossover, a pair of random integers $(m, n)$ are selected and the arithmetic crossover between the strings are carried out as follows [12]:

$$(Z^{(i)}_{ba})_{pq} = f(Z^{(i)}_{ba})_{pq} + (1 - f)(Z^{(j)}_{ba})_{pq}$$

(11)

$$(Z^{(j)}_{ba})_{pq} = f(Z^{(j)}_{ba})_{pq} + (1 - f)(Z^{(i)}_{ba})_{pq}$$

(12)

where the mixing parameter $f$ varies randomly within the range [0.5, 1].

The mutation process was carried out on all post crossover strings with a mutation probability $p_m = 4/N$, where $N$ is the total number of variables in a string, ensuring that a maximum of four variables in the string could undergo mutation at any stage.

$$(Z^{(i)}_{ba})_{pq} = (Z^{(i)}_{ba})_{pq} + (-1)^l \cdot r \cdot \Delta ^{(i)}_{pq} (i_{gen})$$

(13)

In (13) $l$ is a random integer, $r$ is a random number in the range [0, 1], $\Delta ^{(i)}_{pq} (i_{gen})$ is a dynamically evolving mutation intensity. Since we are constrained to use a small population size, there is always possibility of premature convergence.

To forestall premature convergence, a diversification operator is used in order to maintain enough variability in the population. It is a mutation-type process described in equation(13) with a probability $p_d (= 4/(N \times N_{pop})$ and $\Delta ^{d} (i_{gen})$ is the diversification intensity. $p_d$ was thus kept one order lower than $p_m$.

The mutation and diversification intensities are made to evolve dynamically with generations depending on what fraction of the offsprings in the population ($F_p$) were selected in the previous generation under a fully elitist scheme.

Thus if $F_p < 0.1$, the population is parent dominated. So either the offsprings have suffered undesirable mutation or there is inadequate diversity among them. We therefore adjust mutation or diversification intensities according to the following rule:

- a random number is generated ($0 \leq r \leq 1$) and if $r < 0.5$,

$$\Delta ^{m} (i_{gen}) = \Delta ^{m} (i_{gen} - 1) \times [1.0 - \delta]$$

and $\Delta ^{d} (i_{gen}) = \Delta ^{d} (i_{gen} - 1) \times [1.0 + \delta]$, where $\delta$ is a fixed small value ($< 0.1$) defined by the user and is held fixed throughout the generations.

If $F_p > 0.2$, the reverse situation prevails and the imbalance is sought to be corrected as follows:

- for $r < 0.5$,

$$\Delta ^{m} (i_{gen}) = \Delta ^{m} (i_{gen} - 1) \times [1.0 + \delta]$$

Otherwise, we have the diversification intensity modified as follows

$$\Delta ^{d} (i_{gen}) = \Delta ^{d} (i_{gen} - 1) \times [1.0 - \delta]$$

This is enforced throughout ensuring the values of $\Delta ^{m} (i_{gen}), \Delta ^{d} (i_{gen})$ stay within the range $[1.0 \times 10^{-12}, 100]$. The present choice of mutation probability, dynamic adjustment of mutation and diversification intensities improve the performance of the GA substantially. Further tuning is possible and is under way.

After crossover, mutation and diversification operations are over, we compare the fitness values of all the parent and offspring strings and select the best $N_{pop}$ among them for the operation of selection. We have used $p_c = 0.8$, crossover mixing parameter $f$ = random number from the range [0.5, 1], $\Delta ^{m} (0) = 1.0$, $\Delta ^{d} (0) = 1.0 \times 10^{-5}$.

The results that are presented here refer to a particular realization of the GA-based run. The performance can be somewhat better or inferior depending upon the choice of the initial population. The average behaviour is however not much different from what has been presented.

### III. RESULTS AND DISCUSSION

![Evolution of fitness function corresponding to the best string, during the search for the $Z_{ba}$ matrix for three values of $\mu$. The search converges to the perturbed impurity state.](image)

We have tested the proposed method with a model Hamiltonian [6] that supports a discrete bound state ($\epsilon_s \sim 0$)
coupled to the continuum. The Hamiltonian matrix is defined as follows:

\[
H_{ii} = 0, \quad H_{ij} = \lambda \quad \text{for } i = 1, j = 2, ..., N
\]

\[
\mu \quad \text{for } i \neq 1, j = i - 1 \text{and } i + 1
\]

\(N\) being the dimension of the Hamiltonian.

[1] models an impurity state coupled to the states of linear polyene \((\{j\} = 2, ..., N)\), the coupling strength being \(\lambda\). \(\mu\) is related to the width of the Huckel band. The target state is the perturbed impurity state and states around it. If \(\mu = 0\), \(\lambda = 0.005\) is used, \(\mu\) is set to be equal to 0.6 (or \(\geq 0.5\)) and \(N\) is set to 50, the perturbed impurity state cannot be obtained by invoking a Gauss-Seidel type of iterative method to solve equation (7) (starting with \(Z_{ba} = 0\), it converges to the highest eigenvalue) or by the standard Davidson’s method [17] which converges to the lowest eigenvalue. GA-based search with one-dimensional \(a\)-space converges to the perturbed impurity state \((\epsilon_i \sim 0.0)\).

Fig. 2. Evolution of error in the eigenvalue computed with the best string, during the search for the \(Z_{ba}\) matrix for three \(\mu\) values: (a) Using GA, (b) The same profiles in the case of GS-type iterative method.

<table>
<thead>
<tr>
<th>Error in Eigenvalue</th>
<th>No. of Generations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1e-08</td>
<td>1e-07</td>
</tr>
<tr>
<td>1e-06</td>
<td>1e-05</td>
</tr>
<tr>
<td>1e-04</td>
<td>0.001</td>
</tr>
<tr>
<td>0.01</td>
<td>0.0005</td>
</tr>
<tr>
<td>0.05</td>
<td>0.000624648481</td>
</tr>
<tr>
<td>0.1</td>
<td>0.000624648481</td>
</tr>
<tr>
<td>0.2</td>
<td>0.000624648481</td>
</tr>
<tr>
<td>0.3</td>
<td>0.000624648481</td>
</tr>
<tr>
<td>0.4</td>
<td>0.000624648481</td>
</tr>
<tr>
<td>0.5</td>
<td>0.000624648481</td>
</tr>
<tr>
<td>0.6</td>
<td>0.000624648481</td>
</tr>
<tr>
<td>0.7</td>
<td>0.000624648481</td>
</tr>
<tr>
<td>0.8</td>
<td>0.000624648481</td>
</tr>
<tr>
<td>0.9</td>
<td>0.000624648481</td>
</tr>
<tr>
<td>1</td>
<td>0.000624648481</td>
</tr>
</tbody>
</table>

Figures 1-2(a) show how the fitness and the error in the computed eigenvalue of the perturbed impurity state corresponding to the best \(Z_{ba}\) string in the population evolve as the search runs through generations, for three different values of the impurity-polyene coupling strength \(\mu(=0.5, 0.6, 0.7)\). In each case, a population size \(N_{pop} = 10\) was used and the search started with a null \(Z_{ba}\) matrix. Figure 2(b) displays error evolution profiles for calculations based on the GS-type of iterative method. In each case the traditional method also converges. However, the targeted eigenvalue is obtained only when \(\mu = 0.5\). For all other values of \(\mu\) the iterations converge to a wrong eigenvalue, usually the largest one (Figure 2(b)). The GA-based search on the other hand, always converges to the bound state in the continuum (near zero eigenvalue, figure 1(a)).

Once the eigenvalue of the perturbed impurity state \((\epsilon_0)\) and the corresponding eigenvector \(C_0\) have been found, we form the shifted full space matrix \(H = H_0 + \beta P_0\) where \(P_0\) is the full space projector for the perturbed impurity eigenstate \((P_0 = C_0 C_0^\dagger)\), re-order the rows and carry out the GA-based search with the same initial choice of the \(Z_{ba}\) strings. The GA search now converges to a higher eigenvalue. Figures 3(a-b) show how the fitness evolution takes place for the second or the higher eigenvalues, and figures 4(a-b) show the corresponding error profiles for evolution of higher eigenvalues. By varying the initial choice of \(Z_{ba}\) strings, and the shift \(\beta\), we may exhaust all the higher eigenvalues.

Once again, the GA-based search for a higher eigenvalue converges for \(\mu = 0.5, 0.6, 0.7\) (Figure 4(a)).
iterations converge only when $\mu = 0.5$. For $\mu = 0.6$ or 0.7 the iterations diverge. The GA, on the contrary, can be seen to remain effective when the next higher eigenvalue is sought (figure 4(c)) for all the values of $\mu$ used. The GS-type of iterative search converge for $\mu = 0.5$, but diverge for all the other two values of $\mu$ (0.6, 0.7).

Fig. 4. Error Reduction profiles, during the search for $Z_{ba}$ matrices for: (a) GA, (b) GS-type, the third $Z_{ba}$ matrix search using (c) GA, (d) GS-type. ($Z_{ba} = 0$ to start with in each case).

Eigenvalues determined:
(a) GA: $\mu = 0.5 \rightarrow 0.315453486$, $\mu = 0.6 \rightarrow 0.701572992$, $\mu = 0.7 \rightarrow 1.26233338$.
(b) GS-type: $\mu = 0.5 \rightarrow 0.00197365061$, diverges for the rest two cases.
(c) GA: $\mu = 0.5 \rightarrow 1.0000001$, $\mu = 0.6 \rightarrow 1.0000001$, $\mu = 0.7 \rightarrow 1.0000001$.
(d) GS-type: $\mu = 0.5 \rightarrow 0.007885298$.

Fig. 5. Fitness function evaluation, during the search for $Z_{ba}$ matrix for: (a) $n_a = 2$, (b) $n_a = 3$, (c) $n_a = 5$.

If the $a$-space is chosen to be 2-dimensional, 3-dimensional or 5-dimensional, respectively, and $\lambda = 0.6$, $\mu = 0.005$, are used, GS-type of iterations diverge. However, as figures 5(a-c) would indicate the fitness evolution in the GA-based method remains smooth in all these cases. The corresponding error reduction profiles are displayed in figures 6(a-c), all of which converge. GS-type of iterations, on the other hand, diverge in all these cases. A typical example is displayed in figure 6(d).

Once the $Z_{ba}$ matrix corresponding to the string of the highest fitness is obtained, the non-Hermitian matrix $H_{eff}$ of equation (6) ($2 \times 2$, $3 \times 3$, $5 \times 5$) are formed and diag-
Fig. 6. Sum of Absolute Error in Eigenvalue evaluation, during the search for $Z_{ba}$ matrix for: (a) $n_a = 2$, (b) $n_a = 3$, (c) $n_a = 5$. Eigenvalues determined:

GA: (a) $n_a = 2 \rightarrow -0.000561527776, 1.000001$
(b) $n_a = 3 \rightarrow -0.000561527776, 0.235091212, 1.76490879$
(c) $n_a = 5 \rightarrow -0.015560161, -0.000561527776, 0.558251363, 1.441759063, 2.051568792
GS-type: diverges for all the cases reported (d) $n_a = 2$ (initially was converging to $-0.197632097, -0.19057642$).

The most time consuming step is the evaluation of the fitness values of the string representing $Z_{ba}$ matrix. The step is easily parallelized. The performance of the search for $Z_{ba}$ matrix (solution of equation (7)) as more and more processors are switched on is displayed in Figure 7. The computations have been carried out on a parallel Linux cluster of PCs using the master-slave architecture (Gigabit switch is used for master-slave communication and the master is equipped with a gigabit ethernet card while 10 x 100 Mbps ethernet cards are used on the slave nodes). The $H$ matrix (50 $\times$ 50) corresponds to $\mu = 0.6$, and $\lambda = 0.6$. The GS iterations diverge in this case. The GA, on the other hand, succeed in finding the solution matrix ($Z_{ba}$). From the elapsed time versus number of processors plot it is clear that there is a competition between the communication and processing times and that leads to non-linear performance scaling with the number of processors [18]. For population size of 10, the optimal number of processors turns out to be 6 in the present example. As already mentioned, the GS-type of iterations converge much faster than the GA-based search when the GS-type of calculations converge. For example, in the particular case of $\mu = 0.5$, $\lambda = 0.005$ ($N = 50$, $n_a = 2$) and null vector being used as initial guess vector, elapsed time for the GS-type of search (per eigenvalue) was 3.13 sec., serial GA based calculation was 303.321 sec. and the parallelized GA consumed 93.22 sec of elapsed time ($N_{pop} = 10$, no. of processors = 10). However, preprocessing of the initial population of vectors and fine tuning of GA-parameters can lead to further improvement.

IV. CONCLUSIONS

GA is found to be useful for solving the non-linear matrix equation that determines the wave-operator through an energy-independent partitioning scheme leading to few of the eigenvalues and eigenvectors of a real symmetric matrix where routinely used strategies fail to converge. The computational cost is high, but can be reduced substantially by parallelization of the fitness evaluation and fine tuning of the GA. We have not evaluated the effects of preprocessing the input strings which also can cutdown further the computing
cost. The real strength of GA in problems addressed here is its ability to handle problems which diverge during a deterministic search.

ACKNOWLEDGMENT

Thanks are due to the DST, Government of India, New Delhi, for a generous research grant, and fellowship for RS.

REFERENCES