Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational Physics 208 (2005) 735–760
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

A genetic algorithm based inverse band structure method for semiconductor alloys

Kwiseon Kim *, Peter A. Graf, Wesley B. Jones

National Renewable Energy Laboratory, 1617 Cole Boulevard, Golden, CO 80401, USA

Received 19 October 2004; received in revised form 17 January 2005; accepted 7 March 2005
Available online 14 April 2005

Abstract

We present an efficient and accurate method for searching for atomic configurations with target band structure properties. Our approach to this inverse problem is to search the atomic configuration space by repeatedly applying a forward solver, guiding the search toward the optimal configuration using a genetic algorithm. For the forward solver, we relax the atomic positions, then solve the Schrödinger equation using a fast empirical pseudopotential method. We employ a hierarchical parallelism for the combined forward solver and genetic algorithm. This enables the optimization process to run on many more processors than would otherwise be possible. We have optimized AlGaAs alloys for maximum bandgap and minimum bandgap for several given compositions and discuss the results. This approach can be generalized to a wide range of applications in material design.

Keywords: Genetic algorithms; Inverse problem; Material design; Electronic structure; Pseudopotentials; Optimization

1. Introduction

With the increasing accuracy and efficiency of electronic structure methods, it is expected that one could use the ability of these methods to *predict* properties of materials as a tool for *designing* materials. In this paper we describe tentative steps in this promising direction; from a description of the properties desired of a material, the atomic configuration of the material itself is automatically generated.

* Corresponding author. Tel.: +1 303 275 4122; fax: +1 303 275 4007.
E-mail address: kwiseon_kim@nrel.gov (K. Kim).

0021-9991/$ - see front matter
doi:10.1016/j.jcp.2005.03.005

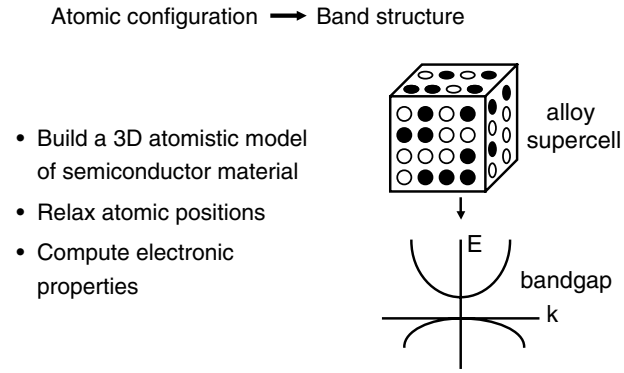

Fig. 1. Illustration of the forward solving method for atomistically calculating electronic structure of crystalline objects such as a semiconductor alloy.

The automatic design problem is cast as an optimization problem. We define one or more properties we wish the material to have, measure the extent to which a candidate material has them, then maximize this quantity. The part of the process where we compute the property in question for a given configuration of atoms we call the "forward solver", and the complete material design process is called the "inverse problem". Fig. 1 illustrates the method for solving the forward problem, typical in the electronic structure calculation of materials. A three-dimensional atomic configuration is constructed to model a physical system under study such as a bulk semiconductor, an alloy, or a quantum dot, and then the band structure is obtained by solving the Schrödinger equation.

The inverse problem asks which atomic configurations produce the desired electronic properties. Previous attempts to solve the inverse band structure problem are described in [1–4]. Fig. 2 describes our approach to solving such inverse band structure problems using optimization based on a genetic algorithm.

The organization of this paper is as follows. In Section 2 we describe the forward solver and the genetic algorithm based global optimization method which solves the inverse problem. Section 3 provides an illustrative application of the method in studying the variation of the bandgap of AlGaAs semiconductor alloys with composition and atomic configuration. We conclude with Section 4.

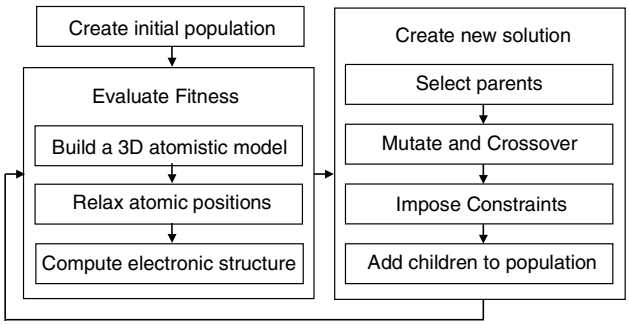

Fig. 2. Illustration of our inverse method. The left box represents the forward solver, and the right box represents the genetic algorithm based inverse solver. An initial population is created. The fitness of each member is evaluated by the forward solver. From this population and the fitness values, new individuals are created by the genetic algorithm. The population is updated and the process repeated until an acceptable solution has been found.

2. Method

2.1. Forward band structure method

Here we describe the framework for solving the forward problem based on the atomistic methods developed and popularized over the last 10 years [5–15]. This methodology is for calculating electronic structural properties of semiconductor alloys and nanostructures involving hundreds to hundreds of thousands of atoms. There are four elements to this methodology:

(1) generating the atomic configuration;
(2) relaxing the atomic positions;
(3) calculating the atomic potential;
(4) solving the Schrödinger equation.

Computationally, each step represents a program or routines that read different input files and generate output files that are used in the next step.

First, an atomistic model (supercell) that represents an alloy or a quantum dot is generated. Typically this is done by building a box with the underlying crystal structure and filling in the lattice points with atom identities. Throughout this paper we will be performing alloy calculations for zincblende crystals, whose underlying lattice is face centered cubic (FCC). Our unit cell is a cube containing eight atoms. The lattice vectors are in the cartesian directions, so the supercell is a repetition of this cube n_x, n_y, n_z times in each cartesian direction, respectively. This rectangular block of sites is now filled with the atom identities. For us, this step is performed by the genetic algorithm described below. We specify only the supercell size and composition and the optimization method fills and manipulates its contents. Finally, the alloy is modeled as the infinite periodic repetition of this supercell. Numerically, this just means that all the calculations are performed with the periodic boundary conditions. We refer to the atomic configuration as σ.

For a given σ, the atomic position relaxation is performed via a classical valence force field (VFF) method. A secondary purpose of this step is to calculate the local strain and the nearest neighbor atom environment used to overlap the atomic pseudopotentials in the next step. Continuum elasticity theory gives good estimates of cell external parameters for a supercell and we can use it to assign the overall lattice constants for the modeled quantum dot or alloy [6]. However, it does not give any information as to the positions and local strains of the atoms inside the supercell. To calculate the relaxed atomic positions within the supercell, we use a generalization (generalized VFF, GVFF) [6,9] of the original [16] valence force field model. The GVFF Hamiltonian (Eq. (24) in [6]) contains harmonic terms in bond length stretching, bond angle distortion and coupled bond stretching and distortion, and an anharmonic term in bond stretching. The bond stretching, bending, and length–angle interaction coefficients are related to the elastic constants of a pure material in zincblende structure (see Eq. (25) in [6]).

For the AlGaAs alloy studied in Section 3 the positions of the atoms were assumed to be in their ideal zincblende lattice sites because the lattice constants of AlAs and GaAs match closely. The lattice constants for AlAs and GaAs were taken as that of GaAs, 5.653 Å which translates into a bond length of 2.436 Å. For the bond angle, the ideal tetrahedral value of 109° is assumed. The atomic position relaxations for a non-lattice-matching alloy, GaInAs, are described in [6].

The third step in our forward solver is the construction of the atomistic Hamiltonian using the empirical pseudopotential method (EPM) [7–9]. When dealing with bulk-like atomic configurations that are free from defects or surfaces, this method can provide an accurate description of the electronic structural properties [1].

In principle the method is simple. The total potential of the alloy system is assumed to be the sum of potentials centered at each atomic site. The potential at each site in the *alloy* is in turn assumed to be

the linear combination of potentials representing the possible local environments of each atom in a *pure* binary compound (e.g., As surrounded only by Al, or As surrounded only by Ga). These potentials are of a specific functional form whose parameters are fitted to match data from experiment and/or local density functional theory calculations for the binary compounds contained in the alloy. We now describe some of the details.

The total pseudopotential of the system is written as the sum of screened atomic pseudopotentials centered at the atomic positions [6]:

$$V_{\mathrm{ps}}(\mathbf{r}) = \sum_n \sum_i v_i(\mathbf{r} - \mathbf{R}_n - \mathbf{d}_i), \tag{1}$$

where the sum over i is over all positions d_i within the supercell and the sum over n is over infinite periodic repetitions $\mathbf{R}_n$ of the supercell. When we assume v_i to be spherically symmetric (i.e., local in reciprocal lattice vector $\mathbf{G}$), we can write [7],

$$V_{\mathrm{ps}}(r) = \sum_n \sum_i v_i(|\mathbf{r} - \mathbf{R}_n - \mathbf{d}_i|). \tag{2}$$

If we now insert the Fourier transform of v_i and use the fact that $\mathrm{e}^{\mathbf{iG}\cdot\mathbf{R_n}} = 1$, we have [7]

$$V_{\mathrm{ps}}(r) = \frac{1}{\Omega} \sum_i \sum_{\mathbf{G}} \mathrm{e}^{\mathbf{iG}\cdot(\mathbf{d}_i - \mathbf{r})} v_i(|\mathbf{G}|), \tag{3}$$

where V_{ps} is normalized by the volume Ω of the supercell.

Now assume that $v_i(q = |\mathbf{G}|)$ is a function of the nearest neighbor environment. That is $v_i = v_{\alpha(i)}$, where $\alpha(i)$ specifies the local atomic environment at site i. For example, in our AlGaAs system, there are seven such environments, which we divide into two groups: the four unalloyed environments $\alpha_{\mathrm{u}} \in \{\mathrm{AlAs, GaAs, AsAl, AsGa}\}$ (where we write AB to indicate B atoms surrounded by A atoms) and the three mixed environments $\alpha_{\mathrm{m}} \in \{\mathrm{Al}_n\mathrm{Ga}_{4-n}\mathrm{As}\}$ for $n = 1, 2, 3$.

For each possible $v_{\alpha_{\mathrm{u}}}(q)$, an analytical functional form is assumed and the parameters are fitted to reproduce the band structure properties. Specifically, the atomic pseudopotentials $v_\alpha(q)$ for the α_{u} local environments are assumed to be linear combinations of four Gaussian functions, multiplied by a smooth function that allows adjustment of the small q components [7]:

$$v_\alpha(q) = \Omega_\alpha \sum_{j=1}^{4} a_{j\alpha} \mathrm{e}^{c_{j\alpha}(q - b_{j\alpha})^2} [1 + f_{o\alpha} \mathrm{e}^{-\beta_\alpha q^2}]. \tag{4}$$

Here, Ω_α is an atomic normalization volume. The parameters a, b, c, f, and β are fit to reproduce available electronic property data such as measured interband transition energies, effective masses, and deformation potentials of bulk GaAs and AlAs.

To take into account the effects of strain in the system, one adds dependence on the local strain ϵ to the pseudopotential $v_\alpha(q)$ [6,10]

$$v_\alpha(q, \epsilon) = v_\alpha(q, 0)[1 + \gamma_\alpha \mathrm{Tr}(\epsilon)], \tag{5}$$

where $\mathrm{Tr}(\epsilon)$ is the trace of the local strain tensor, which is calculated in the previous atom-relaxation step, and γ_α is a fitting parameter. The description of local strain is given in [6,17].

From these unalloyed pseudopotentials, the pseudopotential for the mixed environment α_{m} of As coordinated by $(4 - n)$ Ga atoms and n Al atoms is taken as the composition-weighted average written as [6],

$$v_{\mathrm{Ga}_{4-n}\mathrm{Al}_n\mathrm{As}} = \frac{4 - n}{4} v_{\mathrm{GaAs}} + \frac{n}{4} v_{\mathrm{AlAs}}. \tag{6}$$

The number of nearest neighbor cations (anions) around each anion (cation) is calculated and used as input from the previous atom-relaxation step.

Schematically, the potential for the whole material is constructed from the primitive, fitted, unalloyed potentials as follows:

$$v_{\alpha_u}(q) \to v_{\alpha_m}(q) \to v_{\alpha(i)}(q) \to v_{\alpha(i)}(q,\epsilon) \to v_i(q) \to v_i(r) \to V_{ps}(r).$$

The empirical pseudopotentials for the AlGaAs semiconductor studied in Section 3 are described in detail in [7] (the complete EPM parameters are given in Table 1 therein). These pseudopotentials have been used to study such systems as quantum films, wires, and dots in [11,12] and the inverse band structure problem using simulated annealing in [1]. One takes into account accuracy, predictability, transferability and low kinetic energy cut-off for efficiency when a set of empirical pseudopotential parameters are generated [6–9]. The lattice constants of AlAs and GaAs are within 0.1% of each other; thus it is assumed that the AlGaAs alloy system with their atoms at the ideal zincblende lattice sites are strain free. Therefore, both γ_α and $\mathrm{Tr}(\epsilon)$ described above are set to zero. We note that it is possible to include spin–orbit interaction and additional non-local pseudopotentials in this empirical pseudopotential model [6–9]. However, for the AlGaAs system studied below the effects of the spin–orbit interaction are not significant so we will not discuss these matters further here.

The fourth and final step in our forward solver is the actual solution of the Schrödinger equation written as [6]

$$H\psi_i = \left\{ -\frac{1}{2}\nabla^2 + \sum_n \sum_j v_{\alpha(j)}(\mathbf{r} - \mathbf{R}_n - \mathbf{d}_j) \right\} \psi_i = \epsilon_i \psi_i, \tag{7}$$

where as above the sum runs over all the atoms at the lattice sites $\mathbf{d}_j$ within the supercell whose origin is $\mathbf{R}_n$ and all the possible infinite repetitions of the supercell. The atomic screened pseudopotentials $v_{\alpha(j)}$ are calculated from the previous step. The wavefunctions are expanded in a planewave basis set. Since we are interested in the states around the bandgap, we need not calculate all the eigenvalues. Instead, we use the so-called "folded spectrum" method (FSM). In the FSM described in [13], we replace H by $H' = (H - \epsilon_{\mathrm{ref}})^2$, where ϵ_{ref} is an arbitrary reference energy such as the Fermi energy of the system. The matrix H' has its smallest eigenvalues near ϵ_{ref}, so any eigensolver which finds the eigenvalues of H' in order of magnitude will find the desired band edge eigenvalues first. Parallel implementation of the folded spectrum method (PESCAN) is described in [14]. In this implementation the eigenvalues are found by conjugate gradient minimization of the Rayleigh Quotient $\langle\psi|H'|\psi\rangle/\langle\psi|\psi\rangle$.

To summarize, our forward solver works as follows: We first construct a σ specifying the identity and ideal location of every atom in our system. Then we allow the atoms to relax to their equilibrium positions. Next, we construct the atomic potential by overlapping pregenerated empirical pseudopotentials for each atomic environment. Finally, we find the smallest several eigenvalues of the transformed matrix $H' = (H - \epsilon_{\mathrm{ref}})^2$. This gives us the eigenvalues near the band edges from which we calculate the bandgap and related electronic properties.

These four steps are computed for each iteration of the optimization process for the inverse method described below. Thus we would like our forward solver to be as efficient as possible. We now turn to a description of our genetic algorithm based optimization method.

2.2. Inverse band structure method

Mathematically, the problem of finding the best atomic configuration with respect to a given property is a global optimization problem. As in Section 2.1 let σ be an atomic configuration; σ completely specifies the location and identity of every atom in the system. Now let $P(\sigma)$ be the property of the

material we wish to optimize. For example, throughout this paper P is the bandgap of the material. We suppose without loss of generality that we want to maximize P (the other obvious possibilities, minimize P or achieve a target value of P, are easily converted to this form). Then our problem can be concisely written

$$\max_{\sigma} P(\sigma).$$

However, this description of the mathematical problem is incomplete. We have used σ in both mathematical terms as variables over which we maximize P and physical terms as atomic configurations. However, these are not necessarily the same, and to be clear we must distinguish between them. So let ξ be the independent variables in the problem, i.e., the variables whose best values we want to choose. For us ξ is a string of numbers manipulated by the genetic algorithm. When we call the forward solver, these strings are *mapped* to atomic configurations according to the geometry chosen for the evaluation (lattice type and supercell size). There is no reason the variables over which we optimize need to be in a *one to one* correspondence with atomic configurations. It is easy to imagine the strings manipulated by the genetic algorithm representing a material indirectly, for example, as a sequence of compositions of layers of an alloy superlattice rather than an atom by atom enumeration of the entire superlattice. See [3] for an interesting such application. Representing the material indirectly in this way changes the space of variables being searched. We can tune the size of the search space (thus the ability of our algorithm to actually find the optimum) by representing the material in different ways. So in fact σ is a function of ξ, and the optimization problem is more correctly written

$$\max_{\xi} P(\sigma(\xi)).$$

Note, however, that in this paper we will always optimize over the full atomic representation of the material; ξ does have a one to one correspondence with the atomic configuration, and $\sigma \equiv \xi$. Henceforth we need not refer to ξ, but it is important to keep in mind the dual role of σ as variables over which we optimize and representation of a physical system.

With $\xi = \sigma$, we occupy an extreme end of a spectrum of possible approaches to the automated design of materials in that both the forward and inverse solvers are working at the *atomistic* level. From the physical standpoint this choice provides maximum computational accuracy and configurational potential. But it is clear that this choice makes our optimization problem a difficult one. We are looking at the $Al_xGa_{1-x}As$ alloys. For a total number of atoms N, the composition x determines the numbers N_{Al}, N_{Ga}, and N_{As} of the constituent atoms. The problem amounts to choosing the locations of either the Ga or Al atoms from the $N/2$ possible cation sites each could occupy, since choosing the locations of one determines the locations of the other, and the anion locations are occupied only by As atoms. The number of ways to arrange the N_{Al} Al atoms among $N/2$ sites, for instance, is ("$N/2$ choose N_{Al}")

$$\binom{N/2}{N_{Al}} = \frac{(N/2)!}{N_{Al}!(N/2 - N_{Al})!}.$$

For all but the smallest of systems this is an extremely large number. For example, if $x = 1/2$, and letting $N_c \equiv N/2$, the number of cation sites, we have $N_{Al} = N_c/2$. Using Stirling's approximation to the factorial function, we find that the number of configurations is

$$\binom{N_c}{N_c/2} \sim \sqrt{\frac{2}{\pi}} \frac{2^{N_c}}{\sqrt{N_c}}.$$

Even for a hundred cation system this number is roughly 10^{14} [1]. For a thousand cation system it is almost 10^{300}. For different compositions, the number of configurations does not reduce to such a simple expres-

sion. But we can easily compute it numerically. Fig. 3 shows the number of configurations as a function of composition for 10, 100, and 1000 cation systems.

The actual number of nondegenerate configurations is reduced slightly due to the symmetry of the crystal structure. Specifically, there are $48N_c$ symmetries in our system due to the 48 point group operations of the cubic unit cell and the N_c possible translations between cations. However, in all but the smallest systems the reduction of the number of nondegenerate configurations due to symmetry is insignificant compared to the shear number of total configurations.

The number of configurations scales as a factorial function of the number of constituent atoms. This makes our optimization problem a combinatorial problem (a famous example is the notoriously difficult traveling salesman problem). The fact that the space of variables over which we optimize is *discrete* is a related important characteristic of our problem. There is no such thing as a derivative $\partial P/\partial\sigma$, so calculus based optimization methods are inapplicable.

An approach to solving optimization problems with large configuration spaces is to exploit any decomposability that may exist in the problem. For instance, if we could divide the material into two spatial regions such that the property P could be evaluated by separately evaluating it in the two regions. Then we could solve the optimization problem separately in each part. For some atomic configurations the property of bandgap has this decomposability. The electronic wavefunctions may be localized in one region of the material and the bandgap determined by that region independent of the atomic configuration of the rest of the material. But this is not in general the case. In general, the bandgap of a configuration depends in a sensitive and nonlinear way on the position of each atom in the material.

Our problem is thus a large nonlinear combinatorial global problem. Unfortunately these are among the hardest optimization problems to solve. There is no known method that reliably finds the optimal σ in an efficient manner. We have no choice other than to turn to a heuristic search algorithm. In [1] the simulated annealing method was used to solve the bandgap maximization problem discussed here. We use instead a genetic algorithm (GA). Genetic algorithms are inspired by the process of evolution in nature and originate with the work of Holland and colleagues at the University of Michigan [18]. The basic ingredients of a genetic algorithm are a "population" of potential solutions, a way of testing the "fitness" of any individual solution, and a mechanism for generating new solutions from the existing population based on this fitness. We have written a program, iaga ("inverse method for alloys using genetic algorithm"), that integrates the

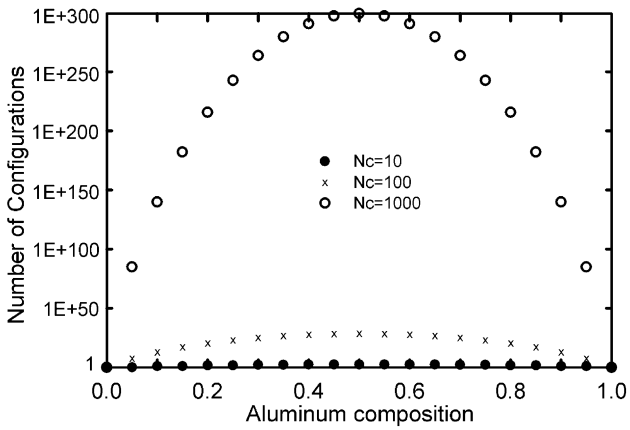

Fig. 3. Number of configurations versus aluminum composition for the pseudobinary substitutional alloy $Al_xGa_{1-x}As$ system represented by supercells with the number of cation sites N_c equal to 10, 100, and 1000.

parallel genetic algorithm package PGApack [19] and the "forward solver" PESCAN described above to perform these steps.

2.2.1. IAGA – inverse method for alloys using genetic algorithm

Here we summarize the operation of iaga; details are discussed below. First, parameters and options for the optimization are read in from an input file. Then an initial set of atomic configurations σ are generated randomly. This set is called the "population", and its members are called "individuals". Since in the genetic algorithm the atomic configurations are represented by strings (i.e., lists) of numbers, the individuals are also called "strings". Now we test the "fitness" of each individual. This occurs in two stages. First, a "functional" value is generated (for example, by calling the physics package to compute, say, the bandgap of the material). From the functional, the fitness of the individual is determined from what the "target" of the simulation is, i.e., whether we are trying to maximize, minimize, or achieve a particular value of the functional. This is the process of computing $P(\sigma)$. When each individual's fitness has been evaluated, we generate a new population by selecting from among the more fit individuals and performing "crossover" (combining two individuals into a single new one) or "mutation" (creating a new individual from a single existing one by randomly changing parts of it). These new individuals replace the least fit of the existing population, giving us the next "generation" of individuals. We repeat this process (thus the generations are sometimes called "iterations") until a stopping criterion (such as reaching a certain number of generations) is reached. This is the process of maximizing $P(\sigma)$. Figs. 4 and 5 illustrate the above procedure. We now discuss some of the details of the method.

Since there are many variants of genetic algorithm, we first describe our choices for some of the most basic features. The two common population models in the field [19,20] are (1) the "steady-state model", in which only a portion of the population is replaced at each iteration, and (2) the "generational model", in which the entire population is replaced at each iteration. In PGAPack the number of individuals replaced each generation is a parameter, so PGAPack supports both models. Most of our runs use the steady state model with the number of individuals replaced per generation approximately 10% of the population. In Section 2.2.2 we examine the choice of this parameter more closely.

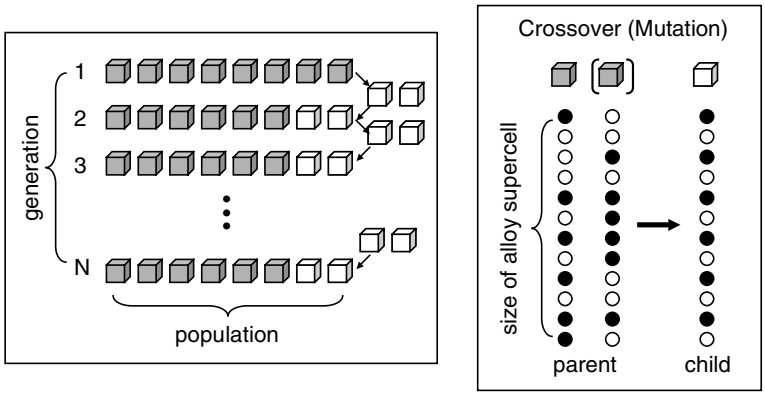

Fig. 4. Features of a genetic algorithm: the population, the generation, and population evolution by mutation and crossover. Left: Each row represents an entire population. The two boxes on the far right represent the newly created individuals, which replace the worst two individuals in the current population to form the new population (the next row). In this example the population size is eight, and the number of individuals replaced per generation is two. Right: The string-of-atomic-numbers representation of a pseudobinary alloy such as $Al_xGa_{1-x}As$ amounts to a binary string. The crossover operation (involving both the left hand strings) creates one new individual from two parents. The mutation operation (involving only one parent) creates a single new individual by swapping atoms of the parent.

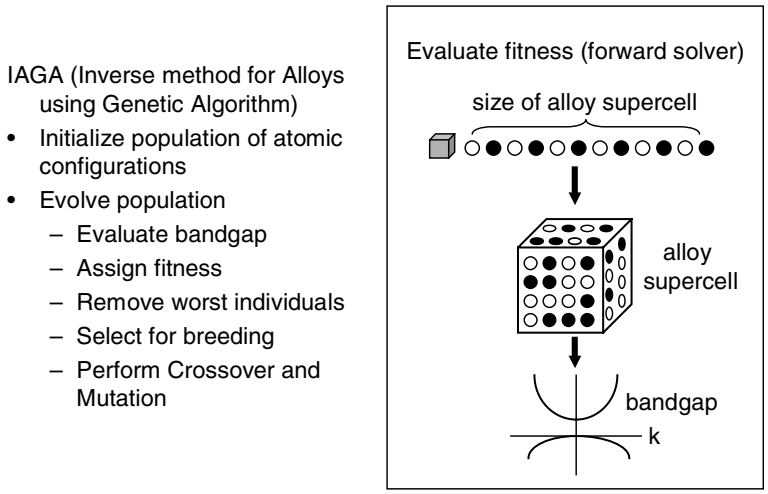

Fig. 5. Schematics of IAGA: A string represents a supercell of atoms. Fitness is evaluated by calling the forward solver.

Our selection method is "binary tournament selection", in which parents are selected by taking the more fit of two randomly selected individuals. Furthermore, we do not implement "survivor selection" [20]; all newly created individuals enter the population, regardless of their fitness.

PGAPack allows individuals created by crossover to also undergo mutation. We have turned this feature off, so individuals are created by *either* mutation or crossover. The percentage of the new population that is created by crossover is determined by the "crossover rate", which for all runs in this paper has been set to 0.8, slightly below the PGAPack default value of 0.85.

We now consider issues related more directly to our particular problem. First, it is well known that the representation of an individual is critical to the success of a genetic algorithm [20,21]. For example, optimization of continuous functions of real variables is made more difficult by representing the real number variables in binary, as changing one bit in a binary representation of a real number does not continuously change the value of the number represented. Our representation of an alloy supercell as a string of atomic numbers amounts, for the case of a pseudobinary alloy such as $Al_xGa_{1-x}As$, to a classic bitstring representation. It might be argued that we have ignored the three-dimensional geometric nature of the objects (namely, semiconductor alloy supercells) we are trying to evolve. While there is some merit in this point of view, it is important to keep in mind the extremely complex relationship between the arrangement of the atoms and the electronic properties we calculate. Specifically, the electronic eigenstates are nonlocal properties involving the positions and identities of *all* the atoms. There is no reason to suppose that "building blocks" of good solutions are necessarily *spatially* related to one another. We point out, then, that in the absence of a priori knowledge of the structure of the fitness function, we have specifically chosen a representation (bitstrings) and form of crossover (uniform crossover) that are *not* spatially biased.

Next, we have introduced the problem without constraints. However, in reality we do wish to constrain the space of configurations we explore. Specifically, we may want the composition of the alloy conserved. The composition is the relative proportion of each of a set of cations or anions making up the alloy. For the pseudobinary alloys considered here it is the single number x in $Al_xGa_{1-x}As$. Given a supercell (and lattice type, which will be FCC throughout this paper), x determines the number of each type of cation atom in the material. Now, the mutation and crossover operations, by randomly changing one atom to another, can change the number of each cation atom in the supercell. In some cases (in fact, in all cases in this paper) we want to prevent this situation. Iaga utilizes two mechanisms to accomplish this goal. The first is to implicitly conserve composition by strongly penalizing individuals whose compositions do not match that

specified in the input file. We add to the functional a term that results in a low fitness for individuals whose composition is not conserved. This is an implicit method because we do not specifically proscribe individuals that violate our initial composition from appearing in the population. We simply force them out by ensuring that they are unfit. The second mechanism is to explicitly ensure every individual conserves composition. We discuss this separately with respect to mutation and crossover.

Iaga (through PGAPack) implements two forms of mutation. In both, each atom of the configuration undergoes mutation with a probability r_m (the "mutation rate"). In the first type, an atom selected for mutation is simply replaced randomly by another atom. In the second, an atom selected for mutation is exchanged with another. Only in the latter case is composition necessarily conserved. This composition-conserving form of crossover has been used in all the computations reported in this paper.

Crossover is the process of combining two individuals (the "parents") to form another (the "child"). In iaga, the form of crossover used is "uniform crossover". For each lattice site in the child, the corresponding atom in the first parent is chosen with probability r_c (the "uniform crossover probability"); otherwise the corresponding atom in the second parent is chosen. Note that we require the parents and child to have the same number of atoms. More importantly, note also that this operation is not guaranteed to conserve composition. In order to explicitly constrain composition, we add a further step to the crossover operation that in a systematic way (that is, in a way that as much as possible results in an individual that really is the result of the crossover of the two parents) guarantees conservation of composition.

We have mentioned the degeneracy in configurations due to symmetry, and noted that the size of the configuration space is not reduced significantly by taking into account symmetry. However, from the standpoint of crossover, it may be useful to take into account the "distance" between two parents; specifically, we have implemented a system that translates one of the parents to the equivalent configuration that has the most in common with the other parent. In this way we maximize the chance of ordered structures being carried forward to the next generation [22].

Both PGAPack and PESCAN are parallel programs. Iaga is thus a "hierarchical parallel" program. The available processors are divided into groups in two ways based on the number of processors per functional evaluation (PESCAN) specified. First, they are divided into a set that will only perform PESCAN calculations and a set that will also interact with PGAPack (the "GA group"). Second, they are divided into "functional groups" that will together perform a particular functional evaluation. Each of the members of the GA group becomes the head node of a functional group. The root node performs all the intergenerational tasks: selection, mutation, and crossover. Fig. 6 illustrates the parallelism employed here.

2.2.2. Genetic algorithm parameter studies

A genetic algorithm is controlled by a multitude of options. The PGAPack User's Guide, for instance, lists 38 different parameters that one can adjust. We will not attempt to analyze them here. The optimization of genetic algorithm parameters is a field unto itself. Here we will concentrate on five parameters in particular that have a large impact on the effectiveness of the algorithm. They are:

- population size;
- total number of generations;
- number of new individuals created at each generation;
- uniform crossover probability;
- mutation rate.

The optimization of these parameters involves inevitable tradeoffs. Given a fixed computational budget, that is, an amount of time we are willing to wait for our computation to finish, we can use a large population for a small number of iterations, or a small population for a large number of iterations; or we can maximize both by replacing only a small number of individuals in each generation. The choice of uniform

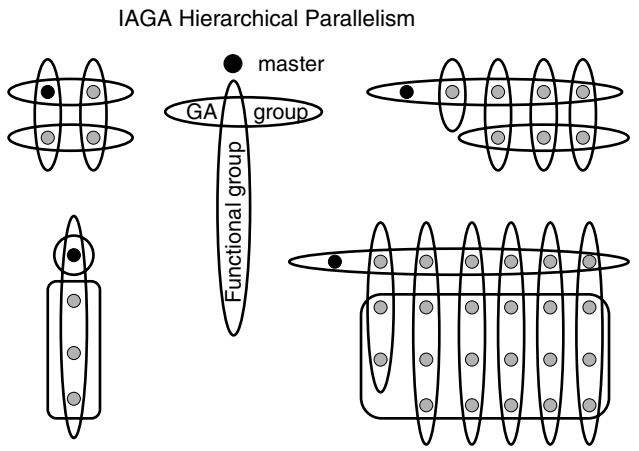

Fig. 6. Hierarchical parallelism in iaga. Shown are the processor groupings for four different (number of processor, number of functional groups) combinations, (clockwise from upper left) (4, 2), (8, 4), (24, 6), and (4, 1). Each circle represents one processor. The black circle is the master processor, which manages all intergenerational tasks (e.g., selection, mutation, and crossover). Each column represents a group of processors that evaluates the functional of one individual in parallel (a "functional group"). Two processors per functional group are indicated in top two groupings and four in bottom two groupings. The top row of processors is the "GA group"; these are the the head nodes for their respective functional groups (the columns). The parallel genetic algorithm distributes evaluation of the population to the members of the GA group, each of which then distributes the evaluation of a single functional to the functional group for which it is the head node. Operations within one functional group are independent of and thus performed in parallel with operations within another functional group. Note that if there are more than two functional groups, the master processor is not part of a functional group and does not participate in electronic structure evaluations but only manages the genetic algorithm.

crossover probability and mutation rate is also a tradeoff. For low uniform crossover probability and mutation rate, we see rapid convergence of the bulk of the population to a similar set of solutions, but often they are nonoptimal (they are "local extrema"); for high rates, the population will retain a larger variety of potential solutions (diversity), but it will have trouble converging to an optimal one. Though in certain special cases the optimal value of some of these parameters can be established [23], the optimal choice of GA parameters is, in general, an unsolved mathematical problem. Here we discuss some simple numerical parameter searches we have performed in the context of an Ising model test case. The test case itself is described in Section 2.2.3.

Fig. 7 shows the typical progress toward convergence of an entire population. An optimization by a genetic algorithm typically has two phases, an "exploratory" or "global" phase during the early generations followed by a "refinement" or "local" phase as the system nears convergence [24]. During the exploratory phase, the population is diverse, and rapid improvement of the solution is typical. In the refinement phase, the system is nearing convergence on one individual or a set of similar individuals, and improvement of the solution slows. By changing the mutation rate and uniform crossover probability, one can tune the length of the exploratory phase [22].

The genetic algorithm is fundamentally stochastic. Each run is different unless we explicitly set up our random number generator to repeat a previous run. As is the case with all heuristic search algorithms, there is no guarantee of finding the global optimum. But through better parameter selection we can at least hope to maximize the *probability* of finding a solution acceptably close to the global optimum in an acceptable amount of time. To this end, our studies allow us to make several remarks:

(1) We are best served by running an optimization from the beginning several times rather than putting all our computational resources into a single long optimization run. Fig. 8 illustrates this point.

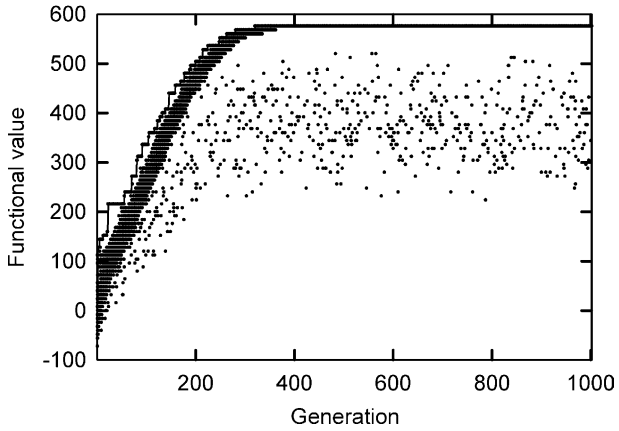

Fig. 7. Functional value versus GA generation for optimization with simple cubic Ising Hamiltonian for 125 cation atoms. Each member of the population for each generation is represented by a point. The curve shows typical progress toward convergence of the entire population. Since we do not use "survivor selection" (all newly created individuals enter the population, regardless of fitness), the plot shows many individuals far below the maximum, even near the end of the run.

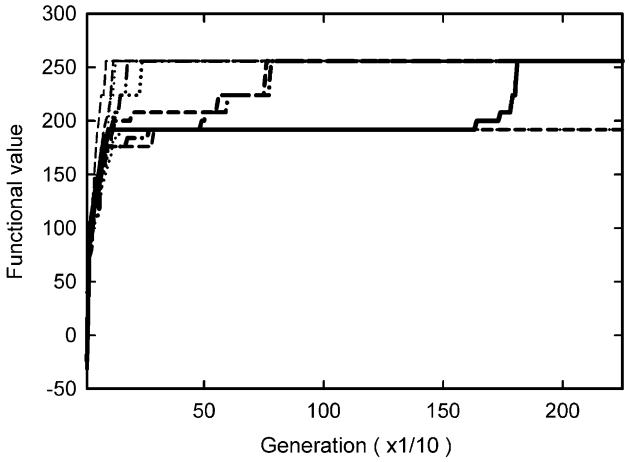

Fig. 8. The paths to convergence of several repetitions of the same optimization. Each run has a different random initial population, but all other parameters are the same for each run. The convergence of the algorithm does not imply the discovery of the global optimum. Note the sudden jump of a run near generation 1600 after seeming convergence.

(2) Population sizing for genetic algorithms is a well studied issue. Most theoretical treatments derive an exponential dependence on system size [25] for their chosen, analyzable system although under certain assumptions this can be reduced to a linear dependence (see Section E1.1 in [21]). Our purely empirical study suggests that in our case the optimum population size is a linear function of the system size. Fig. 9 shows the optimum population size computed for a series of system sizes. The optimum population was defined as the smallest population that achieved the best average value over 10 runs. The runs were allowed to run until convergence (no change in best fitness value for 100 generations, or 99% of the population with the same fitness value) *or* a maximum of 100,000 evaluations. The smaller systems all converged before the evaluation limit, so for these, the time taken to reach these values was a factor in determining which population size was deemed best. However, for the larger systems, the

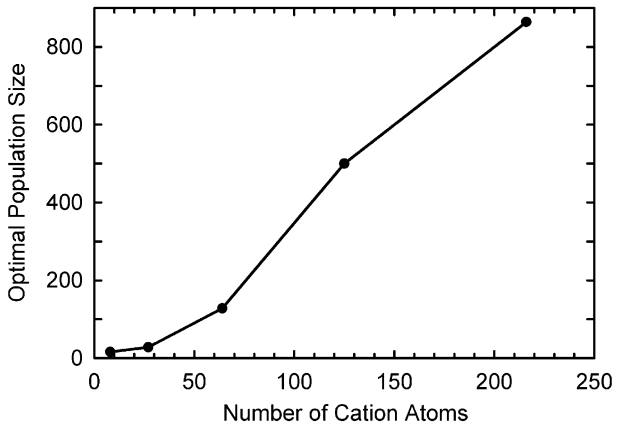

Fig. 9. Optimal population size versus system size for maximization of an Ising model test case.

evaluation limit was encountered, so these best populations are not fully converged (i.e. the average value for even the best population size is not the global optimum). The study suggests that the optimum population size thus defined is roughly four to five times the number of atoms. In general the population should be neither extremely small nor extremely large. Small populations have insufficient diversity, while extremely large populations have such a high proportion of bad solutions that the algorithm fails to pick out the better ones.

(3) For a fixed computational budget (number of allowed functional evaluations), the best functional value achieved shows an interesting dependence on the number of individuals replaced per generation n_r. In Fig. 10 we show the functional value versus population size for a range of population sizes for an Ising model on a 6 by 6 by 6 simple cubic supercell lattice with 216 cations and a maximum total functional evaluations of 65,536. The plot reveals that for the optimum population of 1024, a wide range of n_r works well, but that for smaller populations, performance improves with increasing n_r, while for larger populations, performance is generally bad, and degrades with increasing n_r. The explanation for these results is that for small populations, the system is struggling to maintain diversity, so the increased diversity provided by increased n_r helps. For large populations, the system rapidly reaches the limit of allowed functional evaluations. For the largest population size, there is only one iteration, and the search reduces to a random sampling. In this case the fewer individuals we replace, the more iterations we can perform, and the more able the system is to refine the high quality solutions in the population. The optimum population represents a balance between the forces of global exploration (requiring diversity) and local refinement (requiring many iterations).

(4) In the genetic algorithms literature it is widely recommended [19] and in some cases proved [23] that for a system of size N, a mutation rate r_m of roughly $\frac{1}{N}$ should be used. We have performed two tests of mutation rate r_m and uniform crossover probability r_c. First, typical in a real application, we imposed a fixed computational budget of 30,100 functional evaluations. Here we find that as expected, lower mutation rates (roughly $1/N$) find better solutions faster. Fig. 11 shows the results of this test. In our second test, we let the system run "until convergence" for a variety of r_m and r_c. Our measure of full convergence was that the best fitness achieved had not changed for 100 generations or that the fitness of 99% of the population was the same. Under these rather demanding stopping criteria the final functional value was not found to be sensitive to uniform crossover probability and mutation rate. Fig. 12 shows the results of varying these rates. Neither the value of the functional nor the time it took to reach it varied appreciably. Despite being unrealistic in a practical setting, this test is important,

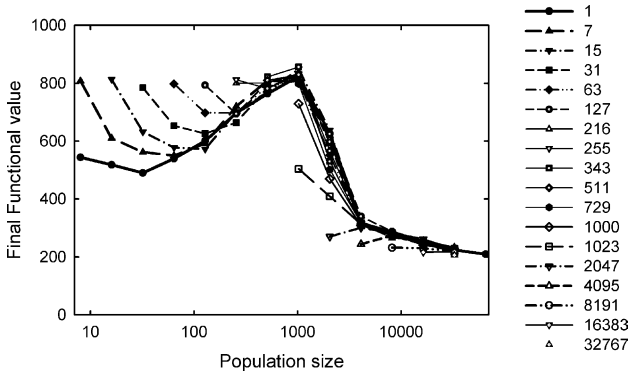

Fig. 10. Final functional value versus population size for different choices of number of replacements n_r (legends on the right) arrived after optimizations for a 6 by 6 by 6 simple cubic supercell with a fixed computing budget of 65,536 evaluations. The poor performance for $n_r = 1023$ relative to $n_r = 1000$ for population size 1024 is most likely a statistical aberration despite our averaging over 5 runs.

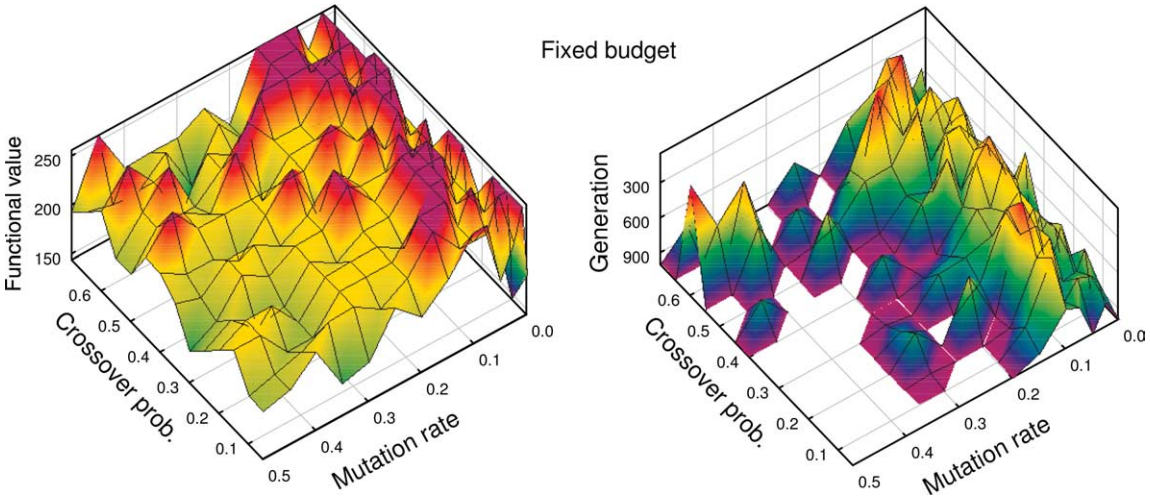

Fig. 11. Parameter study for Ising model test case, 4 by 4 by 4 supercell (64 atoms), for a fixed limit of 30,100 functional evaluations. Left: Value achieved at convergence for a grid of mutation rate and uniform crossover probability ("Crossover prob" in the figure). Shown is the best value achieved over 4 runs. The "flat top" is the optimum functional value; runs were stopped if and when they achieved this value. This figure illustrates the fact that if we want to converge faster, lower mutation rates are advantageous. The population size was 100; the number replaced per generation was 30. Right: Time to convergence (in units of generations). The "flat bottom" at higher mutation rates, exhausted the computing limit we have imposed, 1000 generations.

since it is hard to distinguish convergence to the global optimum from long plateaus in the functional versus generation curve. Note, however, that we are not making the much stronger claim that mutation rate and uniform crossover probability have no effect; this claim is already refuted by our first test. We claim only that at least for this particular problem (and we make no assertion of universality), if one allows the system to fully converge, the final functional values do not vary appreciably with mutation rate and uniform crossover probability. Given enough time, the system finds its way out of all local optima. In either case the length of the exploratory phase of the optimizations does depend on these parameters; a lower mutation rate causes the system to *begin* to converge sooner. However,

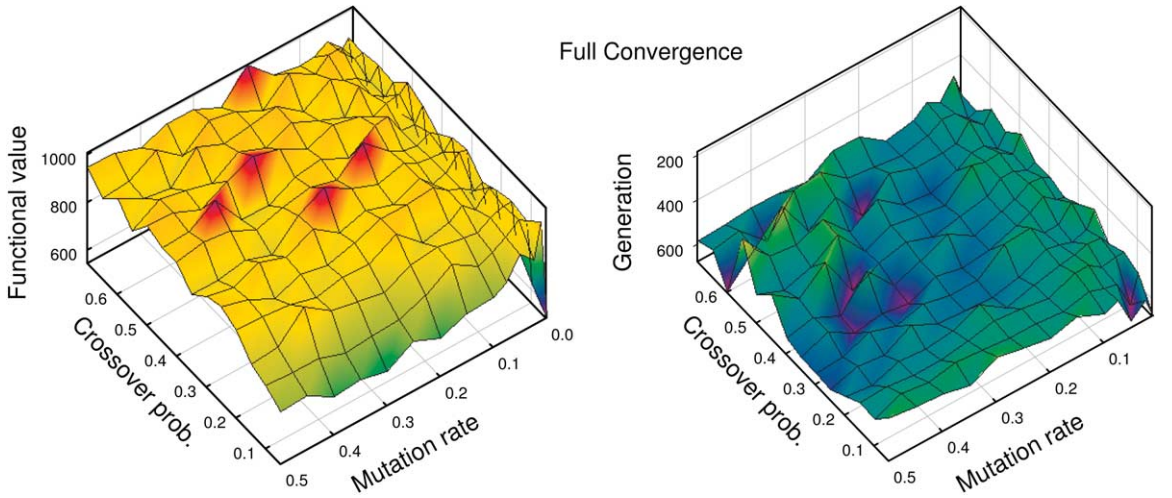

Fig. 12. Parameter study for Ising model test case, 6 by 6 by 6 simple cubic supercell (216 cation atoms). Left: Value achieved at convergence for a grid of mutation rate and uniform crossover probability ("Crossover prob" in the figure). Shown is the best value achieved over 10 runs. From this figure we infer that at least for the Ising model, and as long as we are willing to let the system run to convergence, there is not an extremely sensitive dependence of the algorithm on these parameters. The population size was 864; the number replaced per generation was 128. Right: Time to convergence (in units of generations).

we just do not find the final convergence as defined above to be dependent on these rates. We do not want fast convergence if that convergence is only to a local optimum. Further tests reveal that in some cases mutation and crossover functions are interchangeable. If both rates are too low, the system converges prematurely. But if both rates are too high, the system never finds high quality solutions. We need at least one of these rates to be large in order to inject diversity into the population and at least one to be small in order to refine reasonably good solutions into the optimal or near optimal solutions we desire. Our experiments suggest either parameter can perform either function.

(5) The choice of number of generations to run is also a matter of computational budget. If we have chosen the other parameters as best we can, we can let the optimization run for as long as we are willing to wait, look at the data to see if the results are acceptable, then restart the algorithm from where it stopped. Iaga has a built in restart mechanism which saves and restores the random number state of the system, so stopping the run and then restarting it again has no effect on the data generated.

To summarize, then, we adopt the following simple strategy: For system size N, choose population size roughly equal to $5N$. Then let the number replaced per generation be 10% of the population size. Choose reasonable small values such as $1/N$ for mutation rate and 0.25 for uniform crossover probability. Run the optimization until the number of functional evaluations is roughly 10 times the population size. Restart and continue if necessary. If the onset of convergence is detected (flattening of the functional value versus generation curve) before a high quality solution has been found, repeat the entire run.

2.2.3. Ising model test case

The forward band structure problem is computationally expensive, since the thousand or so functional evaluations necessary to perform a bandgap optimization for even a small system may take several hours. For this reason we have implemented an Ising model test case. The same optimization can now be performed in only a few seconds. The functional value is computed from the model Hamiltonian

$$H = -\sum_i \sum_j^{nn} J_{ij}\sigma_i\sigma_j,$$

where the sum over i is over all lattice sites, and the sum over j (for each i) is over the nearest neighbors of lattice site i. For this model the "nearest neighbors" are those of the same ion type (i.e., cation–cation and anion–anion). J_{ij} is a coupling constant. Here we take a constant interaction $J_{ij} = J$. The "spins" σ_i are defined by what atom occupies site i. For our simple test case, σ_i is 1 if site i contains aluminum, -1 if it contains gallium. The arsenic atoms are assigned zero spin. It is important to note that this test case implements a different functional than that of bandgap, which implies that the optimization problem may be quite different in the two cases. Therefore, we should be wary of putting too much emphasis on it. We use it only for debugging the often complicated series of steps required to manipulate the atomic configurations and for getting rough ideas of the best values of some of the key parameters as described above.

The Ising model test case is implemented for simple cubic (SC) and face centered cubic (FCC) lattices. All the test results reported above are for the simple cubic case, in which there is a cation at the corner of each 1 by 1 by 1 cube of the supercell. The anion, located at the center of each of these cubes, has no effect on the functional value. This of course is not the case for the bandgap functional discussed in Section 3.

Before attempting to run the full electronic structure optimization described in Section 3 we have maximized and minimized the energy of the Ising system as a function of given fixed composition. The optimization is run in the same way for the bandgap energy optimization in Section 3.2. Fig. 13 shows the functional (energy) optimized for the Ising Hamiltonian. Fig. 14 shows the optimized configurations at 50% composition.

3. Bandgap energy optimization of AlGaAs alloys

To explore our method on a substantive yet relatively well studied problem, we have used iaga to examine the bandgap of $Al_xGa_{1-x}As$ alloys. First, primarily to understand the typical dynamics of the optimization process, we discuss the optimization of bandgap for a few supercells of particular single composition specified by x. Next we look at bandgap versus composition, which involves solving both the maximum and minimum bandgap problem for a range of compositions. Some preliminary remarks frame the discussion.

For two reasons, we fix the composition x for each single optimization run. First, real materials need compositions which are lattice matched to within a few percent to the substrate on which they are grown,

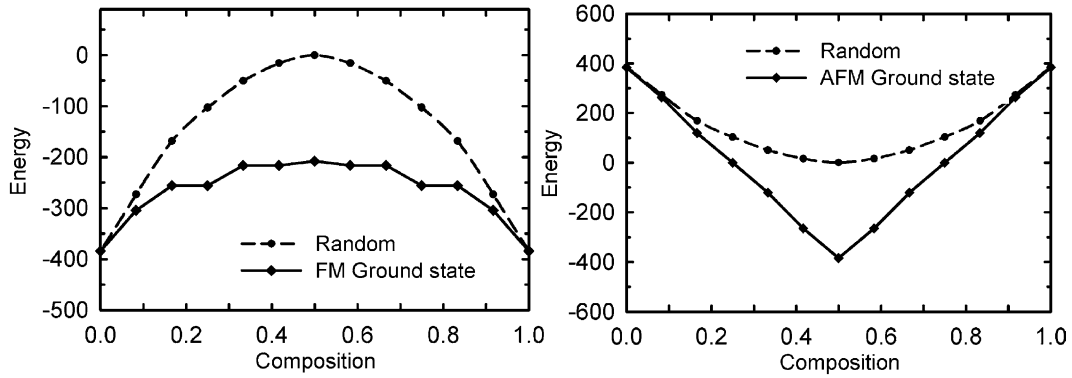

Fig. 13. Composition dependence of ferromagnetic ($J = 1$, left) and antiferromagnetic ($J = -1$, right) energies for the Ising test case on a simple cubic lattice with 64 cation atoms.

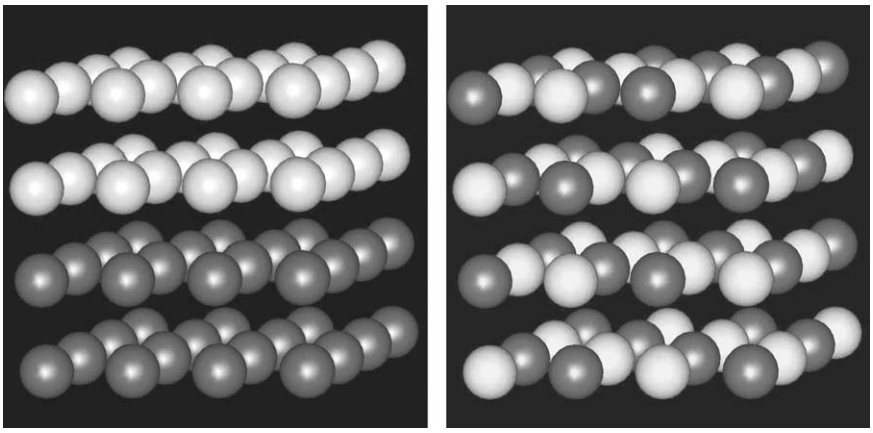

Fig. 14. Configurations of ferromagnetic ($J = 1$, left) and antiferromagnetic ($J = -1$, right) ground states for the Ising model test case on a simple cubic lattice with 64 cation atoms at 50% composition.

so it is not reasonable that just any composition with the desired bandgap is acceptable. Though this is not an issue for AlGaAs in particular (because AlAs and GaAs have the same lattice constant to within 0.1%), it is an issue for most other alloys, so it provides a reason to focus on optimizations where composition is fixed. Second and more importantly from the standpoint of optimization, allowing the composition to change has more effect on bandgap than allowing the arrangement of atoms of a particular composition to change does. Thus allowing the composition to change during, say, a maximization, the composition would shift to the larger bandgap material, AlAs. We would find only that one or the other pure material (AlAs or GaAs) had the higher bandgap.

Note, however, that *if* the optimum bandgap (over all x) occurs for $0 < x < 1$, then iaga could solve the nontrivial problem of finding what the optimum composition is. Relaxing the composition constraint increases the size of the search space by roughly one order of magnitude, which is not significant compared to the already large (e.g., 2^{29}) size of the space for a fixed composition. Preliminary tests with our Ising model test case indicate that the system does hone in quickly on the optimal composition.

An important detail of the forward solver is that the eigenvalues found are those at a particular k point in the Brillouin zone. That is, writing the wavefunction ψ in the Bloch form $\psi(x) = u_k(x)\, e^{\mathrm{ikx}}$, we have

$$(H - \epsilon)\psi = (H - \epsilon)(u_k \mathrm{e}^{\mathrm{i}kx}) = \mathrm{e}^{\mathrm{i}kx}(H_k - \epsilon_k)u_k,$$

where [26]

$$H_k = \frac{\hbar^2}{2m}\left(\frac{1}{\mathrm{i}}\nabla + k\right)^2 + V(r).$$

This allows us to compute the band structure by finding the ϵ_k for a range of k points. To do so, one expresses the wave function in a plane wave basis set, transform the Hamiltonian to a momentum space representation, and solves the resulting matrix equation as described in Section 2.1. In this optimization, all the computations are done at the Γ point $k = 0$.

Due to Brillouin zone "folding", the particular supercell we use for a run is of critical importance. Folding of the Brillouin zone happens when we repeat a primitive cell in one or more directions. A doubling in physical space, for instance, halves the size of the Brillouin zone. The k points in the second half of the original Brillouin zone are "folded" back into the first half toward the Γ point $k = 0$. In terms of Bloch

functions, we have rewritten a wave function which was in terms of a k value outside the new first Brillouin zone in terms of one which *is* in this zone: If $k' = k + G$ where G is a reciprocal lattice vector and k and k' are inside and outside of the first Brillouin zone, respectively, then

$$\psi(x) = u_{k'}e^{ik'x} = (u_{k'}e^{iGx})e^{ikx} \equiv u_k e^{ikx}.$$

So, for example, if we define a supercell that is twice the size of the primitive cell in all directions, then the L point $k = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})\frac{2\pi}{a}$ of the original cell of lattice constant a is folded back onto the Γ point $k = (0,0,0)$.

Zone folding enters as follows. The method we use to find the bandgap is rather simple. For a known reference energy below (above) the conduction (valence) band edge, we find the closest state above (below) it. The bandgap is the difference between these energies. If we compute in a primitive cell, where no states from other k points in the Brillouin zone are folded to Γ, all the bandgaps we find are direct bandgaps at Γ. If, however, the supercell is actually not primitive, then the states from other points in the Brillouin zone that are folded to Γ are also found when we compute the band edges "at Γ", so the bandgaps we find by computing even at just one k point Γ may be indirect. In the case of a primitive supercell that forms a superlattice, i.e., a layered material, certain layerings allow the FCC primitive zone-edge k-points to fold to Γ. In that case, the bandgaps that we report may be "pseudodirect", originating from the zone-folded level. With respect to the actual supercell used, they *are* direct bandgaps. But, considering the nature of the wave functions and the underlying FCC primitive cell, they may be actually indirect in optical character.

In this optimization we make no attempt to distinguish direct from pseudodirect bandgaps, but we mention here that it is possible to do so. Our solver, in addition to the energy values, can also compute the corresponding wave function. By comparing the wave functions to known wave functions of, say, Γ or L or K character, we can distinguish them. This is known as the method of majority representation [15]. Alternatively, we can compute the optical transition probability, which is related to the moment $\langle\psi|\mathbf{p}|\psi\rangle$, which can be computed from the wave functions. This information could be incorporated into our optimization implicitly by penalizing those configurations whose bandgaps were not of the desired (e.g., direct) character.

In this work we are also not considering energetics, thus not considering the relative stability of the configurations we discover. The reasons are twofold. First, our forward solver is efficient because it finds the bandgap by calculating only two energy levels. The hundred or thousand levels necessary in a total energy calculation would be impractical in the context of optimization. Second, modern experimental techniques involving metastable phases have progressed to the point where many less energetically favorable materials *can* be grown, so it is increasingly possible to pursue optimization of electronic properties separately from minimization of total energy.

We now turn to our two optimization studies, first of single bandgap optimizations, then, building on that, an examination of possible bandgaps as a function of composition.

3.1. Dynamics of bandgap energy optimization of AlGaAs alloys

We present three examples of bandgap optimization. The first is run in a 192 atom cell and failed to reached the global optimum after 9000 functional evaluations, but it allows us to point out some features of the typical trend of a GA simulation. The second and third were run in 64 atom cells and reached the global optimum. Comparing these to each other and the first run involves aspects of both the optimizer and the electronic properties of the AlGaAs semiconductor alloy.

First, Fig. 15 shows the bandgap versus genetic algorithm generation for a typical iaga run. The particular problem is the bandgap maximization of an $Al_{0.25}Ga_{0.75}As$ alloy. The supercell is 4 by 3 by 2, making this a 192 atom system (with 96 cations). After 600 generations, the maximum bandgap discovered is 1.855 eV. Shown are the bandgaps for every member of every population.

As in Section 2.2 (see Fig. 7) the plot shows two fairly distinct regimes. For the first 100 or so generations we see a wide variation in the bandgaps at a particular generation, and rapid improvement in the bandgap

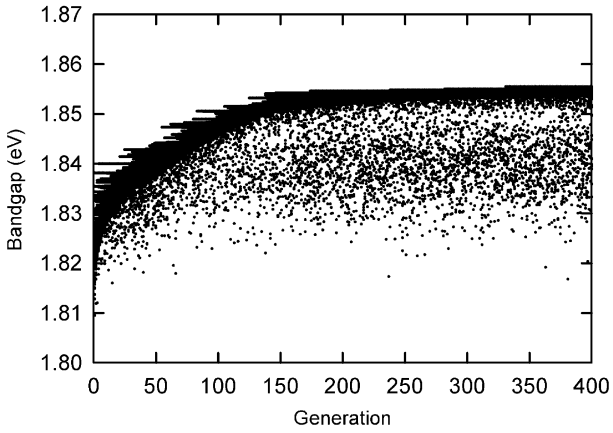

Fig. 15. Bandgap vs generation for maximum bandgap optimization of $Al_{0.25}Ga_{0.75}As$ in a 4 by 3 by 2 supercell. This is the general shape of genetic algorithm optimization curves. They change rapidly and then slow down and reach a plateau. The best fit value at the onset of the plateau at 120 iterations is 1.853. A better convergence criteria could have stopped the optimization run here to save iterations. Note that this run did not find the global maximum. A better strategy would be to stop at the onset of convergence and start again with a new random population.

value. After that, the curve flattens, and the bandgap values are mostly close to each other. The "dots" of lower bandgap in this second phase are newly created members of the population whose bandgaps are not optimal. They will be among the first to be replaced in the next generation. As noted in Section 2.2, the transition from an initial exploratory phase to a longer converging phase is typical of almost all global optimization methods [24,27]. In the initial "global" or "exploratory" phase, the population contains a wide variety of potential solutions; the optimizer is looking for which of the areas of the search space represented by these individuals contains the global optimum. In the second, the "refinement" or "local" phase, the system has already converged on a single region of the search space and is finding solutions which are small variations within it [22].

We observe that the progress of a GA run almost invariably has such a transition to a flatter, converging phase, and that the value of solutions at the beginning of this phase are usually within five percent of the true optimum. With the goal of reducing the number of evaluations our forward solver must perform, we are led to search for a stopping criteria based on detection of this transition and extraction of a desired solution from the population at this point. Typically the desired solutions are ordered structures; these are often the extremes observed experimentally, and these are what have been found by previous inverse band structure calculations [1]. Investigations are ongoing into several approaches. For detection, we can do some simple statistical analysis of the population such as computing the variation of the fitness values and stopping when it falls below a certain threshold. Or we can actually look at the configurations themselves to see how similar they are. Problems here include taking into account symmetries in order to measure similarity of configurations correctly and the fact that though these two regimes are the norm it is possible for the curve to repeatedly flatten, then rise again, then flatten (see Fig. 8). More problematic has been the extraction of meaningful (i.e., ordered) solutions from partially converged populations. We are working on various tools to detect structure, periodicity, and other nascent signs of order [28]. However, the subtle differences between randomness and two or more competing partial orderings are difficult to analyze.

In Fig. 16, we show the progress of an $Al_{0.25}Ga_{0.75}As$ bandgap maximization in a 4 by 2 by 1 (64 atom, 32 cation) supercell. There are over 10 million possible configurations (this run of iaga is not taking into account symmetry), yet the optimum is found in 31 generations, which in this case was about 650 bandgap

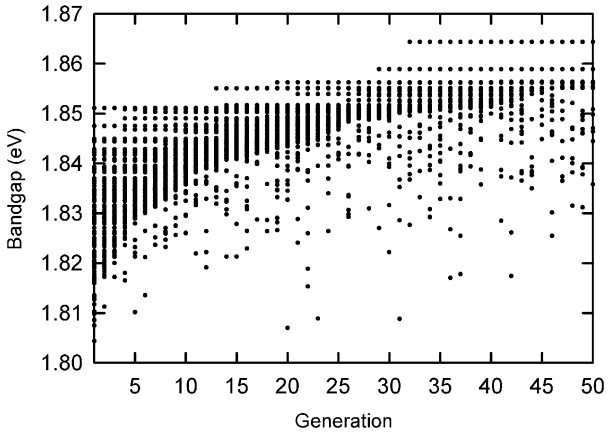

Fig. 16. Bandgap vs generation for maximum bandgap optimization of $Al_{0.25}Ga_{0.75}As$ in a 4 by 2 by 1 supercell. This run found the global optimum, an $(AlAs)_1(GaAs)_4(AlAs)_1(GaAs)_2$ superlattice in the $[0\bar{1}2]$ direction. Only 650 bandgap evaluations were needed to find this solution.

evaluations. Fig. 17 shows the results for bandgap minimization of $Al_{0.5}Ga_{0.5}As$ in a 2 by 2 by 2 (also 64 atoms) cell. Both of these plots differ from Fig. 15 in the relatively discrete jumps being made as the solution improves. This is of course due to the difference in supercell sizes. In the big cell, there are many many configurations which have almost the same bandgap, so improvement is almost continuous. There are two reasons for the discrete jumps in the smaller cells. The first reason is purely combinatorial. The configuration space is smaller, and thus a change in one atom changes the configuration, thus the bandgap, more significantly in the smaller cell. The second reason is physical. We see that the effect is more pronounced as we reach the optima because this is the point where the configuration achieves its final ordered state. In the minimization case, in particular, the minimum is a "segregating state" (see below). When the genetic algorithm puts the last atom in place, the resulting ordered structure is *qualitatively* different from the ones which are even only one atom swap away, explaining the jump in one step of 75 meV to reach the final solution. Note also the range of the bandgaps through which the solutions evolve. In the maximization case, the total range (from 1.851 to 1.865 eV) is only 14 meV, whereas in the minimization case, the range is approximately 165 meV. This is explained by the fact that the random alloy bandgaps are actually much closer to the maximum bandgaps than the minimum bandgaps (see Fig. 19).

Having explained much of the regularity of these plots, it is important also to keep in mind that the genetic algorithm is fundamentally stochastic. We can expect a fair amount of *irregularity* when looking at the progress of any one run. For example, the initial population in the maximization case happens to contain a quite high quality solution (1.851 eV), so the shape of the whole curve is flatter than in the minimization case. This is but one example (see, e.g., Fig. 8 for more evidence of variability from run to run).

For the purpose of comparing simulated annealing with our genetic algorithm, we have used iaga to solve two of the $Al_xGa_{1-x}As$ bandgap maximization problems solved by simulated annealing in [1]. First, in Fig. 18 we show the progress of bandgap maximization of $Al_{0.75}Ga_{0.25}As$ in a 64 atom, 1 by 1 by 32 supercell for both the genetic algorithm and simulated annealing. This is a one-dimensional problem. We are searching the space of [0 0 1] oriented superlattices. The maximum bandgap configuration is an $(AlAs)_3(GaAs)_1$ superlattice (see Table 1 in [1]). For this iaga run the number of individuals replaced each generation was 15, so the total number of evaluations required to solve the problem was approximately 3000. Simulated annealing required approximately 4000 evaluations. Our second comparison is the band-

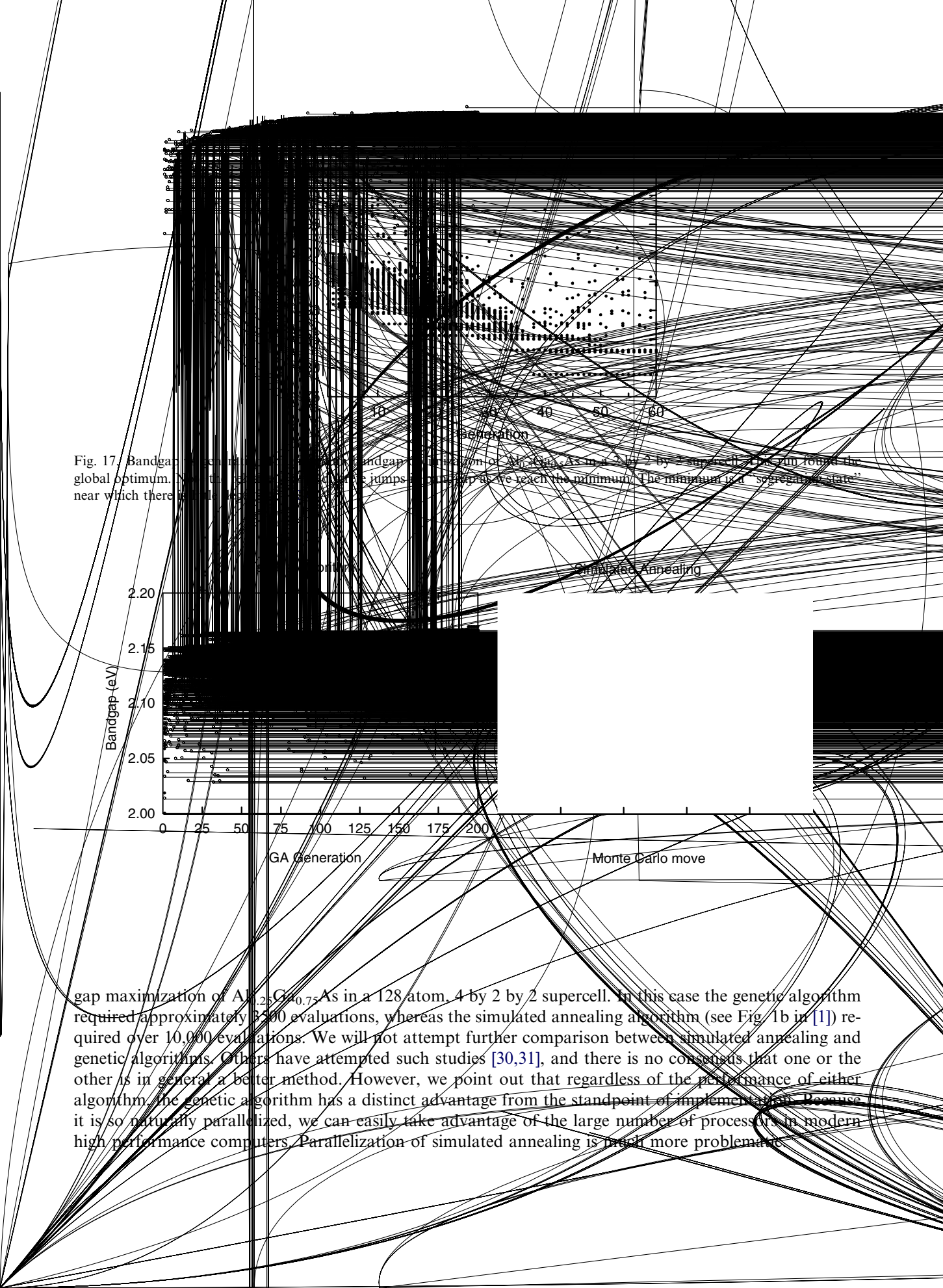

Fig. 17. Bandgap [illegible] bandgap [illegible] atom of $Al_{0.5}Ga_{0.5}As$ in a 2 by 2 by 2 supercell. This run found the global optimum. Note the [illegible] jumps in bandgap as we reach the minimum. The minimum is a "segregated state" near which there [illegible]

gap maximization of $Al_{0.25}Ga_{0.75}As$ in a 128 atom, 4 by 2 by 2 supercell. In this case the genetic algorithm required approximately 3500 evaluations, whereas the simulated annealing algorithm (see Fig. 1b in [1]) required over 10,000 evaluations. We will not attempt further comparison between simulated annealing and genetic algorithms. Others have attempted such studies [30,31], and there is no consensus that one or the other is in general a better method. However, we point out that regardless of the performance of either algorithm, the genetic algorithm has a distinct advantage from the standpoint of implementation. Because it is so naturally parallelized, we can easily take advantage of the large number of processors in modern high performance computers. Parallelization of simulated annealing is much more problematic.

3.2. Bandgap energy versus composition of AlGaAs alloys

We now turn to optimizing bandgaps over a whole range of compositions. Composition dependence of alloy bandgaps is often described by the "bowing curve"

$$E_g(Al_xGa_{1-x}As) = xE_g(AlAs) + (1-x)E_g(GaAs) - bx(1-x). \quad (8)$$

This equation describes the bandgap of an alloy as a composition-weighted average of bandgaps of constituents with a quadratic correction of size determined by the "bowing parameter" b. This is an equation whose origins are empirical, where it is assumed that the alloys in question are *random* alloys of a given composition. In this section we explore the range of possible bandgaps achievable if the configurations are not assumed to be random but are instead specifically optimized to maximize or minimize the bandgap. We will see that the possible points in composition-bandgap space form not a curve but a region.

The following discussion refers repeatedly to Fig. 19. In this figure we show the results of a series of optimizations performed with iaga. Each symbol on the graph represents a separate complete iaga run where we have attempted to either minimize or maximize the bandgap of a particular composition. Recall that in each run, the composition is explicitly conserved.

The dashed line represents the bandgap of random alloys. It is calculated by averaging over bandgaps of a few randomly generated alloy configurations for each composition. The supercells used for these particular calculations were 12 by 12 by 12, with a total of over 13,000 atoms, so these are fairly good representations of truly random alloys.

The shaded regions are the available bandgaps of $Al_xGa_{1-x}As$ alloys achievable by different atomic configurations. The bandgap of random $Al_xGa_{1-x}As$ alloys changes from direct to indirect below 0.55 Ga ($x = 0.45$) composition. The bending in the maximum bandgaps indicates the direct to indirect bandgap transition. There is no transition in the minimum bandgaps, since due to the folding of the L point to Γ discussed above, all of these bandgaps are either indirect or pseudodirect, derived from the zincblende L state.

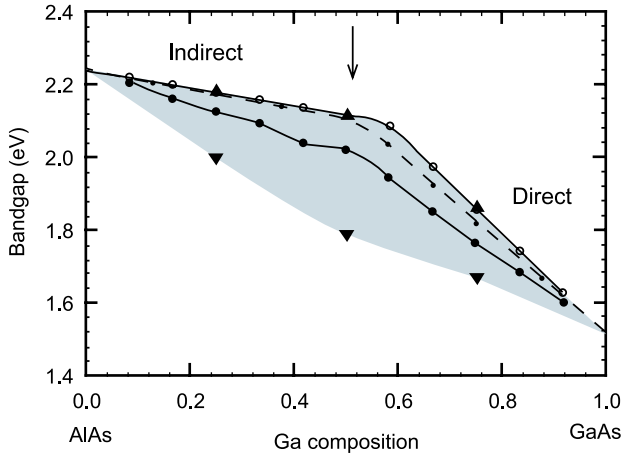

Fig. 19. Bandgap of AlGaAs alloys. Dashed line: Bandgap of random alloys. Up-triangles: Maximum bandgaps. Down-triangles: Minimum bandgaps. The arrow indicates the direct to indirect bandgap transition composition for random alloys. The shaded regions are the available bandgaps of $Al_xGa_{1-x}As$ alloys achievable by different atomic configurations. The solid lines with circles are optimization run with 4 by 3 by 2 cells which did not reach the optima (see text).

Table 1 summarizes some of the optimal maximum and minimum bandgap configurations obtained with the iaga method and discussed below. The maximum configurations are the same as the ones found in [1]. Figs. 20 and 21 show the corresponding atomic configurations.

The choice of supercell is critical and problematic. First, since we are interested in finding direct *or* indirect bandgaps with a single bandgap computation only at Γ, the relevant k points must be folded to Γ as discussed above. Second, the supercell must be an integral multiple of the primitive cell of the optimum structure. For example, if the optimum structure has a 2 by 2 by 2 primitive cell, we will not be able to

Table 1
Optimal bandgap configurations of $Al_xGa_{1-x}As$ alloys

Composition x	Bandgap (eV)	SL direction	Superlattice layer sequence
Maximum bandgap configuration			
0.25	1.86	$[0\,\bar{1}\,2]$	$(AlAs)_1(GaAs)_4(AlAs)_1(GaAs)_2$
0.50	2.12	$[0\,\bar{1}\,2]$	$(AlAs)_2(GaAs)_2$
0.75	2.18	$[0\,\bar{1}\,2]$	$(AlAs)_3(GaAs)_1$
Minimum bandgap configuration			
0.25	1.67		"X" Cmmm structure
0.50	1.79	$[1\,\bar{1}\,1]$	$(AlAs)_1(GaAs)_1$
0.75	2.00		"Luzonite" Cu_3AsS_4-type

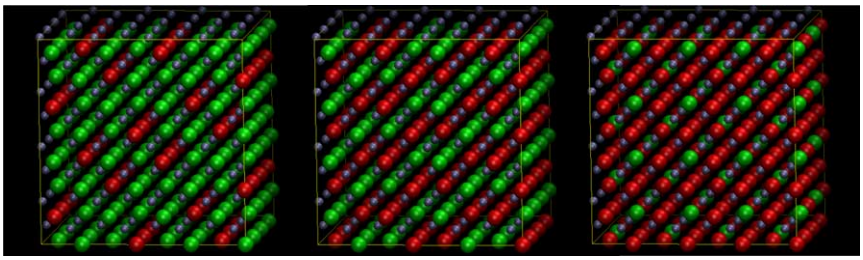

Fig. 20. The atomic configurations for maximum bandgap corresponding to the entries in Table 1. These are a $[0\,\bar{1}\,2]$ $(AlAs)_1(GaAs)_4(AlAs)_1(GaAs)_2$ superlattice (SL) for $Al_{0.25}Ga_{0.75}As$, a $[0\,\bar{1}\,2]$ $(AlAs)_2(GaAs)_2$ SL for $Al_{0.5}Ga_{0.5}As$ and a $[0\,\bar{1}\,2]$ $(AlAs)_3(GaAs)_1$ SL $Al_{0.75}Ga_{0.25}As$. Red(dark) spheres are Al, green(light) spheres are Ga, smaller gray spheres are As atoms. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

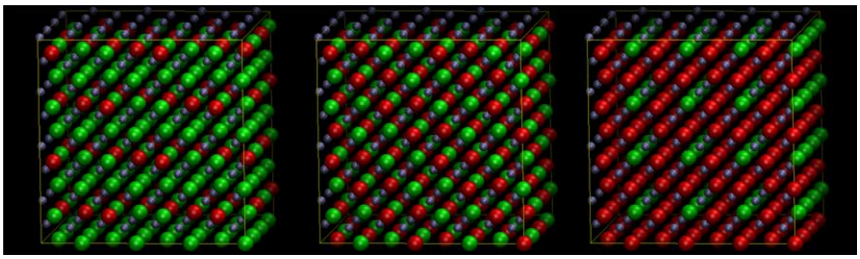

Fig. 21. The atomic configurations for minimum bandgap corresponding to the entries in Table 1. These are "X" Cmmm structure for $Al_{0.25}Ga_{0.75}As$, a $[1\,\bar{1}\,1]$ $(AlAs)_1(GaAs)_1$ superlattice for $Al_{0.5}Ga_{0.5}As$ and "Luzonite" Cu_3AsS_4-type for $Al_{0.75}Ga_{0.25}As$. See [35] for further descriptions of the "X" and "Luzonite" structures. Red(dark) spheres are Al, green(light) spheres are Ga, smaller gray spheres are As atoms. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

represent it in a 3 by 2 by 2 supercell, even if zone folding is not important (e.g., the optima arises from a direct bandgap at Γ). When we say that supercell A "contains" cell B, we mean not just that A is larger than B but that the infinite periodic repetition of B can be represented as an infinite periodic repetition of A. For example, 4 by 1 by 1 contains 2 by 1 by 1 in this sense, but 3 by 1 by 1 does not.

There are two obvious approaches to the problem of supercell choice. First, we can choose a single large supercell which we presume will encompass all the relevant zone foldings and small primitive cells. Second, we can perform runs in a series of smaller supercells, presuming that one of them will include the zone foldings and primitive cell of the global optima. Fig. 19 shows results for two sets of supercells corresponding to these two complementary approaches to this problem.

The set of runs in the large cell takes place in a 4 by 3 by 2 (192 atom) supercell. In the figure these are the solid lines with circles; open circles are the maxima, filled circles are the minima. Our rationale for this particular choice is that while still small enough to attempt an optimization, this cell contains many of the smaller supercells known from previous work [1] such as 4 by 1 by 2. For this cell we ran both maximizations and minimizations for composition $x = \frac{1}{12}i,\ i = 1, \ldots, 11$.

The set of runs in the small cells is shown as the triangles in the figure; up-triangles for maxima, down-triangles for minima. We ran both minimizations and maximizations for $x = 0.25$, $x = 0.50$, and $x = 0.75$ in a variety of supercells; these included $(2 \times 2 \times 2)$, $(4 \times 2 \times 1)$, $(3 \times 2 \times 2)$, and $(4 \times 2 \times 2)$. The cells in the figure, those in which the best solutions were found, are as follows: For the maximum case, for $x = 0.25$, a 4 by 2 by 1 supercell, for $x = 0.5$ and $x = 0.75$, a 2 by 2 by 2 supercell. For the minimum case, 2 by 2 by 2 supercells for all three compositions.

In the maximum case, the same solutions were predicted in [32] and found in previous inverse band structure work by simulated annealing [1]. For the minima, the best results were all achieved in 2 by 2 by 2 supercells. The discussion of "segregating states" in [29] supports our observations. They find that states can segregate such that some are localized on the GaAs sublattice and some are localized on the AlAs sublattice. Thus, the lower GaAs-like state becomes the conduction band minimum. The calculated bandgap approaches the Γ to L bandgap of GaAs, *provided the zincblende L state is folded to* Γ, since the L point states are those showing the segregating GaAs-like character. And the smallest such cell is 2 by 2 by 2.

For the large cells, we can see from the figure that our choice of a 4 by 3 by 2 supercell did not result in the global optima being discovered. If we compare this cell with those containing the optima found by the second approach, this result is not surprising. For the maximum case, the 2 by 2 by 2 cells are simply not contained (in the above sense) in the 4 by 3 by 2 cell, so it is not possible to find this solution. For the $x = 0.25$ case, the large supercell does contain the 4 by 2 by 1 (i.e., a 4 by 2 by 1 cell is equivalent to a 4 by 1 by 2 cell, so it would fit in a 4 by 3 by 2 cell by repeating it 3 times in the y direction). So it is interesting that it is simply the size of the search space, which not counting symmetry is $\binom{96}{24} \sim 10^{24}$ (including symmetry reduces it to approximately 10^{19}), that prevents us from finding the maximum. Note that the bandgap of the solution we do find is quite close to the global maximum, while the structure is not obviously similar. This is a common feature of heuristic search methods; they are good at finding high quality solutions, but not necessarily as good at finding the actual global optimum.

For minimization within the 4 by 3 by 2 supercell, the best solution found is far from optimal. Minima for small cells are ordered structures involving folding of the L point in the Brillouin zone to the Γ point (the bandgaps are indirect or pseudodirect from Γ to L), while minima for large structures are phase separated (tending toward the bandgap of the lower bandgap material, which for us is GaAs). For the 4 by 3 by 2 cell the L point in the Brillouin zone does not fold to the Γ point, so these low indirect bandgaps cannot be found by a computation which calculates electronic states only at the Γ point. The system finds the next best solution for this intermediate sized cell and tries to segregate itself physically into separate regions, each of pure material. These phase separated configurations are hard to achieve with simulated annealing.

The issue of supercell choice deserves further discussion. It would seem that a large enough supercell would contain all of the relevant zone foldings, thus one run of iaga in a large supercell should reveal

all the most important possible structures for a given composition. However, unless we use a prohibitively large supercell, it is not true that big supercells reliably contain all the smaller cells in which the global optima are contained. The only supercell containing all possible 8 atom unit cells is 8 by 8 by 8. This supercell contains 4096 atoms [33]. The more subcells our chosen supercell contains, the larger the search space. In fact, as the size of the supercell grows, the size of the search space grows much faster than the number of subcells it contains. So once our supercell contains all the structures of interest, it is too large to search! This is one of the important results of our efforts so far, and it provides the impetus for ongoing complementary research into more systematic ways to search just the smaller structures. Whereas here we have a relatively sophisticated algorithm searching a large space of possible configurations, this alternative approach involves systematically generating all the possible structures with a certain size unit cell [34] and calculating the bandgap for each. These unit cells are small but not necessarily orthogonal. For each we can calculate the minimal orthogonal unit cell in which we could represent the same structure. The cells in which iaga found the optima for the $x = 0.25$, $x = 0.50$, and $x = 0.75$ compositions contain the best structures we have found by this alternate method. The problem of supercell choice highlights the need to be clear about the goals of our optimization. If we are looking for a high quality solution, defined, say, as one which is within five percent of the true optimum, we *can* generally find it by looking for it in a large cell. But to go further toward the global optimum, we face a difficult situation. On the one hand, the optimizer cannot put the last atom in place to find the global optimum when there are so many possibilities. On the other hand, it cannot extract the ordered structures that may underlie the high quality but globally non-optimal solutions we *can* find. So if we are interested in high quality *ordered* structures, we are best off performing separate optimizations in each of a series of small supercells.

4. Conclusions

In this paper we have described our implementation of an efficient and accurate method for automated material design. It incorporates an existing forward solver with a genetic algorithm based inverse solver. The forward solver is an atomistic electronic structure calculation method with empirical pseudopotentials used for semiconductor alloys and their superstructures. The inverse solver is a parallel genetic algorithm based global optimization method which manipulates atomistic descriptions of populations of materials. The forward and inverse solvers are separable; changes can be made independently to either without affecting the other. Here we have studied the effects of different GA parameters and described a few strategies, illustrating them with examples of optimization of an Ising model test case Hamiltonian. Further, we have optimized AlGaAs alloys for maximum and minimum bandgaps and discussed the dynamics of the optimization process. Finally, we have shown the whole range of available bandgaps of an AlGaAs alloy achievable by different atomic configurations and compositions. We have discussed the important issue of the system size – the supercell must be commensurate with that of the global optimum yet not so large as to be unsearchable. This global optimization based approach can be applied to a variety of applications in material design.

Acknowledgments

The authors thank Dr. Alex Zunger for suggesting the use of genetic algorithm for this inverse band structure problem. We thank Dr. Lin-Wang Wang for providing us with the PESCAN codes. We used the VFF code originally written by Dr. James E. Bernard and then modified by other Solid State Theory team members. Discussions with Drs. A. Franceschetti, G.L.W. Hart, S.V. Dudiy, V. Blum, S.-H. Wei, G. Bester, J. Langou, A. Canning, P. R. C. Kent, L. Bellaiche, and Jooyoung Lee are greatly acknowledged.

This research used the computing resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098. We also acknowledge the computing resources at the Computational Sciences Center at NREL. The work was supported by the Office of Science of the U.S. Department of Energy – Office of the Advanced Scientific Computing Research – Mathematical, Information, and Computational Sciences program through Lab 03-17 Theory Modeling in Nanoscience initiative under Contract No. DE-AC36-99GO10337.

References

[1] A. Franceschetti, A. Zunger, Nature 402 (1999) 60.
[2] G.H. Jóhannesson, T. Bligaard, A.V. Ruban, H.L. Skriver, K.W. Jacobsen, J.K. Nørskov, Phys. Rev. Lett. 88 (2002) 255506.
[3] T. Cwik, G. Klimeck, Proceedings of the 1st NASA/DoD Workshop on Evolvable Hardware, IEEE, 1999.
[4] J. Íñiguez, L. Bellaiche, Phys. Rev. Lett. 87 (2001) 095503.
[5] A. Zunger, Phys. Stat. Sol. (b) 224 (2001) 727.
[6] K. Kim, P.R.C. Kent, A. Zunger, C.B. Geller, Phys. Rev. B 66 (2002) 045208.
[7] K.A. Mäder, A. Zunger, Phys. Rev. B 50 (1994) 17393.
[8] L.-W. Wang, A. Zunger, Phys. Rev. B 51 (1995) 17398.
[9] A.J. Williamson, L.-W. Wang, A. Zunger, Phys. Rev. B 62 (2000) 12963.
[10] L.-W. Wang, J. Kim, A. Zunger, Phys. Rev. B 59 (1999) 5678.
[11] A. Franceschetti, A. Zunger, Phys. Rev. B 52 (1995) 14664.
[12] J. Kim, L.-W. Wang, A. Zunger, Phys. Rev. B 56 (1997) R15541.
[13] L.-W. Wang, A. Zunger, J. Chem. Phys. 100 (1994) 2394.
[14] A. Canning, L.-W. Wang, A. Williamson, A. Zunger, J. Comput. Phys. 160 (2000) 29.
[15] L.-W. Wang, L. Bellaiche, S.-H. Wei, A. Zunger, Phys. Rev. Lett. 80 (1998) 4725.
[16] P. Keating, Phys. Rev. 145 (1966) 637.
[17] C. Pryor, J. Kim, L.-W. Wang, A.J. Williamson, A. Zunger, J. Appl. Phys. 83 (1998) 2548.
[18] J.H. Holland, Adaptation in Natural and Artificial Systems, second ed., MIT Press, Cambridge, 1992.
[19] D. Levine, Users guide to the PGAPack parallel genetic algorithm library, Technical Report ANL-95/18, Argonn National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439, January, 1996.
[20] A.E. Eiben, J.E. Smith, Introduction to Evolutionary Computing, Springer, Berlin, 2003.
[21] T. Bäck, D.B. Fogel, Z. Michalewicz, Handbook of Evolutionary Computation, Oxford University Press, New York, 1997.
[22] S.V. Dudiy, K. Kim, W. Jones, A. Zunger (unpublished).
[23] H. Muehlenbein, How genetic algorithms really work I. Mutation and hillclimbing, in: R. Männer, B. Manderick (Eds.), Parallel Problem Solving from Nature, vol. 2, Elsevier, Amsterdam, 1992.
[24] Z. Michalewicz, D.B. Fogel, How to Solve It: Modern Heuristics, Springer, Berlin, 2000.
[25] G. Harik, E. Cantu-Paz, D.E. Goldberg, B.L. Miller, in: Proceedings of the 1997 IEEE International Conference on Evolutionary Computation, 1997, pp. 7–12.
[26] N.W. Ashcroft, D. Mermin, Solid State Physics, Holt Rhinehart, New York, 1976.
[27] M. Laguna, R. Marti, Scatter Search, Kluwer Academic Publishers, Boston, 2003.
[28] V. Blum, G.L.W. Hart (private communications).
[29] S.-H. Wei, A. Zunger, J Appl. Phys. 63 (1988) 5794.
[30] L. Ingber, Contr. Cybernet. 25 (1996) 33.
[31] T.W. Manikas, J.T. Cain, Technical Report 96-101, Department of Electrical Engineering, University of Pittsburgh, May, 1996.
[32] R. Magri, A. Zunger, Phys. Rev. B 44 (1991) 8642.
[33] G.L.W. Hart (private communications).
[34] L.G. Ferreira, S.H. Wei, A. Zunger, Int. J. Supercomputer Appl. 5 (1991) 34.
[35] L.G. Ferreira, S.H. Wei, A. Zunger, Phys. Rev. B 40 (1989) 3197.