Bull. Mater. Sci. (2018) 41:1
https://doi.org/10.1007/s12034-017-1515-9

First-principle-based computational doping of $SrTiO_3$ using combinatorial genetic algorithms

EMRAH ATILGAN[1] and JIANJUN HU[1,2,*]
[1]Department of Computer Science and Engineering, University of South Carolina, Columbia, SC 29208, USA
[2]School of Mechanical Engineering, Guizhou University, Guiyang 590050, China
*Author for correspondence (jianjunh@cse.sc.edu)

MS received 4 February 2017; accepted 10 March 2017; published online 1 February 2018

Abstract. Developing new materials has historically been time-consuming. One commonly used approach is material doping, in which given a base material, one can change its properties by substituting some elements with new ones or adding additional elements. Computational material discovery involves searching in a large design space to identify candidates for experimental verification. Recently, it was possible to obtain many electrical and physical properties of materials by density functional theory based first-principle calculation, making it suitable for computational doping-based material discovery. In computational doping, one can substitute some of the atoms in a supercell with dopant atoms. However, the actual positions of the dopant elements within the supercell are not known. In this work, we developed a genetic algorithm for finding the most stable structure of the doped material with the lowest free electronic energy. For each candidate atom configuration, we use the Vienna Ab-Initio Simulation Package to calculate its physicochemical properties, which takes about 15–30 h for a supercell grid of 75 atoms. We did computational doping on $SrTiO_3$ perovskite. Experiments showed that our method can reduce the running time for computational doping by up to 70% compared with exhaustive sampling as commonly used now.

Keywords. Computational doping; material doping; genetic algorithm; combinatorial optimization; VASP; DFT.

1. Introduction

New materials are the step-stones of today's technology revolution. However, discovering and developing a commercially successful material is extremely difficult and takes a lot of cost and time. This process could be accelerated by computational studies based on recent progress in material simulation theory such as density functional theory (DFT) and progress in computing speed. Since experimental material studies are time-consuming and expensive, computational studies have become an important area for material discovery and design. Nowadays, material scientists prefer to explore the material design space first computationally to identify the most promising material candidates and then try to verify them in their wet laboratory. The studies on computational material discovery and design have increased significantly in recent years [1].

In this paper, we focused on computational discovery of materials for solid oxide fuel cells (SOFCs), which are energy conversion devices that convert chemical energy directly into electrical energy. Compared with other fuel cell technologies, SOFCs are fuel-flexible devices and are able to use different gases, such as hydrogen, biogas, natural gas, methane, butane, etc. They can minimize emission and are thus environment-friendly. Another advantage of SOFCs over other power generation systems is that they work quietly with no vibration. Additionally, SOFCs can be used as co-generators with gas turbine power systems to enable full exploitation of electricity and heat. In this system, the efficiency can be increased up to 70% [2]. SOFCs have been used in many areas, including distributed power generation, electric vehicles, portable power for military and consumer electronics such as smart phones. In recent years, studies on SOFC have significantly expanded due to its broad application area [3].

One of the main issues of current SOFCs is that they have to run at high temperature. However, high-temperature operation of SOFC needs time to start up and cool down. Current studies are focused on low-temperature SOFCs. Lower temperature SOFC increases system stability and durability. Moreover, inexpensive metallic interconnections could be used at lower temperatures in place of lanthanum chromite-based ceramic interconnections, which require expensive fabrication costs. However, at lower temperatures such as 600°C, the electrode kinetics and electrolyte conductivity decrease dramatically. For example, lowering the temperature increases the YSZ electrolyte resistance [4].

Many experimental studies have been conducted to search for ideal materials for low-temperature (500–600°C) SOFC systems in the past decades [5]. However, it is a big challenge to develop suitable satisfactory electrolyte and electrode materials using only the experimental approach because many requirements have to be met at the same time and the

design space is enormous. Currently, the major approach is experimental doping, wherein tries the material properties of a model material system are modified by adding or substituting some elements to a base material.

The material doping problem can be described as the species/elements that can be added to the base material in particular compositions in order to achieve desired material properties. It is sometimes called as an inverse design problem [6]. The most common approach in doping based material design is to substitute some elements in a material with new ones. This substitution is done in a small percentage to keep the balance of the original material's characteristics with the new element's effects. Both too much or too little dopants are not able to achieve desired results. In experimental doping, one just mixes these ingredients, synthesizes the samples and measures their properties. In computational doping, a supercell is created from the primitive cell of the base material along with dopant atoms. However, given a ratio of base material to substitution elements, the actual configuration of the dopant atoms within the supercell is now known and is determined by the electromagnetic energy, which can be obtained by DFT calculations. Current practice is to exhaustively calculate all the possible dopant element configurations within a supercell and identify the ones with the lowest energy and use it for final DFT calculation to get its physical properties. However, DFT calculation for large supercells may take days for each configuration. Identifying the lowest energy configuration of doped supercells can be naturally mapped as a combinatorial optimization problem, for which genetic algorithms (GAs) have showed great success in the past.

GAs [7,8] are powerful search algorithms for applications in various science and engineering fields. In many real-world optimization problems such as design optimization [9] and scheduling problems [10], GAs give satisfactory results in a reasonable amount of time. GA-based optimization algorithms have also been applied to combinatorial problems, such as Traveling Salesman Problems (TSPs) [11], Minimum Spanning Tree (MST) [12] and the Assemble Line Balancing Problems [13]. The hardness of these problems comes from the huge search space that a GA has to explore. However, in some cases, the objective function, which GAs calculate as the fitness value, may need long time to calculate. Such expensive evaluations make it impractical to explore the whole search space, making GAs desirable for finding optimum solutions for such problems. This is especially true for problems with exponentially increasing search space regarding the problem sizes. One of these problems is computational material design based on expensive DFT calculations.

In this paper, we present a combinatorial GA to reduce the number of fitness calculations to identify the physically probable dopant atom configurations. It can dramatically reduce the computational time for computational doping-based material design. We tested our algorithm on the computational doping problem for fuel cell material design and discovery. Our GA was tested on finding the best doping positions for Nb-doped $SrTiO_3$ material, a SOFC material. Experimental results showed that our algorithms can reduce the computational effort by up to 70% compared with exhaustive search as commonly done by the computational doping community.

1.1 *Related work*

1.1a *Optimization with expensive objective functions*: Vienna Ab-Initio Simulation Package (VASP)-based computational doping is a typical optimization problem with expensive evaluations. Optimization problems with expensive fitness evaluations have been studied before [14]. In such problems, the running time for a single fitness calculation can take hours to days even with supercomputers. One approach to avoid such expensive calculations is to use surrogates: instead of calculating original expensive fitness function, using approximations can dramatically reduce the total computation time [14]. Kim and Cho [15] proposed to use clustering to reduce the number of fitness evaluations. Their method divides the population into several clusters, and then one representative from each cluster is calculated. The fitness values of the remaining individuals are estimated from the representative individual's fitness. In addition to the clustering method, Rasheed [16] calculated a distance-weighted k-nearest-neighbour to classify individuals as 'potentially feasible, infeasible and un-evaluable' to compute. Later, least-square quadratic approximations are used to calculate the approximate fitness. In our algorithms, we record all the fitness evaluations and avoid duplicate evaluations of supercell configurations. Symmetry in the doped supercell can also be exploited to further reduce the fitness evaluations.

1.1b *Computational doping based on DFT calculation*: Due to the conflicting requirements of fuel cell materials, a large number of doping experiments have been conducted to develop novel materials for both anode and cathode sides [17,18] to lower the SOFC operation temperature. However, searching potential doping elements experimentally is a time-consuming and expensive process. Despite years of experimental efforts, the ideal set of SOFC materials remains to be discovered. Thus, theoretical data and computational approaches are needed for guiding experimental doping to help the SOFC material discovery process.

To speed up the search for better SOFC materials, it is helpful to explore the design space using first-principle DFT calculations, in which quantum simulation is used to calculate atom-level material properties and energies. DFT calculations have been applied to explore doped materials since 1978, and the number of such studies is increasing recently as reviewed by Hautier *et al* [19]. The most common approach is to first build a supercell from unit cells (see figure 1) and then replace the elements at some positions with possible dopant atoms and then apply DFT calculations. Guo *et al* [20] studied the electronic band structure of $SrTiO_3$ with different Nb-doped concentrations using first-principle calculations. Zhang *et al* [21] studied the electronic structure and optical properties in heavy-metal-doped ZnO using DFT-based structural and

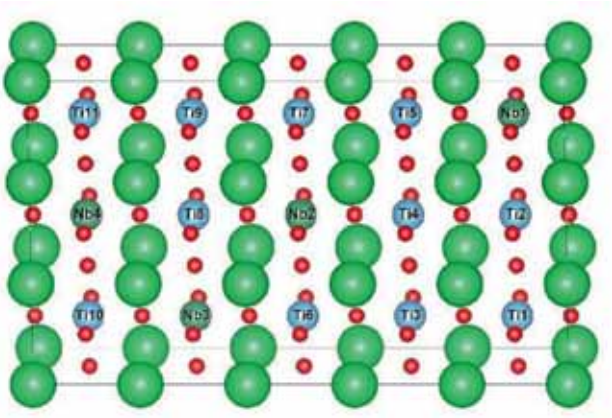

Figure 1. The grid of Nb-doped $SrTiO_3$. The 4 Nb atoms are in the 3rd, 8th, 10th and 14th positions.

band structure calculations. Supercells with 32, 64, 72 and 108 atoms were used in their calculations. They found that Ag- and Au-doped ZnO have little lattice mismatch, while Pt-doped ZnO has a large lattice mismatch. Additionally, the structural, electronic and magnetic properties of Ca-doped α-Cr_2O_3 (chromium oxide) crystal have been investigated using the DFT calculation [22]. It was found that the electronic band structure and the band-gap width are in close agreement with the experimental results. Oxygen vacancy formation and migration in N-doped Cu_2O [23] and Sr- and Mg-doped $LaGaO_3$ [24] were investigated using DFT principles. They calculated the energies based on different modifications of the band structures and oxygen vacancy formations.

First-principle studies have also been applied to fuel cell materials [25,26]. Zhou *et al* [27] studied oxygen reduction reaction (ORR) on cathode material $LaSrCoO_4$ using the periodic DFT (DFT + U). They investigated that the Co site is a more preferred site for the absorption of oxygen. Shishkin and Ziegler [28] also performed a DFT + U study of the electronic structure and chemical properties of the Ni–CeO_2 and Ni–CeO_2–YSZ systems and compared the change in the electronic charge localization. Chen *et al* [29] studied spin-polarized DFT calculations to investigate ORR on Sr-doped $LaMnO_3$ cathode material. Ahmad *et al* [30] used the CRYSTAL09 software package and studied the thermodynamic phase stability of $LaMnO_3$ and its competing oxides. An *et al* [31] applied DFT to determine catalytic activity of bimetallic nickel alloys for SOFC anode reactions.

2. Material and methods

Given a base material such as $SrTiO_3$, there are dozens of possible dopant elements for Sr–Ti–O, and for each dopant element, there are many possible positions for the substitution, which leads to a large complex doping space. It is practically infeasible to exhaustively search all possible doping configurations using DFT calculation since each DFT calculation of a supercell with about 75 atoms needs 30–35 h. To reduce the number of such expensive DFT simulations, we developed a combinatorial GA to search the dopant element configuration space on the supercell to find the most stable configuration with the lowest free electronic energy. This configuration is most likely to correspond to the real atomic structure, which allows calculating additional material properties such as band structures, electric conductivity, etc. In the following sections, we will describe the details of our GA, including the representation of individuals, the mutation and crossover operators, the usage of elitism, the type of selection method and the objective function for our computational doping experiments.

0	0	1	0	0	0	0	1	0	1	0	0	0	1	0

Figure 2. A representation of an individual that places 4 Nb atoms at 3rd, 8th, 10th and 14th position of 15 possible positions.

Since we look for the best configuration of atoms in the supercell, our purpose is to find the best doping positions among all possible configurations. Consider the example of a supercell consisting of 75 atoms of $SrTiO_3$ material with Nb as the dopant element. If 4 Ti atoms are substituted with 4 Nb atoms, it makes approximately 27% Nb-doped $SrTiO_3$ (figure 1). Since there are 15 Ti atoms in the supercell, we need to choose 4 of them to substitute with Nb. It turns our problem into a combinatorial optimization problem. Mathematically, 15-choose-4 is equal to 1365, which means there are exactly 1365 different options for the placement of Nb atoms in the supercell for the Nb–Ti substitution (see figure 1). The energy of each configuration of dopants will be evaluated by running VASP [32] simulation, which is the state-of-the-art DFT calculation software. Identifying the configuration with the lowest energy by exhaustively calculating all these many VASP calculations is simply impractical because each of them takes approximately 13–16 h to complete. Our goal is to exploit the global search capability of GA to find the optimal or approximate optimal configurations with significantly less DFT calculations.

Our GA is generation based, with fixed numbers of generations. A binary representation is used for individuals but with some constraints compared with traditional binary representations (see figure 2). Each individual's length is equal to the number of possible doping positions. The bits with 1 indicate the occupation of dopant elements at the corresponding positions. With this representation, we start the genetic search by creating a random population (see figure 3). Simultaneously, we create another array, named *Pool*, which holds all evaluated configurations. As soon as we create and evaluate an individual for the population, we put this individual into the *Pool*. After the crossover or mutation operator creates a new individual, we first check the *Pool* if the new individual has been evaluated before or not. In our algorithm, we do not want to evaluate the same individual twice to avoid the fitness calculations, which are the most time-consuming tasks in the algorithm. Instead, we keep all the fitness values from previous

calculations in the *Pool* for future reuse. If crossover or mutation operators create an individual evaluated before, we substitute it with a different yet similar individual from the *Pool*.

To handle the expensive fitness evaluations, we develop a mechanism to avoid duplicate fitness evaluations. Traditional GAs can generate identical individuals that were already evaluated in previous generations or even in the same generation through the crossover or mutation operator. To save fitness evaluations, such evaluations should be avoided for duplicate individuals. Thus, we implemented our GA in three different cases depending on how we avoid evaluating the same individual multiple times and what we should do when a created individual has already been evaluated before: (1) GA-basic algorithm: it checks the new individual created by crossover and mutation operators. If it has been evaluated before, it randomly creates a similar but different individual. (2) GA-SS algorithm: it checks the individual after it is created by crossover and mutation, and if the individual has been evaluated before, GA-SS creates another similar individual whose gene values are assigned based on their distributions of previously evaluated individuals. (3) GA-SC: it uses statistical idea similar to that of GA-SS; however, it applies these statistics during the crossover process. The details about these three different versions will be explained in corresponding subsections and the results will be explained in the 'Results' section.

After creating the initial population, the algorithm calculates the fitness values of the population and is ready to start the generation loop. We used a fixed number of generations to keep the number of evaluations fixed. In the loop, first we find the *elite* individuals in the population based on user-defined elite percentage. These elite individuals can be used for crossover and mutation with any others. However, eventually they will survive to the next generation. Among the other selection methods, *Tournament selection* is used as the selection operator with tournament size 3. We tested different selection methods to find the most suitable one for our GAs for 13% Nb-doped $SrTiO_3$ doping problem (see Section 3.1). Tournament selection, Roulette-wheel selection and Random selection methods were tested by keeping the other parameters such as population size and number of generations the same. For each run we set the population size to 40 and maximum number of generations to 10, and then ran our GA-basic algorithm. Since we already have exhaustive search results, we compared the genetic search results with different selection methods to the best known results. To avoid the randomness, we ran our GA 100 times for each selection method and compared the results in table 1. The 'rank' column shows the means and standard deviations of the ranks of the best found results for the 100 independent runs with each selection method, while the other column shows in which generation the GA converged to the best found individuals. With tournament selection, the GA can find the best result in most runs consistently compared with the other selection methods. Although the GA converges faster with Roulette-wheel selection, this method is not efficient to select good parents to explore the search space and tends to get stuck in local optima. Although we report here the comparison of selection methods for only GA-basic, Tournament selection outperforms other selection methods for GA-SS and GA-SC algorithms as well. Thus, we used Tournament selection for all the GA implementations in this study.

```
Set parameters
Initial population
Fitness calculation
Generation loop:
        Find elite individuals
        Selection
        Crossover
        Mutation
        Fitness calculation
        Combine elite and new pop
End loop
```

Figure 3. Pseudo-code of our algorithm.

The selected parents are used in our uniform *crossover* operator. The number of parents for crossover operator is 2(*PopSize-Elites*). Since the 1's in the individuals represent the actual position of the individual dopant elements, there will always be the same number of 1's. Thus, we divide each individual to get an equal number of 1's at the same side. If we have an even number of 1's, say M, in an individual, both parts of each parent will have $M/2$ 1's. If it is odd, however, the first part of each parent will have $(M+1)/2$, and the second part will have $(M-1)/2$ 1's. Then, we exchange the first part

Table 1. Comparison of selection operators in GA.

	Rank		Best-hit generation	
Method	Mean	Std. dev.	Mean	Std. dev.
Tournament selection	1.47	1.14	5.53	2.61
Roulette-wheel selection	5.47	4.91	5.52	3.12
Random selection	3.1	2.8	5.77	2.88

'Rank' column shows that the rank of the best result in the best known results from exhaustive search. 'Best-hit-generation' column shows in which generation the GA converged to its best result; 100 independent GA runs were conducted to calculate the mean and standard deviations of the ranks.

i) If there are no common positions

Let the parents be:

3	5	7	13	and	2	4	6	8

Then, uniform crossover is applied, and the offspring will be:

3	5	6	8	and	2	4	7	13

ii) If there are common positions:

- *If there is only one different position for each parent.*

In this case, if we apply uniform crossover, then the offspring will be the same with parents. Ex:

4	6	7	12	and	2	6	7	12

2	6	7	12	and	4	6	7	12

Instead, we search for the other available positions from the *Pool* that includes the common positions. Let us say 1^{st}, 3^{rd}, 5^{th}, 8^{th}, 9^{th}, 10^{th}, 11^{th}, 13^{th}, 14^{th}, 15^{th} positions was not used with 6^{th}, 7^{th}, and 12^{th} positions together so that to build the individuals, such as [1, 6, 7, 12], [3, 6, 7, 12], or so on. Then we randomly choose one of them to create a similar offspring:

6	7	8	12	and	6	7	9	12

- *If there are 2 or more different positions for the parents:*

The idea is to crossover the different parts of parents and keep the common positions unchanged. E.g: Let the parents be:

4	6	7	12	and	2	5	7	12

Then, the offspring will be :

4	5	7	12	and	2	6	7	12

Figure 4. The crossover operator of the GA-basic algorithm.

of the first parent with the second part of the second parent to create the first child. The second child is created with other parts. Since there should be fixed number of 1's in the representation of an individual, uniform crossover may create a child with more or less 1's. In other words, if we are looking for 4 positions to place our substitution element, crossover operator may create a child with two 1's, and another child with six 1's, which is not acceptable. Another problem with uniform crossover is when parents have some common positions. We will explain our crossover operator with an example (see figure 4).

After crossover, some individuals are chosen for *mutation* with user-defined mutation probability. The idea behind the mutation is simply flipping bits. One of the 1's is chosen randomly and swapped with a randomly chosen 0. The *fitness* function is called for new individuals after crossover and mutation. Finally, these new individuals and elite individuals are added together to create a new population for the next generation.

2.1 *Material preparation*

We tested our algorithms on 2 different systems of Nb-doped $SrTiO_3$ material: 13% Nb-doped $SrTiO_3$ (2 Nb atoms in 15 possible positions) and 27% Nb-doped $SrTiO_3$ (4 Nb atoms in 15 possible positions).

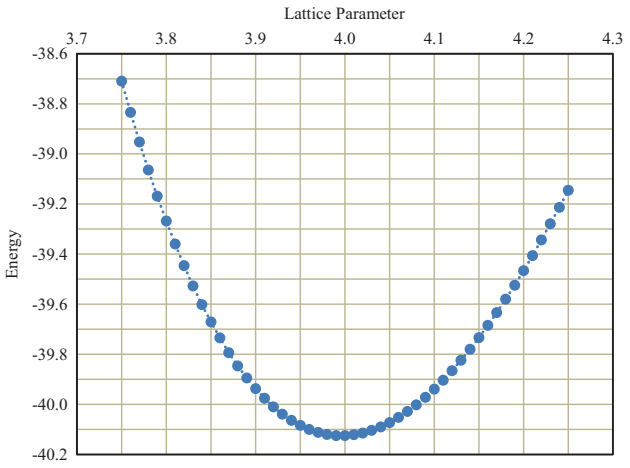

Figure 5. Lattice parameter optimization for the $SrTiO_3$ primitive cell.

Since the material preparations are similar for both systems, only the 27% Nb-doped $SrTiO_3$ (4 Nb atoms in 15 possible positions) will be explained in detail. The crystal structure for the primitive cell of $SrTiO_3$ material was obtained from materialproject.org [33]. We initially optimized the lattice parameter of the $SrTiO_3$ unit cell. Figure 5 shows the energy values of different lattice parameters in the search process. Our optimal lattice parameter with the lowest energy (3.946) is consistent with materialproject.org (3.945).

After lattice optimization, a supercell of 75 atoms containing 15 unit cells ($5\times3\times1$) was created from fully relaxed structure. A template file is created by substituting 4 Nb atoms with 4 Ti atoms from the POSCAR file of $SrTiO_3$ supercell. When the GA creates, an individual representing the positions in which the Nb atoms will be placed, then the actual POSCAR file for this individual is created from the template file. For this system of Nb-doped $SrTiO_3$, there are 1365 possible configurations. Exhaustively evaluating all these configurations would take almost 3 months on a 10-node Linux cluster.

In our calculations, we used the free electronic energy as the fitness value of individuals in our GA based on plane-wave DFT implementation of VASP 5.3 [32]. The exchange and correlation potential was treated within the Generalized Gradient Approximation (GGA) with the Perdew-Burke-Ernzerhof (PBE) functional [34]. The interaction between ions and electrons was described by the projector-augmented wave (PAW) method [35]. The cut-off for the kinetic energy was set to 520 eV for all calculations with respect to ENMAX values of corresponding elements in POTCAR file.

2.2 *Implemented GAs*

As mentioned earlier, we implemented and evaluated three GAs for finding stable doped materials based on VASP DFT calculation.

2.2a *GA-basic*: GA-basic is a traditional generational GA for our doping problem. It includes our special representation type. Since our main goal is to reduce the number of fitness calculations while still finding the optimal solution, our GA does not evaluate the individuals that have already been evaluated before. If the crossover or mutation operators create an individual that has already been evaluated before, we need to replace it with a new one. In this case, GA-basic randomly chooses one single position of the individual, let us say 8th position of [3,5,8,11], and then looks for individuals consisting of the remaining positions, 3th, 5th and 11th. Finally, the algorithm chooses one of the available individuals from the *Pool* and creates the new individual. If there is no individual in the *Pool* consisting of these 3th, 5th and 11th positions, then the algorithm chooses another position to change, let us say 5th, and looks for the individuals consisting of 3th, 8th and 11th positions. If there is no individual left in the *Pool* for 3-tuples, then the algorithm randomly chooses a pair, let us say 3th and 5th positions, and looks for the individuals consisting of 8th and 11th positions in the *Pool*. This procedure is repeated for 3-tuples to change and one position to keep, if no individuals are found in the *Pool*.

2.2b *GA-SS*: The second version of our implementation is GA-SS—GA with statistical similarity. Here we use a statistical procedure to pick better parents for the mutation and crossover operators to generate better offspring. In GA-basic, if an individual is created that has already been evaluated before, the algorithm will choose a random but similar positions for individuals from the *Pool*. Instead, GA-SS runs a statistical process. Instead of choosing only similar positions, GA-SS checks the previously evaluated individuals containing each position separately. For example, let the individual be [3,8,10,14], which placed the Nb atoms in 3th, 8th, 10th and 14th positions, respectively. We randomly choose the 8th position and we will keep 3th, 10th and 14th positions occupied. Then, we look for the candidates from *Pool* that have not been used before. We determine the positions consisting of [3,10,14] and keep the fourth position as the candidate for substation for 8th. Let us say 1st, 2nd, 5th, 7th and 12th positions are available and can be used. Then, we calculate the average fitness values of the individuals that have already been calculated and choose the best position, which has the maximum average fitness (lowest energy). Let the 5th position be the one that has the maximum; then we create the individual as [3,5,10,14]. This procedure is repeated if there is no individual left in *Pool*, consisting of [3,10,14]. If so, we choose another single element from [3,8,10,14] except what we already chose before, 8th. If all the possibilities are over for the single elements, the procedure is repeated for the pairs, such as [8,10], to substitute and to keep the rest.

2.2c *GA-SC*: The last implementation of our GA is GA-SC, a GA with statistical crossover. The method is similar to that of GA-SS. However, we implement this statistical

sampling process at a different level of the GA. While GA-SS uses statistical sampling after crossover or mutation, GA-SC uses statistical sampling during the crossover process. The statistical sub-combinatorial approach (explained in GA-SS) is applied to parents who have the common positions. GA-SS still allows running uniform crossover in part (i) in figure 4; on the other hand, in part 2, GA-SC uses statistical sampling to choose 'not-common' positions rather than choosing randomly.

3. Results

We implemented our GAs in MATLAB, and tested their performances using the fitness values obtained for all the 1365 possible doping configurations of exhaustive search. We compared our different implementations of the GA with each other and with exhaustive search to see how much our algorithms can speed up the whole optimization process to find the most stable doping configurations.

3.1 *GA performances on 13% Nb-doped $SrTiO_3$*

In this system, the supercell is created from 15 unit cells (5×3×1). There are 15 Sr, 15 Ti and 45 O atoms in the supercell, and the goal is to find the doping positions with the lowest free electronic energy replacing 2 out of 15 Ti positions with Nb atoms. There are 105 possible doping configurations in total for these experiments if run exhaustively. Calculating the free electronic energy of each configuration using VASP DFT package needs 8–10 h on our Linux cluster computing node with 6-core 2.46-Ghz CPU with 24 GB of RAM. Although this search space is small for a GA to expose its full capability, we did this experiment to see how our GA's behaviour changes from small search spaces to larger ones. The fitness landscape space is shown in figure 6. We duplicated the values for symmetry view in 3D. It means we have only individuals such as, for example, [3,5], but not [3,5], since they occupy the same positions for dopant elements. Also, we set the values for couples like [3,3] a little lower than the maximum value of all results. The search space showed that there exists gradient information that GAs can exploit to quickly find the optimal or near-optimal results.

In the first experiment, we tested the ability of our different GA implementations for finding the optimal doping configurations given a fixed number of evaluations. Here we allowed only 42 fitness evaluations (PopSize is 8 and number of generations is 5, excluding the initial population) out of 105, and compared the best-found results of three GAs with respect to the exhaustive search results. Additionally, we executed 50 independent runs for each algorithm to check the robustness of the algorithms (see figure 7). Even though the population size and number of generations are so small for a GA, it is found that with only 40% of total evaluations, even

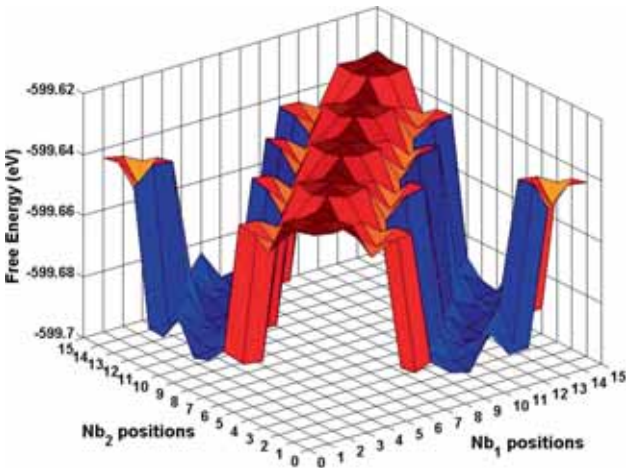

Figure 6. The search space for 13% Nb-doped $SrTiO_3$.

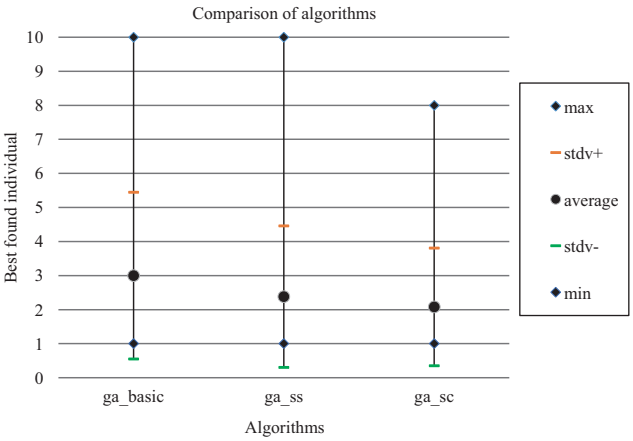

Figure 7. Results of 50 independent runs for algorithms given 42 fitness evaluations. The numbers in columns show the ranks.

the basic GA can find one of the top 3 results in most of the 50 runs. The GA-SS algorithm had a similar or slightly better performance. The GA-SC with statistical crossover works the best as indicated by figure 7. For most of the runs, it successfully located the best solution and it found one of the top 3 results with only 42 out of 105 evaluations. This means we can save 63×8 = 504 h of computing resources on a 6-core high-end computer.

In the second experiment, we tested how fast our algorithms find the lowest energy doping positions. Thus, we ran the algorithms as long as possible (PopSize is 10 and number of generations is 12), then checked when they converged and in what generation they found the lowest energy (worst-case scenario) doping configurations. Figure 8 shows the final results. It shows that the GA-SC not only found one of the top 3 best solutions out of the 40 fitness run experiments as shown in figure 7, but also found such solutions much faster than the other two algorithms.

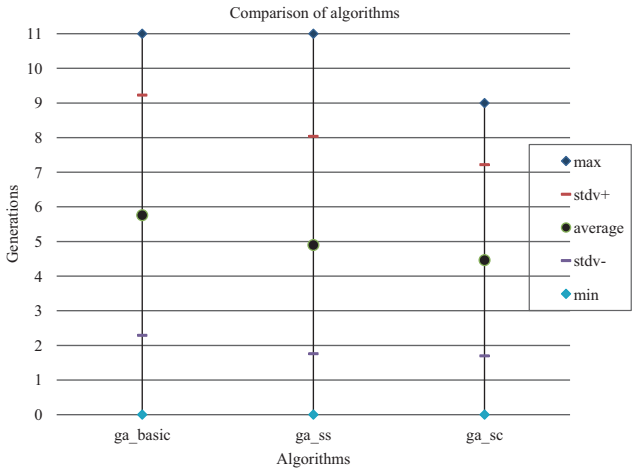

Figure 8. Fifty independent runs of three algorithms to check when the algorithms find the best solutions. The numbers in columns show the ranks.

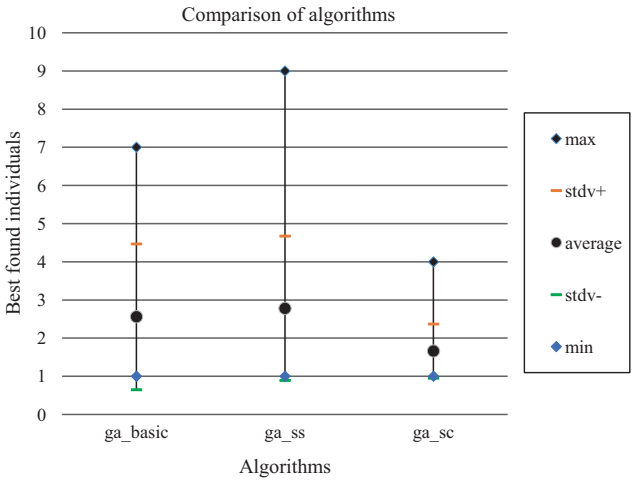

Figure 9. Results of 50 independent runs for algorithms given 400 fitness evaluations. The numbers in columns show the ranks.

3.2 *GA performances on 27% Nb-doped $SrTiO_3$*

In this system, the supercell is created from 15 unit cells ($5\times3\times1$). There are 15 Sr, 15 Ti and 45 O atoms in the supercell. Our goal is to find the best positions for 4 Nb atoms to substitute 4 Ti atoms. In this real-world optimization problem, 1365 DFT calculations are needed to find the best dopant positions if run exhaustively and each takes approximately 15 h on a 12-core 2.4-GHz CPU Linux computing node.

Similar to 13% Nb-doped $SrTiO_3$ experiments, we tested the ability of our different GA implementations for finding the optimal doping configurations given a fixed number of evaluations. Here we allowed only 400 fitness evaluations out of 1365, and compared the best-found results of three GAs with the exhaustive search results. The population size was set to 40 and only 10 generations were run. Additionally, we executed 50 independent runs for each algorithm to check the robustness of the algorithms (see figure 9). It is found that with only 29% of total evaluations, the basic GA can get one of the optimal results in half of the 50 runs. The GA-SS algorithm had a similar performance. In some cases, it found better results than those of the basic GA. The GA-SC with statistical crossover works the best, as indicated by figure 9. In 49 of the 50 runs, it successfully located one of the top 3 results with only 400 changes out of 1365 evaluations. This means we can save $965\times15 = 14475$ h of computing resources on a 12-core high-end computer.

Similarly, in the second experiment, we also tested how fast our algorithms find the lowest energy doping positions for this real-world computational doping experiments. Thus, we run the algorithms as long as possible until they find the optimal doping position. We then checked when they converged and in what generation they found the lowest energy (worst-case scenario) configuration. It shows that the GA-SC not only found one of the top 3 best solutions out of the 400 fitness run experiments as shown in figure 9, but also found such

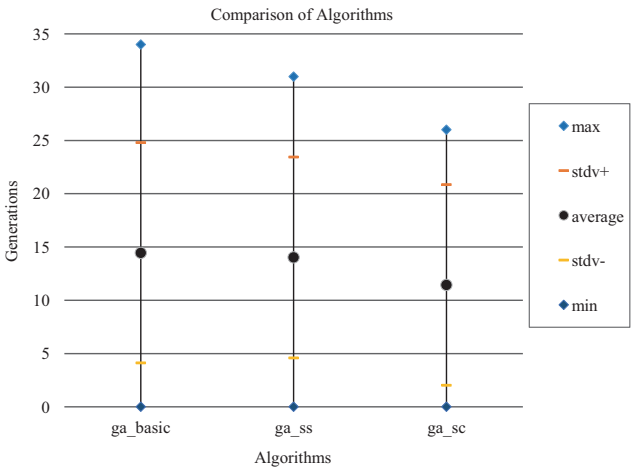

Figure 10. Fifty independent runs of three algorithms to check when they find the best solution. The numbers in columns show the ranks.

solutions much faster than the other two algorithms. On average, it finds the best solutions after on average 13 generations have been run, which is significantly better than the other two algorithms. Compared with exhaustive search, it uses only about 30% of evaluations to find the optimal solution, which is a great saving in terms of computational resource, speeding up computational doping experiments (see figure 10).

4. Discussion and conclusions

We have proposed three versions of specialized combinational GAs for finding stable doping positions in computational material doping based on VASP DFT calculations. Experimental results showed that our algorithms are robust and can achieve up to 70% saving of the computational resources

and speed-up compared with exhaustive search approach currently used by most of the computational material scientists. Since each fitness evaluation costs 15–30 h in most of our experiments, it is extremely important to avoid evaluations of duplicate individuals or very similar individuals. It is also important that the designed GA does not waste simulations on disrupted individuals with very low fitness values, usually resulting from large mutation operations. The main idea of our specialized GA is to use statistical sampling to generate the most promising candidate individuals for evaluation. Our GA with statistical crossover was shown to have the best performance among all three algorithms we tested, which confirmed our observation. In our future work, we plan to further improve the GA algorithms so that they can adaptively search the design space. We will also apply genetic programming to search different dopant elements and their doping positions simultaneously.

References

[1] Curtarolo S, Morgan D and Ceder G 2005 *Calphad* **29** 163
[2] Zhu W and Deevi S 2003 *Mater. Sci. Eng. A* **362** 228
[3] Kendall K, Singhal S C and Kendall K 2003 *High-temperature solid oxide fuel cells: fundamentals, design, and applicatons* (Oxford, UK: Elsevier Advanced Technology)
[4] Will J, Mitterdorfer A, Kleinlogel C, Perednis D and Gauckler L J 2000 *Solid State Ion.* **131** 79
[5] Jacobson A J 2010 *Chem. Mater.* **22** 660
[6] Perkins J D *et al* 2011 *Phys. Rev. B* **84** 205207
[7] Holland J H 1975 *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence* (Cambridge, MA: MIT Press)
[8] Goldberg D E 1989 *Genetic algorithms in search, optimization and machine learning* (Boston, MA: Addision-Wesley Longman Publishing Co. Inc.)
[9] Caldas L G and Norford L K 2002 *Autom. Constr.* **11** 173
[10] Gonçalves J F, De Magalhães Mendes J J and Resende M G C 2005 *Eur. J. Oper. Res.* **167** 77
[11] Carter A E and Ragsdale C T 2006 *Eur. J. Oper. Res.* **175** 246
[12] Zhout G and Gen M 1998 *Comput. Oper. Res.* **25** 229
[13] Anderson E J and Ferris M C 1994 *INFORMS J. Comput.* **6** 161
[14] Bhattacharya M 2013 *Int. J. Adv. Res. Artif. Intell.* **2** 53
[15] Kim H S K H S and Cho S B C S B 2001 *Proceedings of the 2001 Congress on Evolutionary Computation* vol. 2 p 887
[16] Rasheed K 2000 *Proceedings of the 2000 Congress on Evolutionary Computation, CEC00 (Cat. No.00TH8512)* vol. 2
[17] Chen Y, Chen F, Wang W, Ding D and Gao J 2011 *J. Power Sources* **196** 4987
[18] Wang Y, Zhang L, Chen F and Xia C 2012 *Int. J. Hydrog. Energy* **37** 8582
[19] Hautier G, Jain A and Ong S P 2012 *J. Mater. Sci.* **47** 7317
[20] Guo X G, Chen X S, Sun Y L, Sun L Z, Zhou X H and Lu W 2003 *Phys. Lett. A* **317** 501
[21] Zhang X D *et al* 2012 *Comput. Mater. Sci.* **54** 75
[22] Maldonado F, Rivera R and Stashans A 2012 *Phys. B: Condens. Matter* **407** 1262
[23] Li M, Zhang J-Y, Zhang Y and Wang T-M 2012 *Chin. Phys. B* **21** 87301
[24] Zhang J, Liang E-J, Sun Q and Jia Y 2012 *Chin. Phys. B* **21** 47201
[25] Suthirakun S, Ammal S C, Xiao G, Chen F, Zur Loye H-C and Heyden A 2011 *Phys. Rev. B* **84** 205102
[26] Suthirakun S *et al* 2012 *Solid State Ion.* **228** 37
[27] Zhou J *et al* 2012 *Appl. Surf. Sci.* **258** 3133
[28] Shishkin M and Ziegler T 2010 *J. Phys. Chem. C* **114** 21411
[29] Chen H-T, Raghunath P and Lin M C 2011 *Langmuir* **27** 6787
[30] Ahmad E A, Liborio L, Kramer D, Mallia G, Kucernak A R and Harrison N M 2011 *Phys. Rev. B* **84** 85137
[31] An W, Gatewood D, Dunlap B and Turner C H 2011 *J. Power Sources* **196** 4724
[32] Kresse G and Furthmüller J 1996 *Phys. Rev. B* **54** 11169
[33] Jain A *et al* 2013 *APL Mater.* **1** 011002
[34] Perdew J P, Burke K and Ernzerhof M 1996 *Phys. Rev. Lett.* **77** 3865
[35] Blöchl P E 1994 *Phys. Rev. B* **50** 17953