Genetic programming for multitimescale modeling

Kumara Sastry,[1,3] D. D. Johnson,[1,3] David E. Goldberg,[2] and Pascal Bellon[1]
[1]*Department of Materials Science & Engineering, University of Illinois Urbana-Champaign, Urbana, Illinois 61801, USA*
[2]*Department of General Engineering, University of Illinois Urbana-Champaign, Urbana, Illinois 61801, USA*
[3]*Fredrick Seitz Materials Research Laboratory, University of Illinois Urbana-Champaign, Urbana, Illinois 61801, USA*
(Received 22 April 2005; published 16 August 2005)

A bottleneck for multitimescale thermally activated dynamics is the computation of the potential energy surface. We explore the use of genetic programming (GP) to symbolically regress a mapping of the saddle-point barriers from only a few calculated points via molecular dynamics, thereby avoiding explicit calculation of all barriers. The GP-regressed barrier function enables use of kinetic Monte Carlo to simulate real-time kinetics (seconds to hours) based upon realistic atomic interactions. To illustrate the concept, we apply a GP regression to vacancy-assisted migration on a surface of a concentrated binary alloy (from both quantum and empirical potentials) and predict the diffusion barriers within ~0.1% error from 3% (or less) of the barriers. We discuss the significant reduction in CPU time (4 to 7 orders of magnitude), the efficacy of GP over standard regression, e.g., polynomial, and the independence of the method on the type of potential.

DOI: 10.1103/PhysRevB.72.085438 PACS number(s): 68.35.Fx, 02.60.−x, 02.70.Wz, 31.50.−x

I. INTRODUCTION

Molecular dynamics (MD) is extensively used for kinetic modeling of materials. Yet MD methods are limited to nanoseconds of real time, and hence fail to model many processes directly. Recently, several approaches were proposed for multiscaling.[1–11] Methods such as temperature-accelerated dynamics (TAD) (Ref. 2) provide significant acceleration of MD, but they still fall 3–6 orders of magnitude short of real processing times. These methods assume that transition-state theory applies, and concentrate only on infrequent events. An alternative approach to bridge timescales[4] uses kinetic Monte Carlo[12] (KMC) combined with MD by constructing an *a priori* list of events (i.e., "look-up table"). The table look-up KMC yields several orders of magnitude increase in *simulated* time over MD depending on temperature, see later. The table of events is commonly comprised of atomic jumps, but collective motions (or off-lattice jumps), e.g., see Ref. 7, can be added if they have been identified, for instance, by MD. Additionally, tabulating barrier energies from a list of events is a serious limitation. For example, multicomponent alloys have an impossibly large set of barriers, due to configurational dependence, making their tabulation impractical, especially from first principles. An alternative approach is calculating energies "on-the-fly,"[6,13] but it too has serious time limitation (see Fig. 1). Recent developments and limitations of KMC methods are given, e.g., in Ref. 13.

To avoid the need or expense of explicit calculation of all activation barriers—frequent or infrequent—and thereby facilitate an effective hybridization of MD and KMC for multiscale dynamics modeling, we suggest genetic programming (GP)—a genetic algorithm that evolves computer programs—to regress symbolically the potential energy surface (PES) (in the present, nontrivial case, saddle-point barriers only) from a limited set of directly calculated points on the PES using MD via semiempirical, tight-binding, or *ab initio* potentials. Importantly, from only a few calculated barriers relative to the total, GP regression provides an *in-line* barrier function for increasing number of active configurations (or complexity) as a machine-learned replacement to the look-up table approach. A key point is that multiscale modeling requires only relevant (often referred to as coarse-grained) information at the appropriate length or timescales. Hence, only the diffusion barriers are needed for kinetics, not the underlying atomic-scale details; how that information is obtained, direct calculation or machine learning, is not relevant to the scaling, only that the barriers are accurate. Therefore, an accurate GP-regressed PES extends the KMC paradigm, as suggested in Fig. 1, permitting simulation over experimentally relevant time frames, which may not be possible from standard *table look-up* or *on-the-fly* KMC. Interfacing GP with TAD-MD and/or pattern-recognition methods will further extend its applicability, e.g., by finding system-specific mechanisms. We refer to this approach as *symbolically regressed table KMC (sr-KMC)*. Of course, sr-KMC benefits from any advances in KMC methods. In addition, GP-based symbolic regression holds promise in other multiscaling areas, e.g., regressing constitutive rules[14] and chemical reaction pathways, which we are now studying. Also, as

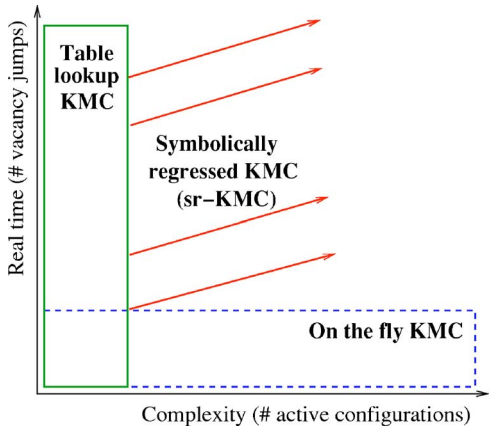

FIG. 1. (Color online) Schematic illustration of simulation capabilities and bottlenecks of on-the-fly KMC, table look-up KMC, and symbolically regressed KMC (sr-KMC).

we exemplify, standard basis-set regression is generally not competitive to GP for fixed accuracy due to the difficulty in choosing (i.e., guessing) appropriate basis functions to represent that space, which are here configurationally dependent diffusion barriers.

Genetic algorithms and programs are used extensively within optimization and machine-learning groups, but to a lesser extent in the physical sciences (e.g., in quantum chemistry studies of clusters). Thus, for completeness, we first give an introduction into the key ideas of GP, which use the concepts of genetic algorithms (GA) for optimization. Importantly, we discuss the key parameters, such as population size of the solution space, to ensure valid results. Hopefully, it is clear that all concepts may migrate to other application areas.

To demonstrate, we discuss GP and its application to a nontrivial case of vacancy-assisted migration on (100) surface of phase-separating Cu_xCo_{1-x} at a concentrated alloy composition, i.e., $x=0.50$. Although there are millions of configurations, only the atoms in the environment locally around vacancy and migrating atom significantly influence the barrier energies. We refer to these as the *active* configurations. The results show that GP predicts barriers within 0.1%–1% error using calculated barriers of less than 3% of the total *active* configurations, depending on the type of potential (error is less for the more accurate potentials, and greater for semiempirical). We also show the efficacy of the GP approach relative to polynomial regression, where the more complex the space, the lower percentage of total barriers is required to regress the potential energy surface, in contrast to standard regression. (Basically, it is too difficult to guess a good basis to fit a complicated PES, but a computer can machine learn it efficiently.) Our initial results hold promise to enable the use of KMC (even with realistic potentials) for increased problem complexity with a scale-up of simulation time.

II. GENETIC PROGRAMMING

To obtain a viable multiscaling method, we require a regression technique that provides an accurate prediction of barrier energies from a very limited set of calculated diffusion barriers. Furthermore, the regression method should be generally applicable regardless of how the data were acquired, as long as the multiply derived data contain roughly similar characteristics, and it should not need to know about the underlying potential. Therefore, the performance of the method should not depend on whether the potential is linear or nonlinear, pairwise or multibody, as long as the potentials used are meaningful.

One such machine-learning method, which not only searches for the optimal regression function but also optimizes the coefficients, is *genetic programming*.[15–25] Genetic programming is a genetic algorithm[26–28] that evolves computer programs. Here, we use GP to regress an in-line barrier function that saves CPU time by requiring fewer number of barrier calculations and by not requiring table look-ups in KMC.

In GP, the in-line barrier function is represented by a tree consisting of functions in the internal nodes and terminals in

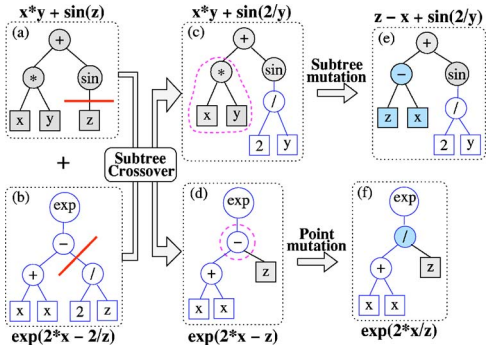

FIG. 2. (Color online) Illustration of tree representation, subtree crossover, subtree mutation, and point mutation used in GP.

the leaf nodes [Fig. 2(a)]. Here, we use the function set $\mathcal{F}=\{+,-,*,/,\hat{},\exp,\sin\}$, where $+$, $-$, $*$, and / are the arithmetic operators addition, subtraction, multiplication, and division, respectively, and $\hat{}$, exp, and sin denote power, exponential, and sine functions, respectively. We used the terminal set $\mathcal{T}=\{\vec{x},\mathcal{R}\}$, where $\vec{x}$ is a vector representing the *active* alloy configuration, and $\mathcal{R}$ is an ephemeral random constant.[16] The ephemeral random constant is a "place holder" for numerical coefficient values. Each time an ephemeral random constant is encountered in a program tree, it is replaced by a random value drawn from a uniform random distribution. An important thing to ensure is that the functions satisfy *closure* property,[16] i.e., each of the functions should be able to accept as its arguments any value or data type taken by a terminal or returned by any function. Once the closure property is satisfied, then new program trees created by crossover and mutation will be syntactically correct and executable. For example, divide by zero is normally defined, but in order to satisfy closure, a divide function must return a valid value even if the divisor is zero. In this paper, division returns 1 if the divisor (second argument) is zero. Thus, all floating point exceptions are trapped and a valid value is returned, which is a common practice in genetic programming approaches.

As we use GP for predicting the diffusion barriers, a *tree* represents a PES-prediction function that takes a configuration and ephemeral constants as inputs and returns the barrier for that configuration as output. A tree's quality is given by its fitness f. For this, we calculate the barriers $\{\Delta E_{\text{calc}}(\vec{x}_1),\ldots,\Delta E_{\text{calc}}(\vec{x}_M)\}$ for M random configurations $\{\vec{x}_1,\vec{x}_2,\ldots,\vec{x}_M\}$. These configurations are used as inputs to the tree, and the barriers $\{\Delta E_{\text{pred}}(\vec{x}_1),\ldots,\Delta E_{\text{pred}}(\vec{x}_M)\}$ are predicted. The fitness is then computed as a weighted average of the absolute error between the predicted and calculated barriers

$$f=\frac{1}{M}\sum_{i=1}^{M} w_i|\Delta E_{\text{pred}}(\vec{x}_i)-\Delta E_{\text{calc}}(\vec{x}_i)|, \tag{1}$$

with $w_i=|\Delta E_{\text{calc}}|^{-1}$, which gives preference to accurately predicting lower energy (most significant diffusional) events. We note that parsimony pressure (or a penalty for unnecessarily complex programs) can also be added into the fitness

measure. Complexity measures such as the number of nodes in the program tree, tree height, tree depth, and model description length metrics have been used to favor simpler programs with good accuracy.[29] While in this study we do not use any parsimony pressure, for solving complex problems, it might not be only helpful, but also essential.

Unlike traditional search methods, GP uses a population of candidate solutions (PES prediction functions) that are initially created using the *ramped half-and-half* method.[16] Once the population is initialized and evaluated, the following genetic operators are repeatedly applied until one or more convergence criteria are satisfied:

Selection: allocates more copies to solutions with better fitness values. We use an *s-wise tournament selection*,[30] where s candidate solutions are randomly chosen and pitted against each other in a tournament. A solution with the best fitness wins.

Recombination combines bits and pieces of two solutions to create new, hopefully better, solutions. We use *subtree crossover*,[16] where a crossover point for each solution is randomly chosen and subtrees below the point are swapped to create two new solutions; see Fig. 2.

Mutation locally but randomly modifies a solution. We use two mutation techniques (see Fig. 2): *Subtree mutation*, where a subtree is randomly replaced with another randomly created subtree, and *point mutation*, where a node is randomly modified.

Parameter setting in GP

Before using GP for symbolically regressing an in-line diffusion-barrier function, or any other evolutionary algorithm, we need to set some parameters such as the population size n, number of generations to run GP t_{conv}, tournament size s, crossover probability p_c, and mutation probability p_m. Researchers who use evolutionary algorithms have oftentimes set these parameters in an *ad hoc* manner. However, many of the parameters in genetic algorithms and genetic programming can be set based on facetwise modeling and dimensional arguments.[28]

A key parameter determining performance of a GP is the population size. For example, small population sizes might lead to premature convergence and yield substandard solutions, whereas large population sizes lead to unnecessary expenditure of valuable computational time. Population size should be large enough to ensure that all the raw substructures—tree segments that are potentially part of the solution to the search problem—are present in the initial population so that recombination and mutation can assemble and fine-tune them to yield high-quality PES predicting functions. The minimum population size, n, required to ensure that at least one copy of different *subcomponents* (or tree fragments) with k_f functionals and k_t terminals is given by[31]

$$n = \frac{1}{\lambda p_{\text{exp}}}(2\chi_f)^{k_f}(2\chi_t)^{k_t}[k_f \ln \chi_f + k_t \ln \chi_t - \ln \epsilon], \qquad (2)$$

where λ is the average tree size in the initial population, p_{exp} is the proportion of the tree that gets *expressed*, i.e., contributes to the fitness or objective value, χ_t is the number of terminals in the terminal set $\mathcal{T}$, χ_f is the number of functions in the function set $\mathcal{F}$, and ϵ is the failure tolerance. Empirically, it has been observed that the subcomponent size, $k = k_f + k_t$, is small and ranges between 1–5. The validity of the model is demonstrated in Fig. 3, where we plot the probability of having at least one copy of all the raw subcomponents, $1-\epsilon$, versus population size. As shown in Fig. 3, as the population size is increased, the probability of initializing at least one copy of all the competing raw subcomponents increases.

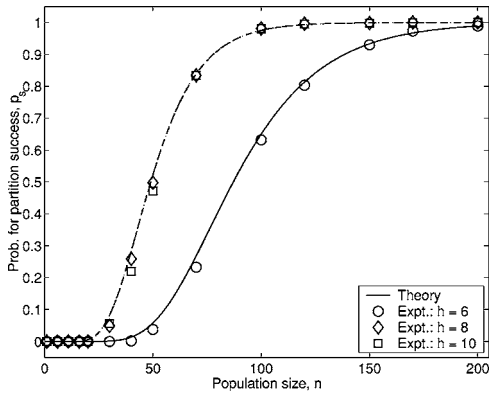

FIG. 3. Probability of at least one copy of all raw subcomponents being present in the population vs population size, n for different tree sizes $\lambda = 2^h$. Empirical results depict the proportion of runs having at least one copy of a primitive and its complement in the population out of 1000 trials.

While ensuring the presence of raw subcomponents is important, it is not always sufficient for ensuring GP success. We need to consider a large enough population that ensures sustained growth of better subcomponents over bad ones. This decision making among different, competing subcomponents is *statistical* in nature, and, as we increase the population size, we increase the likelihood of making the best possible decisions.[32–34] The population sizing required for ensuring sustained growth of good subcomponents is given by[35]

$$n = c\frac{\sigma_{bb}^2}{d^2}\kappa\frac{(c_k m_k - 1)2^{k+1}}{p_{BB}^{\text{expr}}\lambda}. \qquad (3)$$

Each of the terms in the above population-sizing equation is explained in the following paragraphs. In essence, the population sizing for GP consists of the following factors:

(i) *Competition complexity*, quantified by the total number of competing subcomponents, $\kappa = \chi_f^{k_f}\chi_t^{k_t}$.

(ii) *Ease of decision making*, quantified by the signal-to-noise ratio, d^2/σ_{bb}^2, where d is the difference between the mean fitness of candidate solutions containing competing subcomponents, and σ_{bb} is the variance of fitness of the competing subcomponents.

(iii) *Probabilistic safety factor*, quantified by the coefficient $c = z^2(\alpha)$, where $z(\alpha)$ is the ordinate of a unit, one-sided normal deviate, and α is the probability of making an error (choosing the wrong competing subcomponent).

(iv) *Subcomponent complexity*, which depends not only on the minimum number of subcomponents (building blocks) required to solve the problem m_k, but also on the average tree

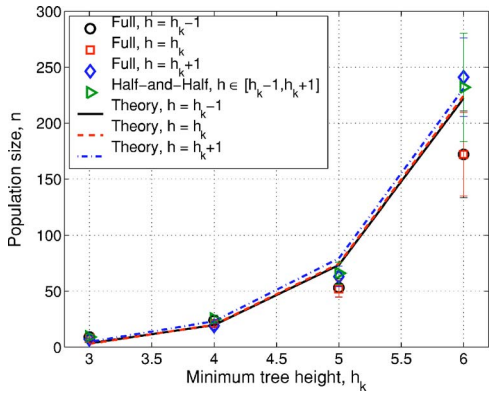

FIG. 4. (Color online) Empirical validation of the population-sizing model [Eq. (3)] for GP. Empirical results denote the minimum population size required to decide correctly between competing substructures with probability $(m_k-1)/m_k$. The tree height h_k equals 2^{m_k} and $\lambda_k=2m_k-1$.

size λ, and the proportion of the tree that gets expressed on an average, $p_{BB}^{\exp}$. Empirical evidence shows that λ is a multiple (usually 3–5 times) of the Kolmogorov complexity of the problem.

The above population-sizing equation is verified with empirical results in Fig. 4. The initial population was randomly generated with either full trees or by the ramped half-and-half method with trees of heights, $h \in [h_k-1, h_k+1]$, where h_k is the minimum tree height with an average of $2m_k$ leaf nodes. We have empirically observed that the population size scales quadratically with Kolmogorov complexity, $n=\mathcal{O}(2^k\lambda_k^2)$.

Another parameter affecting scalability of GP is the expected number of generations required for the population to converge successfully to a high-quality solution. Facetwise analysis of GP shows that the number of generations required to converge to a high-quality solution scales linearly with Kolmogorov complexity, $t_{\text{conv}}=\mathcal{O}(\lambda_k)$.[36,37]

III. CASE STUDY: VACANCY-ASSISTED SURFACE DIFFUSION IN BINARY ALLOY

Via GP-based regression, we consider the prediction of diffusion barriers for vacancy-assisted migration on (100) surface of phase-separating fcc Cu_xCo_{1-x}[38,39] for a concentrated alloy with $x=0.5$, a nontrivial case with many configurations that affect the diffusional barrier height. The system consists of five layers with 100 to 625 atoms in each layer; see Fig. 5. The bottom three layers are held fixed to their bulk bond distances, while the top layers are either held fixed (as a test) or fully relaxed via MD. The input to the barrier regression and prediction function, $\vec{x}=\{x_j\}$, is a binary-encoded vector sequence, where $x_j=0$ (1) represents a Cu (Co) atom. We consider only first and second nearest-neighbor (n.n.) jumps, along with first- (as a test) and second-n.n. environmental atoms in the active configuration, as shown in Fig. 5. This system already exhibits large complexity and is still small enough so that table look-up and GP-regressed KMC can be implemented and directly compared. Table I gives the number of active configurations when first- and second-n.n. environments are considered for a binary alloy.

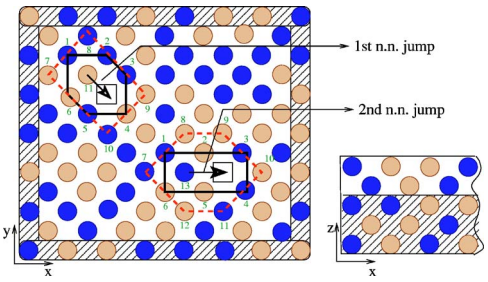

FIG. 5. (Color online) Sketch of simulation cell for vacancy-assisted migration on (100) surface of an fcc binary alloy. Atoms in all but bottom layers and the boundary can fully relax. Solid (dashed) lines around migrating atom and vacancy represent first- (second-) n.n environmental atoms. Atoms for first- (second-) n.n. jumps are labeled from 1–7 (1–13) as they occur in the encoded vector $\vec{x}$ along with the barrier energy $\Delta E(\vec{x})$.

If one were to tackle this problem with *on-the-fly* KMC simulations, it would require the explicit calculations of four activation energies for the four possible jumps to first-nearest-neighbor sites. For most practical cases, this approach is prohibitively expensive in computing time since activation energies depend strongly on the local environment of the vacancy; even with a rather small range of interatomic interactions, this results in activation energy calculations well in excess of a million. There are, however, two exceptions for which on-the-fly calculations are possible. The first one is for very large dilution ($x \ll 1$), and the second is for small systems. For the present demonstration, we consider a small system, so that we can calculate explicitly and exactly all relevant activation energies. This approach will make it possible to compare directly the effectiveness of GP-based KMC to that of on-the-fly KMC simulations. We will discuss below how these results scale as the size of the system increases, an important question for practical applications.

Note that, for the sake of simplicity, we have restricted the dynamics of the atoms to vacancy-assisted jumps. This simplification, however, does not limit the generality of our demonstration. Indeed, as long as the list of possible jumps is known *a priori*, which is a standard requirement for lattice KMC simulations,[7] the present approach can easily be extended. One would simply use one symbolically regressed function for activation energies corresponding to each migration mechanism, for instance, divacancy migration, adatom migration, and atom exchanges at step surfaces.[40]

TABLE I. Number of active configurations for first- and second-n.n. jumps, and for first- and second-n.n. active atoms.

	First-n.n. jumps	Second-n.n. jumps
First-n.n. active configurations	128	128
Second-n.n. active configurations	2048	8192
Total configurations	$\gg 2^{100}$	$\gg 2^{100}$

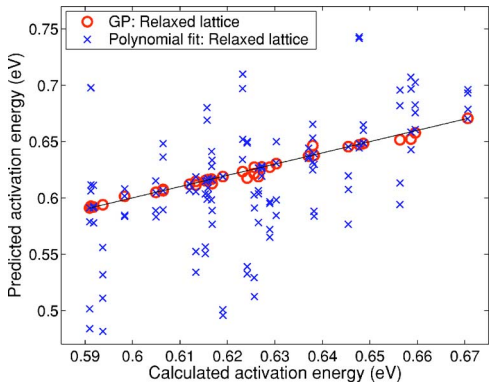

FIG. 6. (Color online) Activation energies (in electron volts) predicted by regression. GP (circles) and a quadratic polynomial (crosses) are compared to the calculated (Morse) barriers for first-n.n. jumps on (100) surface of $Cu_{0.5}Co_{0.5}$ for relaxed lattices. As a simple test, only first-n.n. environments are considered in the active configuration. The line is a guide for the eye.

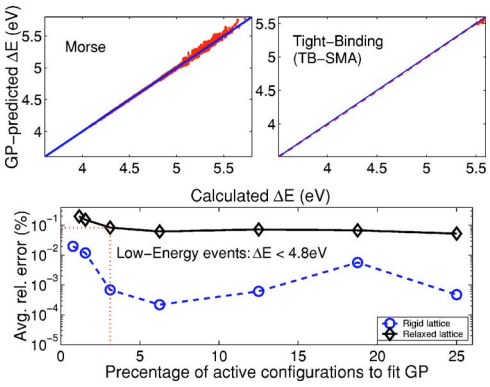

FIG. 7. (Color online) Upper: Calculated vs GP-predicted (Morse and TB-SMA) barriers (in electron volts) for second-n.n. jumps on relaxed $Cu_{0.5}Co_{0.5}(100)$ from configurations out to second-n.n. Lower: Morse GP-barrier error vs number of configurations used in learning set: 0.1% (1%) error for most- ($\Delta E < 4.8$ eV) and least- ($\Delta E > 4.8$ eV) significant events from only 3% of active configurations; TB-SMA GP error is 0.1% for less than 3%.

We model atomic interactions with a Morse potential,[41] which includes at least second-n.n. interactions, and a tight-binding potential within a second-moment approximation (TB-SMA).[42–47] The atomic interactions of TB-SMA range over fifth-nearest neighbors, which are longer range and, hence, more computationally intensive (as in timings given later) but more accurate.[48] For the TB-SMA with interactions up to fifth-n.n. atoms, we only consider up to second-n.n. environmental atoms as variables for GP regression of the barrier function (which is just to minimize the large variable space). If the TB-SMA potential were truncated to second-n.n. interactions, its timings would be approximately equal to those of the Morse potential, but it requires additional terms to ensure continuity of potential, no truncation forces, etc.[48] To validate interactions, we model vacancy-assisted migration on (100) surface of Cu and consider only first-n.n. jumps. The predicted barrier for n.n. vacancy jumps with fully relaxed lattice in Cu is 0.39 eV for Morse and 0.45 eV for TB-SMA, agreeing with 0.42 ± 0.08 (0.47 ± 0.05) from *ab initio* (EAM) (Ref. 46) calculations.

A. Efficacy of GP regression

For simplicity, we begin by considering only seven surface first-n.n. environmental atoms, i.e., six neighboring atoms of both the diffusing atom and vacancy—for a total of seven atoms that can be either Cu or Co, yielding 128 *active* configurations. The environment outside this configuration is fixed. About 20, i.e., 16%, different active configurations, are randomly chosen and their barriers are computed using the conjugate-gradient method and are used in the GP fitness function; see Eq. (1). The barriers predicted by GP for the relaxed configurations are compared to the exact values in Fig. 6. We note that the prediction error for the rigid lattice case ($0.4 \pm 0.04\%$) is significantly less than that for the relaxed lattice case ($2.8 \pm 0.08\%$). Due to the weighting used in the fitness function, GP predicts barriers for most significant, low-energy events more accurately than for less significant, higher-energy events. The *in-line barrier function* symbolically regressed via GP is simple

$$f_{\text{barrier}}(\vec{x}) = x_1 - 2x_2 + x_3 + x_4 - 2x_5 + x_7 - \frac{x_7}{x_6} - g_1(\vec{x}) - g_2(\vec{x}) + g_3(\vec{x}), \quad (4)$$

where, once again, $\vec{x}=\{x_1,x_2,x_3,x_4,x_5,x_6,x_7\}$, and x_i is either 0 or 1 denoting a Cu or Co atom, but the functions $g_i(\vec{x})$ are highly nonlinear; see the Appendix.

Figure 6 also compares the barriers predicted by GP to those predicted by a least-squares-fit quadratic polynomial, showing clearly its inadequacy for alloys. Furthermore, while GP requires only 16%, the quadratic (cubic) polynomial fit needs 28% (78%) of the barriers. [For clarity, the large percentage needed for the polynomial regression arises because of the number of variables. In the quadratic case for this simple test case, the variable $\vec{x}$ has 7 occupation components of 0 or 1, so we need to fit coefficient terms from *1 constant, 7 linear*, and *7 (21) (off-)diagonal quadratic*, for 36/128 or 28%.] In limited cases, such as dilute $Fe_{1-x}Cu_x$, the barriers can be predicted via a simple polynomial fit.[50]

To test the scalability of GP with *active* configuration size, we consider the second-n.n. jumps and first- and second-n.n. environmental atoms in the *active* configuration. As shown in Table I, there is a total of 8192 configurations. The energies predicted by GP are compared with direct calculations in Fig. 7, along with the error in the Morse (worst case) example. The GP predicts the barriers for most significant events with less than 0.1% error by fitting to energies from only 3% (i.e., 256/8192) of the active configurations (see the error definition in Ref. 51). From Fig. 7, clearly the nonadditive and nonlinear tight-binding potential has less error than the Morse case for even fewer barriers in the learning set. In comparison, a cubic polynomial fit requires energies for ~6% of the configurations, predicting the barriers with 2.5% error for the most significant events.

The results shown in Figs. 6 and 7 clearly demonstrate the effectiveness of GP in predicting the potential energy surface, with high accuracy and little information. As expected, since the regression and barrier calculation are nearly independent, the GP performance does not depend on the potentials used, e.g., Fig. 7 shows results for both Morse, and nonadditive and nonlinear tight-binding potentials. The regression only requires a database of barriers and has no knowledge (nor the need) of the underlying potential used. We also find that the GP performance is independent of the configuration set used in calculating the fitness function, the order in which they are used, and the labeling scheme used to convert the configuration into a vector of inputs. Differences in activation-energy scale on the PES prediction via GP are also negligible. That is, even though the barriers for the first- and second-n.n. jumps differ by an order of magnitude, GP predicts the barriers with similar accuracy. Moreover, for more complex, cooperative effects, such as island diffusion via surface dislocations,[52] sr-KMC could be interfaced with pattern-recognition methods,[53] as well as long-range fields.[54]

B. Relative time comparisons

The CPU time savings by coupling GP-regressed *in-line barrier function* with KMC (i.e., sr-KMC) are simple to estimate. For our example, with ~33 times fewer calculated barriers GP symbolically regresses an in-line barrier function—rather than the complete look-up table—and thus, sr-KMC provides a direct CPU savings of ~100 over table look-up methods. Additionally, each sr-KMC time step requires only 10^{-3} CPU seconds for an *in-line function* evaluation, as opposed to *on-the-fly* KMC that requires seconds (empirical potentials) to hours (quantum methods), providing a gain of 10^4–10^7 CPU seconds. For our example, one relaxed barrier calculation takes ~10 s (~1800 s) for Morse (tight-binding).

An important question, especially for bulk diffusion, is how the gain from sr-KMC scales with system complexity (e.g., range of environment considered—the active environment—or additional alloying components). While we cannot fully answer this question yet, in the present study it is remarkable and promising that the fraction of explicit barrier calculations required by sr-KMC decreases as the number of active configurations increases.

For sake of completeness, we note the simulation time enhancements over MD (from nanoseconds to seconds) by sr-KMC. (Of course, if a complete *look-up* table is also calculated, the estimate is the same.) With event occurrence following a Poisson distribution, the real time in KMC is given by[12,55]

$$\tau_r = \sum_{j=1}^{N_{KMC}} \frac{-\ln U}{\sum_{i=1}^{N_{cfgs}} \nu_0 e^{-\beta \Delta E(\vec{x}_i)}}, \tag{5}$$

where N_{KMC} is the number of Monte Carlo steps, $U \in (0,1]$ is a uniform random number, N_{cfgs} is the number of active configurations. Using $\nu_0 \approx 27 \times 10^{12}$ Hz for Cu-Co,[49] Eq. (5) gives—per time step of KMC relative to MD (assuming an MD time step of 10^{-15} s)—an increase in *simulated* time of 10^9 at 300 K, 10^4 at 650 K, and $10^{2.3}$ at 1000 K. Direct timing runs from KMC agree with these estimates. So, the key increase in timings comes from learning the table from very few barriers, saving all the calculating time, and allowing more complex problems to be addressed potentially.

IV. SUMMARY AND CONCLUSIONS

Here, we have presented an approach using a machine-learning method based upon symbolic regression via genetic programming to determine, accurately, and with little information, complete details of the potential energy surface and output as an *in-line* function. We have shown, on a nontrivial example of vacancy-assisted migration on a surface of fcc Cu_xCo_{1-x}, that GP predicts all barriers with 0.1% error from calculations for less than 3% of active configurations, independent of the type of potentials used to obtain the learning set of barriers via molecular dynamics. The genetic programming-based KMC approach avoids the need or expense of calculating the entire potential-energy surface, is highly accurate, and leads to significant scale-up in real simulation time for complex cases as it enables use of KMC and, more importantly, leads to a significant reduction in CPU time needed for KMC (>7 orders of magnitude for quantum-based calculations), not possible by any other current means. For alloys, we believe the number of explicit barrier calculations for the learning set can be reduced further by over an order of magnitude (~0.3% of the *active* configurations) using local cluster expansion methods.[56]

The genetic programming regression allows atomic-scale information (in our example, diffusion barriers on the potential energy surface) to be included in a long-time kinetic simulation without maintaining a detailed description of the all-atomistic physics, as done within molecular dynamics. Our multiscale approach does not require finding pertinent "hidden variables," but just uses necessary information at the appropriate timescale (or length scale)—a coarse graining of sorts. We emphasize that the genetic programming is nontrivially regressing an *in-line* function and its coefficients that approximate the potential-energy surface, and its efficacy over standard basis-set regression was made clear. Moreover, the genetic programming approach is not problem specific and requires little modification, if any (say, by choice of operators and functions), to address increasingly complex cases, and, thus holds promise as an efficient tool for multiscaling in more than one area, as suggested.

ACKNOWLEDGMENTS

This work was supported by the NSF at Materials Computation Center (ITR Grant DMR-03-25939), CPSD (ITR Grant DMR-0121695), AFOSR (Grant F49620-00-0163), and the Department of Energy through the Fredrick Seitz MRL (Grant DEFG02-91ER45439) at UIUC. We gratefully acknowledge discussions with R.S. Averback, Y. Ashkenazy, and J.-M. Roussel.

APPENDIX: GP-REGRESSED IN-LINE BARRIER FUNCTION

The inline barrier function f_{barrier} symbolically regressed via GP for second-n.n. jumps considering only seven surface first-n.n. environmental atoms (our initial simple case) is given by

$$f_{\text{barrier}}(\vec{x}) = x_1 - 2x_2 + x_3 + x_4 - 2x_5 + x_7 - \frac{x_7}{x_6} - g_1 - g_2 + g_3, \tag{A1}$$

where $\vec{x}=\{x_1,x_2,x_3,x_4,x_5,x_6,x_7\}$, and x_i is either 0 or 1 denoting a Cu or Co atom, and

$$g_1 = \frac{1}{x_5}\left[x_4 + \frac{x_5}{(x_1 + x_4 + 2x_5 + 40x_7)}\right]^{0.025},$$

$$g_2 = \frac{-g_4 \times [x_2 + 0.25x_5]^{(g_7 x_6^{g_6(1+x_3)})}}{[x_1 + x_3 + x_4 + 0.945 \times x_4 x_6/x_7]},$$

$$g_3 = 40x_2[x_2^{(1+x_2+2x_3+x_5+g_5)}]. \tag{A2}$$

See Sec. II for handling of floating point and other expectations that might occur in the symbolically regressed barrier functions. The functions $g_i(\vec{x})$ are highly nonlinear functions of the configurations, i.e.,

$$g_4 = g_8\left[\frac{0.473 \times x_3 x_5 (x_2^{(x_7/x_4)})^{2x_3}}{0.177 \times x_1 x_6 g_9 (x_1 + x_4)(x_2 + x_5)}\right],$$

$$g_5 = \frac{(x_2 + 0.025x_5)^{g_{10}} \times g_{11}}{x_2 + x_3 + x_4 + g_{12}},$$

$$g_6 = x_2 + \frac{x_5\left[2(x_2 + x_5) + \frac{x_7}{x_6} + 40\right]}{x_5 + x_3\left[2(x_2 + x_5) + \frac{x_7}{x_6} + 40\right]},$$

$$g_7 = (g_{13} + g_{14})[x_2^{(x_2+x_5/(x_2+x_5+x_4))}]. \tag{A3}$$

$$g_8 = [(x_2^{g_{14}} + g_{17})(x_2^{(x_2+g_{18})} + g_{17})]^{0.025},$$

$$g_9 = \frac{x_6(0.473 + x_6)}{x_4 (x_2^{g_{19}})^{x_5/x_7}},$$

$$g_{10} = g_{15} x_2^{g_{20}(2x_2x_5 + x_2x_4x_6 + x_5)},$$

$$g_{11} = \frac{3x_3x_5x_2^{(2x_2)}}{g_{21}\left[x_3 - 1 - \frac{x_6}{x_7} + (0.59 - x_1)(x_5 - x_6)\right]},$$

$$g_{12} = x_1 + x_2 + 3x_3 + x_4 + x_5 + x_2^{2(x_2+x_5)} + 0.95\frac{x_4x_6}{x_7} + (0.59 - x_1)(x_5 - x_6) + \frac{x_7}{x_6 - 0.23}. \tag{A4}$$

$$g_{13} = x_2 + \frac{x_5}{0.473 + x_1 + x_3 + x_4 + x_2^{g_{22}}},$$

$$g_{14} = x_2 + \frac{x_5[0.473 + x_2^{g_{23}}]}{x_5 + x_2[0.473 + x_2^{g_{23}}]},$$

$$g_{15} = 1 + x_2 + x_5 + \frac{x_5}{x_6} + \frac{x_7}{x_4} + \frac{x_5}{x_1 + x_4} + g_{16}^{(x_5/x_7)} + x_2^{[x_3 - 1 + (x_6/x_5) - (x_6/x_7) + (x_6/x_2)]},$$

$$g_{16} = x_2^{[-1 - x_1 + x_3 - x_4 + (x_6/x_7) + (x_6/x_5) - g_{24}]},$$

$$g_{17} = \frac{x_5[0.473 + 40x_7]}{x_5 + x_2[0.473 + 40x_7]}. \tag{A5}$$

$$g_{18} = \frac{x_5[0.473 + x_2^{(x_2+x_5)}]}{x_5 + (x_2 + x_5)[0.473 + x_2^{(x_2+x_5)}]},$$

$$g_{19} = \left[-x_1 + x_3 - x_4 + \frac{x_6}{x_5} - g_{24}\right]^{(x_5/x_7)},$$

$$g_{20} = \frac{[x_5 + x_7/x_6]^{(x_2+x_5)}}{x_2x_5 + x_5 + x_4x_6},$$

$$g_{21} = x_6(x_2 + x_5)\left[x_3 + \frac{x_5}{x_2 + x_5 + \frac{x_7}{x_6} + g_{25}}\right]. \tag{A6}$$

$$g_{22} = \frac{x_6}{x_7} + \frac{x_6}{x_5} - x_1 - x_3 - x_4 - \left(\frac{x_2}{x_1}\right)^{[(x_6/x_3) - 0.47(x_2+x_5)]},$$

$$g_{23} = \left[x_3 + \frac{x_6}{x_5}\right]^{(3x_3 + x_7/x_6)},$$

$$g_{24} = \left(\frac{x_2}{x_1}\right)^{[(x_6/x_3) + 0.473(x_2+x_5)]},$$

$$g_{25} = 0.47 + \frac{x_5}{x_2 + x_5 + \frac{x_7}{x_6} + \frac{1}{x_2 + x_5} + \frac{x_4}{0.473 + x_6}}. \tag{A7}$$

[1] A. F. Voter, Phys. Rev. Lett. **78**, 3908 (1997).
[2] A. F. Voter, Phys. Rev. B **57**, R13985 (1998).
[3] A. F. Voter, F. Montalenti, and T. C. Germann, Annu. Rev. Mater. Res. **32**, 321 (2002).
[4] J. Jacobsen, B. H. Cooper, and J. P. Sethna, Phys. Rev. B **58**, 15847 (1998).
[5] T. Diaz De La Rubia, M. J. Caturla, E. Alonso, M. J. Fluss, and J. M. Perlado, J. Comput.-Aided Mater. Des. **5**, 243 (1998).
[6] G. Henkelman and H. Jónsson, J. Chem. Phys. **111**, 7010 (1999), **113**, 9978 (2000); **115**, 9657 (2001).
[7] M. R. Sørensen and A. F. Voter, J. Chem. Phys. **112**, 9599 (2000).
[8] W. Cai, M. H. Kalos, M. de Koning, and V. V. Bulatov, Phys. Rev. E **66**, 046703 (2002).
[9] M. M. Steiner, P.-A. Genilloud, and J. W. Wilkins, Phys. Rev. B **57**, 10236 (1998).
[10] G. T. Barkema and N. Mousseau, Phys. Rev. Lett. **77**, 4358 (1996).
[11] G. T. Barkema and N. Mousseau, Comput. Mater. Sci. **20**, 285 (2001).
[12] *Monte Carlo Methods in Statistical Physics*, edited by K. Binder (Springer, Berlin, 1986).
[13] J. L. Bocquet, Defect Diffus. Forum **203–205**, 81 (2002).
[14] Kumara Sastry, D. D. Johnson, D. E. Goldberg, and P. Bellon, Int. J. Multiscale Comp. Eng. **2**, 239 (2004).
[15] J. R. Koza, *Proceedings of the 11th International Joint Conference on Artificial Intelligence* (Morgan Kaufmann, San Francisco, CA, 1989), Vol. 1, p. 768.
[16] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection* (MIT Press, Cambridge, MA, 1992).
[17] J. R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs* (MIT Press, Cambridge, MA, 1994).
[18] J. R. Koza, F. H. Bennett III, D. Andre, and M. A. Keane, *Genetic Programming III: Darwinian Invention and Problem Solving* (Morgan Kaufmann, San Francisco, CA, 1999).
[19] J. R. Koza, M. A. Keane, M. J. Streeter, W. Mydlowec, J. Yu, and G. Lanza, *Genetic Programming IV: Routine Human-Competitive Machine Intelligence* (Kluwer Academic Publishers, Boston, MA, 2003).
[20] V. Babovic and M. Keijzer, J. Hydroinformatics **2**, 35 (2000).
[21] W. Banzhaf, P. Nordin, R. Keller, and F. Francone, *Genetic Programming—An Introduction on the Automatic Evolution of Computer Programs and its Applications* (Morgan Kaufmann, San Francisco, CA, 1998).
[22] C. Ryan, J. Collins, and M. O'Neill, in *Proceedings of the EuroGP Conference* (Springer-Verlag, New York, 1998), Vol. 1391, p. 83.
[23] A. Ratle and M. Sebag, Appl. Soft Comput. **1**, 105 (2001).
[24] D. J. Montana, Evol. Comput.3, 199 (1995) (Also BBN Technical Report No. 7866).
[25] T. D. Haynes, D. A. Schoenfeld, and R. L. Wainwright, in *Advances in Genetic Programming 2*, edited by P. J. Angeline and K. E. Kinnear (MIT Press, Cambridge, MA, 1996), Chap. 18, pp. 359–376.
[26] J. H. Holland, *Adaptation in Natural and Artificial Systems* (University of Michigan Press, Ann Arbor, MI, 1975).
[27] D. E. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning* (Addison-Wesley, Reading, MA, 1989).
[28] D. E. Goldberg, *Design Of Innovation: Lessons from and for Competent Genetic Algorithms* (Kluwer Acadamic Publishers, Boston, MA, 2002).
[29] W. B. Langdon, *Genetic Programming and Data Structures: Genetic Programming + Data Structures = Automatic Programming*! (Kluwer Academic Publishers, Boston, MA, 1998), Chap. 1, pp. 30–32.
[30] D. E. Goldberg, B. Korb, and K. Deb, Complex Syst. **3**, 493 (1989).
[31] K. Sastry, U.-M. O'Reilly, D. Goldberg, and D. Hill, in *Genetic Programming Theory and Practice*, edited by R. Riolo and B. Worzel (Kluwer Academic Publishers, Boston, MA, 2003), Chap. 9, pp. 155–172.
[32] K. A. De Jong, Ph.D. thesis, University of Michigan, 1975 (University Microfilms No. 76–9381).
[33] D. E. Goldberg and M. Rudnick, Complex Syst. 5, 265 (1991) (Also IlliGAL Report No. 91001).
[34] D. E. Goldberg, K. Deb, and J. H. Clark, Complex Syst. 6, 333 (1992) (Also IlliGAL Report No. 91010).
[35] K. Sastry, U.-M. O'Reilly, and D. E. Goldberg, in *Genetic Programming Theory and Practice II*, edited by U.-M. O'Reilly, T. Yu, R. Riolo, and B. Worzel (Kluwer Academic Publishers, Boston, MA, 2004), Chap. 4, pp. 49–66.
[36] D. E. Goldberg and U.-M. O'Reilly, in *Proceedings of the First European Workshop on Genetic Programming*, edited by W. Banzhaf, R. Poli, M. Schoenauer, and T. C. Fogarty (Springer-Verlag, Paris, 1998), Vol. 1391 of *LNCS*, pp. 16–36; URL: ⟨http://www.ai.mit.edu/people/unamay/papers/eurogp.final.ps⟩
[37] U.-M. O'Reilly and D. E. Goldberg, in *Genetic Programming 1998: Proceedings of the Third Annual Conference*, edited by J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, and R. Riolo (Morgan Kaufmann, University of Wisconsin, Madison, WI, 1998), pp. 269–277; URL: ⟨http://www.ai.mit.edu/people/unamay/papers/timing-final.ps⟩
[38] We note that Cu-Co alloys are characterized by a small size misfit (Ref. 39), and the effectiveness of the GP-based symbolic regression for Au-Ni or Cu-Ag alloys, which have a much larger size misfit, is left as a future study.
[39] H. W. King, J. Mater. Sci. **1**, 79 (1966).
[40] Although divacancy diffusion is faster than single-vacancy diffusion in metals, a divacancy will have a shorter lifetime due to reaching vacancy sinks more quickly; so, single-vacancy diffusion is relevant to long-time diffusion, even in our nontrivial case study.
[41] L. Girifalco and V. Weizer, Phys. Rev. **114**, 687 (1959).
[42] N. A. Levanov, V. S. Stepanyuk, W. Hergert, D. I. Bazhanov, P. H. Dederichs, A. A. Katsnelson, and C. Massobrio, Phys. Rev. B **61**, 2230 (2000).
[43] F. Cleri and V. Rosato, Phys. Rev. B **48**, 22 (1993).
[44] G. Mazzone, V. Rosato, M. Pintore, F. Delogu, P. F. Demontis, and G. B. Suffritti, Phys. Rev. B **55**, 837 (1997).
[45] V. S. Stepanyuk, D. I. Bazhanov, A. N. Baranov, W. Hergert, P. H. Dederichs, and J. Kirschner, Phys. Rev. B **62**, 15398 (2000).
[46] V. S. Stepanyuk, D. I. Bazhanov, W. Hergert, and J. Kirschner, Phys. Rev. B **63**, 153406 (2001).
[47] V. S. Stepanyuk, D. V. Tsivline, D. I. Bazhanov, W. Hergert, and A. A. Katsnelson, Phys. Rev. B **63**, 235406 (2001).
[48] J.-M. Roussel (private communication).
[49] G. Boisvert and L. J. Lewis, Phys. Rev. B **56**, 7643 (1997).
[50] Y. Le Bouar and F. Soisson, Phys. Rev. B **65**, 094103 (2002).

[51] The average relative error for N'_{cfgs} configurations within the desired energy range is given by $\bar{\varepsilon}_{\mathrm{rel}}=100/N'_{\mathrm{cfgs}}\Sigma_{i=1}^{N'_{\mathrm{cfgs}}}|\Delta E_{\mathrm{pred}}(\vec{x}_i)-\Delta E_{\mathrm{calc}}(\vec{x}_i)/\Delta E_{\mathrm{calc}}(\vec{x}_i)|$.

[52] J. C. Hamilton, M. S. Daw, and S. M. Foiles, Phys. Rev. Lett. **74**, 2760 (1995).

[53] Talat Rahman (private communication).

[54] For long-range fields (e.g., elastic fields from coherent interfaces, such as multilayers or precipitates), a description based solely on local configurations may have to be extended, say, with phase field methods.

[55] K. A. Fichthorn and W. H. Weinberg, J. Chem. Phys. **95**, 1090 (1991).

[56] A. Van der Ven, G. Ceder, M. Asta, and P. D. Tepesch, Phys. Rev. B **64**, 184307 (2001).