Computer Physics Communications 193 (2015) 66–71

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

A GPU enhanced approach to identify atomic vacancies in solid materials

Joaquín Peralta [a,*], Claudia Loyola [a], Sergio Davis [b]

[a] Departamento de Física, Facultad de Ciencias Exactas, Universidad Andrés Bello, Chile
[b] Departamento de Física, Facultad de Ciencias, Universidad de Chile, Chile

ARTICLE INFO

Article history:
Received 10 October 2014
Received in revised form
27 March 2015
Accepted 31 March 2015
Available online 11 April 2015

Keywords:
Atomic vacancy
GPU
Crystal

ABSTRACT

Identification of vacancies in atomic structures plays a crucial role in the characterization of a material, from structural to dynamical properties. In this work we introduce a computationally improved vacancy recognition technique, based in a previous developed search algorithm. The procedure is highly parallel, based in the use of Graphics Processing Unit (GPU), taking advantage of parallel random number generation as well as the use of a large amount of simultaneous threads as available in GPU architecture. This increases the spatial resolution in the sample and the speed during the process of identification of atomic vacancies. The results show an improvement of efficiency up to two orders of magnitude compared to a single CPU. Along with the above a reduction of required parameters with respect to the original algorithm is presented. We show that only the lattice constant and a tunable overlap parameter are enough as input parameters, and that they are also highly related. A study of those parameters is presented, suggesting how the parameter choice must be addressed.

1. Introduction

Materials that contain defects and impurities can exhibit some of the most scientifically interesting and economically important phenomena known. The nature of disorder in solids is a vast subject. The smallest degree of disorder that can be introduced into a perfect crystal is a point defect, of which the most common type are the vacancies. A vacancy forms when an atom is missing from its expected lattice site. Nowadays there is no well-established procedure to identify a vacancy in a structure snapshot provided by computational simulation techniques. It has been shown that the identification of vacancies is fundamental to understand different materials properties, such as: (i) electronic and mechanical behavior due to the presence of vacancies in oxide/inter-metallic alloy interfaces [1,2]; (ii) the relevance of their migration near to the melting temperature [3,4], which provide a relevant information on the melting process; and (iii) collapse of crystals, where simulations suggest a strong connection with ring-like atomic movement, due to the vacancies [5,6], among others. A previous work of Davis et al. [7] gives us a complete and well-guided process to identify a vacancy in a crystalline or amorphous structure, by the use of virtual spheres [8].

This work proposes two main branches: first to redesign the algorithm by the use of GPU architecture which will improve computational efficiency; and second, to reduce the number of input parameters, where the original method [7] calls at least three different parameters to be tuned independently.

The GPU architecture is a scheme based in multi-threading, where each thread could be slower than a regular CPU, but the advantage relays in the large number of simultaneous threads that a GPU is capable to execute. Albeit the CPU has been widely used as basis for high performance computing in the last decades, this is clearly changing [9]. The above is mainly due to the cost, since the use of commodity-scale processors in supercomputing clusters is considerably more expensive that the price compared with a GPU (or a hybrid CPU + GPU system), which is also more amenable to general-purpose computations than 10 years ago. Along with this transition, comes the need to adapt the computer codes to these cache-based systems, because otherwise and due to their different memory access scheme, the vector-machine codes performance will be poor. Nowadays a large number of codes are being migrated to GPU or hybrid systems [10], this means they are being modified or in some cases rewritten from scratch.

In this work we will use the NVIDIA CUDA [11] version 6.0 with double precision floating point arithmetics, to generate the

* Corresponding author.
E-mail addresses: joaquin.peralta@unab.cl (J. Peralta), claudia.loyola@unab.cl (C. Loyola), sdavis@gnm.cl (S. Davis).

http://dx.doi.org/10.1016/j.cpc.2015.03.022
0010-4655/

GPU code. A good performance scaling is expected when it is possible to split the calculations in different sub-tasks, and assign these to each GPU thread. In this particular case, the algorithm is parallelizable mainly because we can simultaneously analyze different regions of space and thus perform the vacancy search faster. Our technique will allow us to improve the analysis time. On the other hand, we eliminate one tunable parameter and relate the others in a general way, based completely on the unit cell of the material under study.

The work is arranged as follows: Section 2 describes the algorithm step by step; Section 3 shows detailed information of the material used to test the algorithm and the GPU and CPU description; Section 4 presents the results for different stages of the process: parameter fitting, comparison of the results, and speed-up tests; a final discussion of the results and scopes of the present work are shown in Section 5.

2. The search and fill algorithm

The recently developed search and fill (SF) algorithm [7] provide an atomic vacancies recognition, based in the incorporation of *virtual spheres* (VS), which are overlapped with the atomic neighborhood. In this method, each atom is considered as a sphere, with the same radius as the VS. First, the total spatial overlap of each atom with its atomic neighborhood is evaluated. For the atoms with lower overlap, a test vacancy site is taken randomly around them, then the total spatial overlap between the VS, centered in the test vacancy site, and the atomic neighborhood is evaluated. If the total overlap value is below the tolerance parameter, it will be considered as a vacancy site, and an "empty sphere" will be placed. Otherwise, the VS is randomly displaced a finite number of times using a fictitious temperature (simulated annealing minimization method) in order to minimize local overlap, until a vacancy is found or the maximum number of iteration is completed. Unfortunately the SF technique is not directly parallelizable because in the simulated annealing minimization there is a dependency of the actual step with the previous one. It is possible to use more elaborate techniques such as parallel tempering to run in CPU or GPU, but the alternative proposed in this work is more simple to implement, computationally efficient and affordable for large systems. In spite of the above, in Section 4.4 we will present the differences between the running times of the SF and the proposed technique.

In this work we keep the use of the VS as the main component in the atomic vacancy search. However the spatial mapping of the neighborhood of this VS is analyzed using GPU-based techniques. The use of the GPU allows a more efficient analysis for a considerable large number of VS (and possible vacancies) in the search space. The proposed technique GPU SF (GSF) will allow us to avoid the use of additional refinements, such as simulated annealing minimization, which also bring additional parameters to be fit tuned by hand. The overlap function used in this work is given by:

$$f(r/R_0) = 1 - 0.75(r/R_0) + 0.0625(r/R_0)^3, \quad (1)$$

where R_0 correspond to the VS radius, and must be specifically tuned to the problem. The $f(r/R_0)$ function is restricted to $0 \leq r \leq 2R_0$, which implies than an overlap contribution of each neighbor atom is between 0 and 1. The overall performance of the algorithm is not dependent on the exact choice of the overlap function. With this definition in mind, we will proceed to describe the algorithm below.

As a first step, the algorithm determines the atomic overlap of each atom in the structure associated to its neighborhood, defined by $r < 2R_0$, using a single GPU thread for each. A sorting process, from lower to higher overlap, is carried out on the atoms. For each atom, we build a three dimensional cube around it of side $2R_0$. A large number of random points, n, are generated homogeneously inside that cube, using the `curandGenerateUniformDouble` method of the CUDA CURAND libraries.[1] Here a larger value of n will not significantly improve the accuracy of the vacancy found, and just will reduce the performance of the search process, as we will discuss later.

Table 1
For each crystal structure are presented the distance of the first and second neighbors and their respective number of neighbors, the virtual sphere radius and R_2/R_1.

Crystal	R_1	N_1	R_2	N_2	R_0	R_2/R_1
FCC	$a\sqrt{2}/2$	12	a	6	$a(\sqrt{2}+2)/8$	$\sqrt{2}$
BCC	$a\sqrt{3}/2$	8	a	6	$a(\sqrt{3}+2)/8$	$2\sqrt{3}/3$
SC	a	6	$a\sqrt{2}$	12	$a(\sqrt{2}+1)/4$	$\sqrt{2}$

Once the total overlap is determined for each random point (at each GPU thread), we search for the minimum overlap. Based in a criteria value, f_{ovp} (Eq. (2)), the minimum overlap could or could not be designated as a vacancy. Once the point is recognized as a vacancy, a single VS is located in that position, to avoid multi-vacancies overlap. The overlap value of *every atom* in the structure is updated once a vacancy is found.

Finally, recognition of a VS as a vacancy will be directly related to the choice of f_{ovp} as well as the value of R_0. This will remove the requirement of a fictitious temperature or any other additional parameter, as in the original algorithm [7]. The GSF algorithm is briefly summarized as follows:

1. Sort the atoms of the original structure by their total overlap value.
2. Build a cubic structure around each atom, starting for the atom with minimum total overlap value to the maximum, and generate uniformly distributed random points inside.
3. Each point is considered as a virtual sphere with radius R_0.
4. Using a large number of GPU threads evaluate the total overlap for each random point.
5. Search and find the minimum total overlap value of the set of random points.
6. Identify if the point is a vacancy comparing its total overlap value with the tolerance value, f_{ovp}.

In what follows, we present guidelines that could help determine the necessary parameters to find vacancies in a solid material.

2.1. The parameters

The choice of R_0, is suggested to be considered at some value between $R_1/2$ and $R_2/2$, that correspond to the half of the distances to the first and second neighbor respectively. A value of R_0 close to $R_1/2$ will give us a larger number of vacancies, because it becomes most probable to find a lower overlap in some spatial points. On the other hand, if R_0 is close to $R_2/2$ the number of vacancies will be reduced because the total overlap will include partial contribution provided by second neighbors. An initial reasonable value for R_0, used here, is the average value between the mentioned limits, $R_0 = (R_1 + R_2)/4$. Table 1 presents some values of R_0 for typical crystalline structures. All the following analysis will use the value of $a = 3.1652$ Å. The effect of different choice of R_0 will be presented in Section 4.

Once R_0 is chosen, the determination of the appropriate value of f_{ovp} is not straightforward. For a perfect crystalline structure, the expected total overlap value of a missing atom is exactly the sum of the overlap function with the closest neighbors. As an example, a BCC crystalline structure is composed by 8 nearest neighbors; if we remove an atom, the virtual sphere will have a total

[1] https://www.clear.rice.edu/comp422/resources/cuda/html/curand/index.html.

overlap value of $8 \times f(r/R_0)$, with $r = R_1$. Any value less than or equal to this, will be considered a vacancy. Particularly, if we use $R_0 = (R_1 + R_2)/4$ and $r = R_1$, the total overlap value will be equal to $f_{ovp} = N \cdot f(4/(1 + R_2/R_1))$, where N is the number of first nearest neighbors, and any point with a total overlap value minor or equal to f_{ovp} must be considered as a vacancy. Despite the above, the structure is not always related to a perfect crystal, because atoms are vibrating around an equilibrium position at finite temperature. For those cases, the value of $r = R_1$ must be modified. Thus, the right value of f_{ovp} is always related to the particular study case. In order to not introduce additional external parameters, we will use the same $r = R_1$ value, but with an additional percentage of displacement, x. Using this simple approach to choose this parameter, we are able to define:

$$f_{ovp} = N \cdot f\left(\frac{4(1 - x/100)}{1 + R_2/R_1}\right), \tag{2}$$

with R_1, and R_2 as the first and second neighbor distance respectively, N the number of first nearest neighbors, $R_0 = (R_1 + R_2)/4$ and $r = R_1 - xR_1/100$. The value of x is an estimate of the percentage of displacement of the atoms from their equilibrium position. In the case of $x = 0$, the process works for a perfect crystalline structure. A reasonable number for x is around 5%, but it always will depend on the structure under study. We will discuss this parameter in Section 4.

3. Computational procedure

The algorithm was tested using a tungsten BCC crystalline structure of 2000 atoms, with a lattice constant of $a = 3.1652$ Å, at different temperatures. The molecular dynamics simulations were performed in the microcanonical ensemble using the Finnis–Sinclair [12,13] inter-atomic potential, with a total time of $50{,}000\Delta t$ where $\Delta t = 1$ fs. The temperatures chosen for the samples were 300 K, 1000 K, 2000 K, 3000 K, 3500 K and 6000 K. In order to achieve this, each sample was thermalized to the desire temperature during the first $30{,}000\Delta t$, by rescaling the velocities. Then the sample was relaxed during the next $20{,}000\Delta t$. The final structure was taken for each temperature, as an example see Fig. 1, and used to generate random vacancies, and then the new configuration was tested in the vacancies recognition code. We will identify this structure later on, as the *initial structure*, associated to each temperature. All the MD calculations were performed using a standard code [14].

In this work we present the results of the algorithm performance using a NVIDIA Quadro K6000 card, with compute capability 3.5. The CUDA version used supports a new shared memory scheme,[2] which is used in the vacancies and atom management, taking advantage of the C++ object-oriented design in the algorithm. The Central Processing Unit (CPU), to compare with, is an AMD Opteron 6272 Interlagos 2.1 GHz 16 MB L3 cache. In the CPU, the algorithm was compiled with `GNU-g++ version 4.7` using the compiler flags `-O3 -Wall`.

4. Results

For each temperature we use an initial structure with 2000 atoms, as it was mentioned in Section 3, in a cubic cell of 31.652 Å. For these structures, with different temperature each one, the vacancies were randomly generated, from 2% to 12% every 2% one hundred times for each percentage, taking the same initial structure. The final results are the average of those hundred structures analyzed with the algorithm. In the vacancy search process we use $n = 10{,}000$ random points.

[2] http://devblogs.nvidia.com/parallelforall/unified-memory-in-cuda-6/.

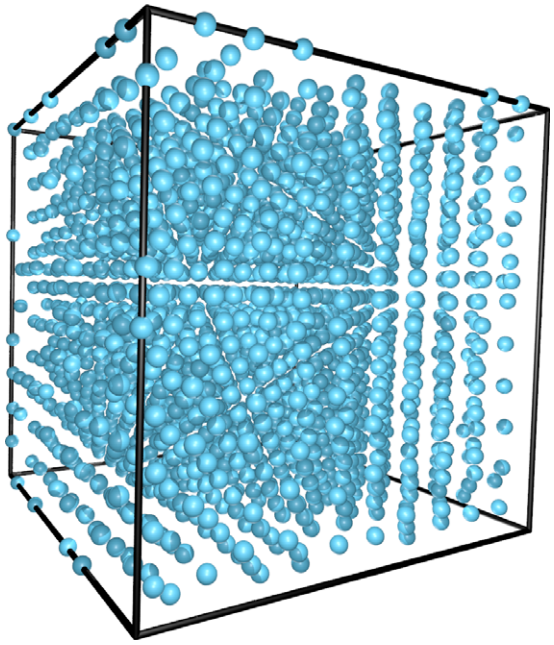

Fig. 1. The figure shows a tungsten crystal at 1000 K. The configuration correspond to the last step of the molecular dynamics simulation.

4.1. The R_0 parameter

A first analysis, related to the parameter R_0 with f_{ovp} in a range between 0 and 0.3, was made using the structures with a temperature of 1000 K. The values of R_0 used were $R_0^1 = 1.4765$ Å, $R_0^2 = 1.3805$ Å and $R_0^3 = 1.5726$ Å, and correspond to the average between $R_1/2$ and $R_2/2$, $R_1/2 + 0.01$ Å and $R_2/2 - 0.01$ Å, respectively. The results for these three different values of R_0 are displayed in Fig. 2. We can observe a "plateau" for a specific range of values in f_{ovp} associated to each R_0, which gives the correct ratio of vacancies found over vacancies generated ($v_f/v_g \sim 1$). For R_0^1, Fig. 2(b), it is found a reasonable number of vacancies when the overlap value is between 0.11 and 0.19. On the other hand, when the values are closest to $R_1/2$ and $R_2/2$ (R_0^2 and R_0^3), the optimal values of f_{ovp} are displaced to lower and higher values respectively, as we can observe in Fig. 2(a) and (c).

Based on Eq. (2) and taking the value of R_1/R_2 from Table 1, we can obtain the adequate percentage, x, from the f_{ovp} values found. If we take R_0^1, the suggested value of x is 5% approximately. If we take this 5% for R_0^2 and R_0^3, the values of f_{ovp} correspond to 0.029 and 0.3549 respectively, which are in agreement with the optimal expected values in Fig. 2(a) and (c). The above suggests that for different choices of R_0 we must set different values of f_{ovp}. Based on the previous results the choice of f_{ovp} (or x) is highly dependent of the choice of R_0. From here, we will use the average as the parameter choiced for R_0, i.e. R_0^1. The next step will analyze the effect of the temperature in the system.

4.2. Temperature effect in overlapping

The effect of the temperature in the f_{ovp} value, for a fixed R_0, was also analyzed. The results are presented in Fig. 3 for temperatures of 300 K, 2000 K, 3500 K, and 6000 K. We can observe that at room temperature the optimal values of overlap are located in the same range that the case presented at 1000 K in Fig. 2(b). The plateau, that is observed clearly at low temperature, starts to disappear when we approach the melting temperature of tungsten ∼3683 K [15]. Despite this, the optimal values of f_{ovp} are in the same range, and directly related to the number of intrinsic vacancies

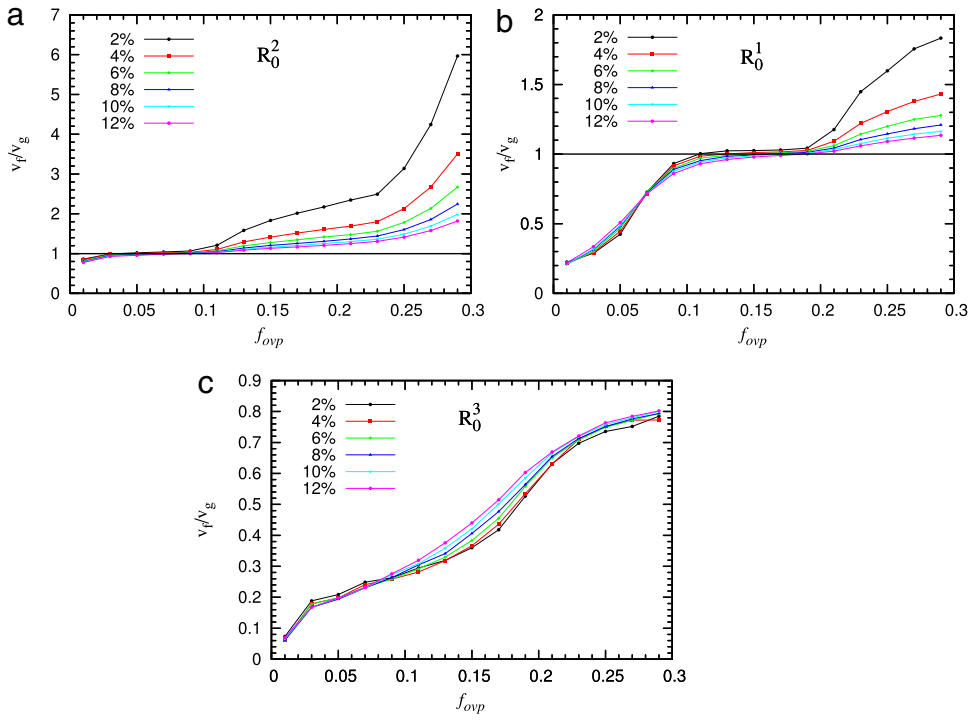

Fig. 2. The figure shows the average found vacancies v_f with respect to generated ones v_g. Three different cases are presented for different values of R_0: 1.380571, 1.476586, and 1.572600, which correspond to (a), (b), and (c) respectively. The optimal values for R_0 and f_{ovp} are given when $v_f/v_g \sim 1$.

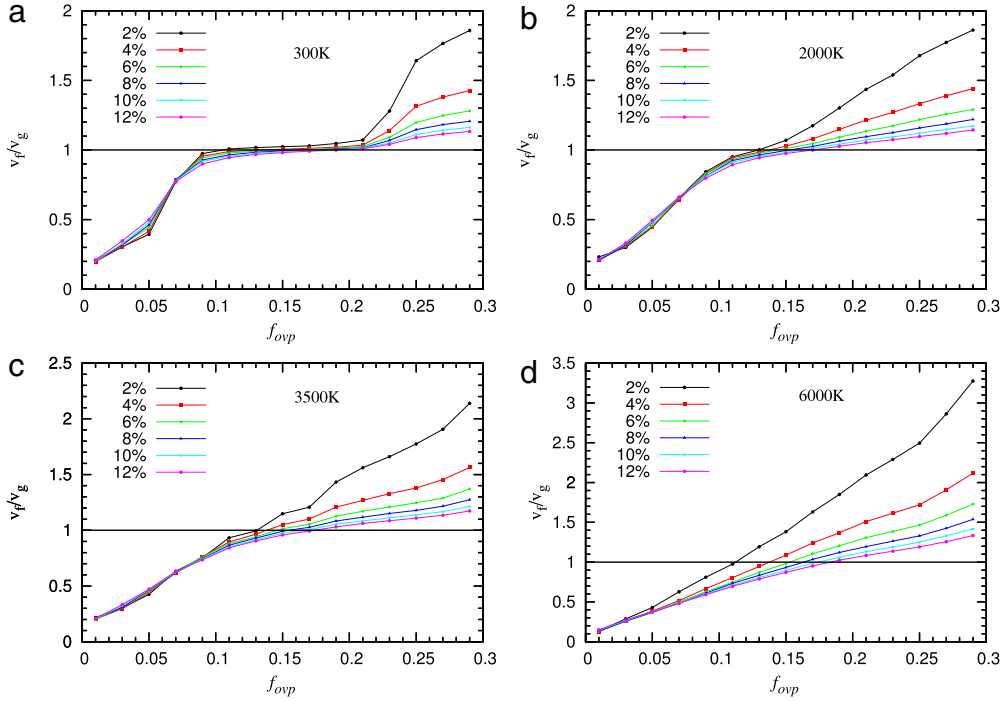

Fig. 3. The figures show the effect of the temperature in the f_{ovp} parameter. The temperatures correspond to 300 K, 2000 K, 3500 K and 6000 K, for (a), (b) (c) and (d) respectively. As the temperature increases, the plateau (that identifies the optimal values of f_{ovp}) disappears. Notice that for the liquid case (d), the range of f_{ovp} is between 0.11 and 0.19, similar to the case for (a), the solid case.

present in the sample. It is well-known that a vacancy cannot be defined in a liquid. Regardless, some atoms were removed from a structure snapshot from a liquid state, to test the behavior of the algorithm, this could be of interest in the context of amorphous structures. Interestingly the range is almost the same when the structure corresponds to a liquid state.

4.3. The algorithm accuracy

The algorithm accuracy was assessed by calculating the distance Δx between the position of the vacancies generated and found, and the total number of vacancies in each case. The results are presented in Table 2 for three different temperatures and three

Table 2
Estimated distance between the position of the vacancy generated and found, with a confidence of 68% and 95%, for different temperatures, vacancies and number of random points. This table shows at the same time, the number of vacancies generated (v_g) and the number found (v_f) for 100,000 random points.

Temperature	%Vac (v_g)	max Δx %68 confidence			max Δx %95 confidence			v_f (for n_2)
		n_1	n_2	n_3	n_1	n_2	n_3	
300 K	2 (40)	0.4211	0.3991	0.5046	0.8712	0.9036	0.8777	40.77±0.88
	6 (120)	0.3979	0.3831	0.4482	0.8221	0.8253	0.8680	119.46±1.45
	12 (240)	0.4217	0.4071	0.4353	0.8007	0.7812	0.7971	234.2±2.75
1000 K	2 (40)	0.4016	0.3748	0.3805	0.8121	0.8267	0.8448	40.92±0.67
	6 (120)	0.4230	0.3896	0.3612	0.8248	0.8075	0.7959	119.47±1.38
	12 (240)	0.4443	0.4160	0.4122	0.8061	0.7917	0.7830	233.69±3.01
3000 K	2 (40)	0.4802	0.4362	0.4266	0.8431	0.8377	0.8358	41.5±1.35
	6 (120)	0.4921	0.4595	0.4516	0.8508	0.8411	0.8299	116.37±2.5
	12 (240)	0.5063	0.4753	0.4703	0.8572	0.8383	0.8376	223.49±4.26

Table 3
Comparing the GSF and SF algorithm using different architectures.

# of Atoms	Architecture	GSF (seg)	SF (seg)
2,000	GPU	8.05	272.61
	CPU	19.40/529.78 (32CPU/1CPU)	74.11 (1CPU)
16,000	GPU	42.25	2,328.85
	CPU	500.95/15,543.33 (32CPU/1CPU)	1,881.64 (1CPU)

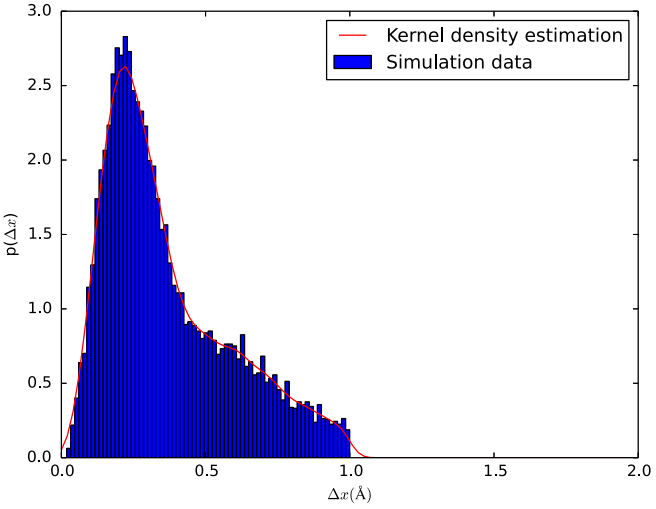

Fig. 4. The error distribution (blue bars) for the case of $T = 1000$ K, $n = n_2$ and 6% of generated vacancies. The kernel density estimation (red line) was used to calculate the maximum bounds.

number of random points, n (see Section 2), uniformly distributed: $n_1 = 10{,}000$, $n_2 = 100{,}000$, and $n_3 = 500{,}000$. The value of f_{ovp} chosen was 0.13, due to the good results obtained for the temperature of 1000 K. The average calculation time for each case were 13.6±0.2s, 30.9±0.7s, 107.0±2.8s for n_1, n_2 and n_3, respectively.

We only present the number of vacancies found, v_f, in the case of n_2, because the results are in the same range for n_1 and n_3. We report maximum bounds for Δx with 68% and 95% confidence, calculated using kernel density estimation [16]. We use these bounds instead of the center confidence intervals because the error distribution is not symmetrical, as shown Fig. 4.

The results show that the use of a higher n does not significantly improve the spatial location of the vacancy, and becomes detrimental to the calculation speed. The accuracy of the number of vacancies found slightly decreases when the temperature is close to the melting point or the percentage of vacancies increases. In general the GSF algorithm finds (or is close to) the right number and coordinates of the placed vacancies. The lower vacancies found at 3000 K for 6% and 12% of generated vacancies is expected based on the overlap value used. As we can see in Fig. 3(b) and (c) for $f_{ovp} = 0.13$ the rate v_f/v_g is lower than 1.

In Fig. 5 the structures at 1000 K and 3000 K, show the positions of generated and found vacancies, both with a 6% of vacancies. A visual inspection also suggests that a reasonable number of vacancies were found in the process, and the location of each of them is near to the original removed atoms.

4.4. Computational efficiency

A time consuming comparison between the original SF [7] and the GSF algorithms has been made using CPU and GPU architectures. There is no clear procedure to do these comparisons [17,18], for this reason we just consider the simulation time for each case. In GPU, we have incorporated CUDA code in the SF algorithm (easily migratable parts), to compare with the GSF. On the other hand the GSF algorithm has been migrated to CPU using one and multi-CPU (32 CPU, based in OpenMP), in order to compare with SF. Table 3 shows the results for 2000 and 16,000 atoms with 4% and 2% of vacancies respectively.

The larger time corresponds to the original SF algorithm with GPU, the poor result remains in that SF is not directly convertible to GPU during the Monte Carlo procedure, and the constant calls to the CUDA kernels implemented dropped the performance. For the CPU case, the GSF is faster than SF with the use of OpenMP, however without OpenMP the performance is lower than the original SF algorithm using a single CPU. The results show that an improvement is achieved from the original SF algorithm, with the new GSF (particularly for the GPU implementation). Next, a comparison between the cases with GPU/GSF and CPU/SF (using one single CPU) is presented for different cases of vacancies and number of atoms.

The comparison presented in Table 4 shows five different structure sizes that were considered: with 2000, 4,000, 8,000, 16,000, and 32,000 atoms at $T = 1000$ K. The structures were generated using the procedure described in Section 3. Three different vacancy percentages were randomly generated for each case, 2%, 6%, and 12%. The codes were executed using the values of $R_0 = 1.4766$ Å and $f_{ovp} = 0.15$.

Table 4 shows the calculation time for the different simulation sizes for GPU/GSF and CPU/SF algorithms. We can see an improvement from ∼15X to ∼100X from the smaller to larger systems respectively. This is a fundamental change in the study of large-scale

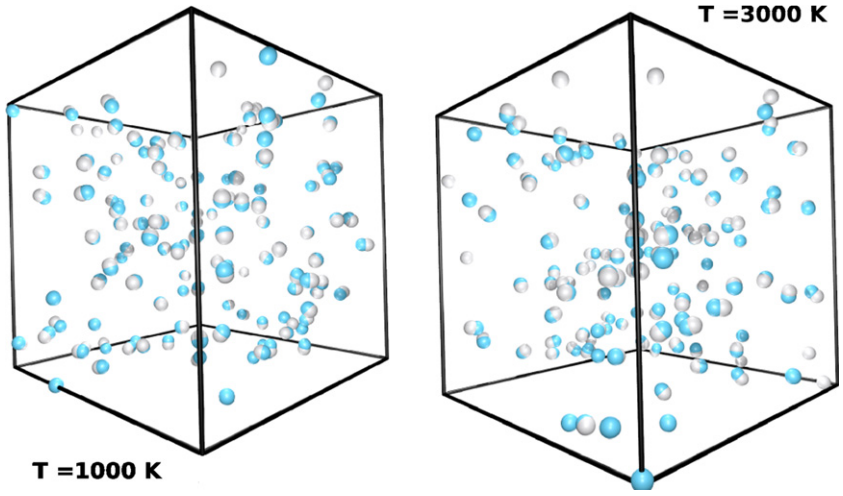

Fig. 5. The figure shows the removed atoms (light blue spheres) and the vacancies found with the algorithm (white spheres) at two different temperatures. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 4
Values of the calculation time for the different algorithms, GSF and SF, with different structure sizes and vacancies.

# of Atoms	2% GSF/SF (s)	6% GSF/SF (s)	12% GSF/SF (s)
2,000	6.50/75.23	7.60/95.42	8.9/160.1
4,000	14.58/223.26	12.52/440.76	15.2/461.7
8,000	37.93/1046.34	47.12/1555.72	58.5/2679.0
16,000	118.00/4803.81	137.90/10769.61	177.3/13193.96
32,000	422.01/30917.45	549.08/53870.07	746.1/92219.03

systems. Albeit the speed up seems promising, it is fundamental to compare with the CPU/GSF case using parallel implementations, where with 32 CPU using OPENMP, show that the GPU speed up is just between ∼2X and ∼11X for small and larger systems respectively.

5. Conclusion

The use of GPU architecture in the search and fill algorithm, to find vacancies in crystalline and non-crystalline structures, has been incorporated. We have used spatial decomposition which is straightforward to implement in GPU. The results show a radical improvement in the calculation speed, due to the use of GPU architecture without loss of accuracy. Because simulated annealing minimization is not longer needed, a reduction of adjustable parameters is also achieved. We have determined a relation between the parameters needed by the GSF algorithm, f_{ovp} and R_0, which are highly related to the unit cell of the structure under study. The range of the f_{ovp} is mostly preserved independently of the temperature, which allow us to obtain a reasonable percentage of vacancies.

An enhancement was achieved in speed from ∼15X to ∼100X compared with one-single CPU and SF algorithm, and from ∼2X to ∼11X compared with multicore-CPU (32 CPU) and GSF algorithm. This means that problems that involve a very large number of atoms are easily treatable now. Further treatment can be addressed to multi-component systems and structures with voids.

Acknowledgments

This work is supported by FONDECYT Iniciación 2013, 11130501. SD and JP also acknowledge partial funding from FONDECYT 1140514.

References

[1] V. Maurice, G. Despert, S. Zanna, M.-P. Bacos, P. Marcus, Self-assembling of atomic vacancies at an oxide/intermetallic alloy interface, Nature Mater. 3 (10) (2004) 687–691. http://dx.doi.org/10.1038/nmat1203.
[2] K. Badura-Gergen, H.-E. Schaefer, Thermal formation of atomic vacancies in Ni3Al, Phys. Rev. B 56 (6) (1997) 3032–3037. http://dx.doi.org/10.1103/physrevb.56.3032.
[3] H. Zhang, M. Khalkhali, Q. Liu, J.F. Douglas, String-like cooperative motion in homogeneous melting, J. Chem. Phys. 138 (12) (2013) 12A538. http://dx.doi.org/10.1063/1.4769267.
[4] M. Forsblom, G. Grimvall, How superheated crystals melt, Nature Mater. 4 (2005) 388–390. http://dx.doi.org/10.1038/nmat1375.
[5] F. Delogu, Cooperative dynamics and self-diffusion in superheated crystals, J. Phys. Chem. B 109 (32) (2005) 15291–15296. http://dx.doi.org/10.1021/jp052000x.
[6] X.-M. Bai, M. Li, Ring-diffusion mediated homogeneous melting in the superheating regime, Phys. Rev. B 77 (13) http://dx.doi.org/10.1103/physrevb.77.134109.
[7] S.M. Davis, A.B. Belonoshko, B. Johansson, Searchfill: A stochastic optimization code for detecting atomic vacancies in crystalline and non-crystalline systems, Comput. Phys. Comm. 182 (5) (2011) 1105–1110. http://dx.doi.org/10.1016/j.cpc.2010.12.009.
[8] M.J. Pozo, S. Davis, J. Peralta, Statistical distribution of thermal vacancies close to the melting point, Physica B 457 (0) (2015) 310–313. http://dx.doi.org/10.1016/j.physb.2014.10.023.
[9] M. Ciżnicki, M. Kierzynka, P. Kopta, K. Kurowski, P. Gepner, Benchmarking data and compute intensive applications on modern CPU and GPU architectures, Procedia Computer Science 9 (2012) 1900–1909. http://dx.doi.org/10.1016/j.procs.2012.04.208.
[10] E. Danovaro, A. Clematis, A. Galizia, G. Ripepi, A. Quarati, D. D'Agostino, Heterogeneous architectures for computational intensive applications: A cost-effectiveness analysis, J. Comput. Appl. Math. 270 (2014) 63–77. http://dx.doi.org/10.1016/j.cam.2014.02.022.
[11] J. Nickolls, I. Buck, M. Garland, K. Skadron, Scalable parallel programming with CUDA, Queue 6 (2) (2008) 40. http://dx.doi.org/10.1145/1365490.1365500.
[12] M.W. Finnis, J.E. Sinclair, A simple empirical *n*-body potential for transition metals, Phil. Mag. A 50 (1) (1984) 45–55. http://dx.doi.org/10.1080/01418618408244210.
[13] G. Ackland, A. Sutton, V. Vitek, Twenty five years of Finnis–Sinclair potentials, Phil. Mag. 89 (34–36) (2009) 3111–3116. http://dx.doi.org/10.1080/14786430903271005.
[14] S. Davis, C. Loyola, F. González, J. Peralta, Las Palmeras molecular dynamics: A flexible and modular molecular dynamics code, Comput. Phys. Comm. 181 (12) (2010) 2126–2139. http://dx.doi.org/10.1016/j.cpc.2010.08.030.
[15] V. Krsjak, S. Wei, S. Antusch, Y. Dai, Mechanical properties of tungsten in the transition temperature range, J. Nucl. Mater. 450 (1–3) (2014) 81–87. http://dx.doi.org/10.1016/j.jnucmat.2013.11.019. Special Theme Issue on Spallation Materials Technology. Selected papers from the Eleventh International Workshop on Spallation Materials Technology (IWSMT-11).
[16] B.W. Silverman, Density estimation for statistics and data analysis.
[17] V.W. Lee, C. Kim, J. Chhugani, M. Deisher, D. Kim, A.D. Nguyen, N. Satish, M. Smelyanskiy, S. Chennupaty, P. Hammarlund, R. Singhal, P. Dubey, Debunking the 100x gpu vs. cpu myth: An evaluation of throughput computing on cpu and gpu, SIGARCH Comput. Archit. News 38 (3) (2010) 451–460. http://dx.doi.org/10.1145/1816038.1816021.
[18] C. Gregg, K. Hazelwood, Where is the data? why you cannot debate cpu vs. gpu performance without the answer, in: Performance Analysis of Systems and Software (ISPASS), 2011 IEEE International Symposium on, 2011, pp. 134–144. http://dx.doi.org/10.1109/ISPASS.2011.5762730.