Computer Physics Communications 182 (2011) 1105–1110

Contents lists available at ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc

# SearchFill: A stochastic optimization code for detecting atomic vacancies in crystalline and non-crystalline systems

Sergio M. Davis [a,*], Anatoly B. Belonoshko [a], Börje Johansson [a,b]

[a] Applied Materials Physics, Department of Materials Science and Engineering, KTH, SE-100 44 Stockholm, Sweden
[b] Condensed Matter Theory Group, Department of Physics, Uppsala University, Uppsala, Box 530, Sweden

ARTICLE INFO

*Article history:*
Received 7 June 2010
Received in revised form 26 November 2010
Accepted 2 December 2010
Available online 7 December 2010

*Keywords:*
Monte Carlo
Vacancies
Interstitials
Computer simulation
Numerical optimization

ABSTRACT

We present an implementation of a stochastic optimization algorithm applied to location of atomic vacancies. Our method labels an empty point in space as a vacancy site, if the total spatial overlap of a "virtual sphere", centered around the point, with the surrounding atoms (and other vacancies) falls below a tolerance parameter. A Metropolis-like algorithm displaces the vacancies randomly, using an "overlap temperature" parameter to allow for acceptance of moves into regions with higher overlap, thus avoiding local minima. Once the algorithm has targeted a point with low overlap, the overlap temperature is decreased, and the method works as a steepest descent optimization.

Our method, with only two free parameters, is able to detect the correct number and coordinates of vacancies in a wide spectrum of condensed-matter systems, from crystals to amorphous solids, in fact in any given set of atomic coordinates, without any need of comparison with a reference initial structure.



## 1. Introduction

The presence and behavior of defects and atomic vacancies is an important factor in the study of solids [1,2]. While structural point defects, as well as dislocations and disclinations in crystals can be detected, for example, by analyzing the coordination numbers of individual atoms [3], or by more elaborated procedures such as Voronoi-like constructions [4,5] or the Common Neighbor Analysis (CNA) [6,7], a general procedure for the detection of hollow regions of space is more complicated.

In atomistic simulations where the initial structure is regular and well known, the identification of vacancy sites is a relatively straightforward procedure. This initial structure can be used as a reference to label any given site as "occupied" or "vacant" [8].

Unfortunately, a reference structure is not always available. Consider for example the problem of locating defect interstitials in an amorphous solid, vacancies in a polycrystalline structure, or in a solid structure obtained by "freezing" a liquid. Even thermal vibrations in the solid can make the "reference structure" approach not trivial.

In this work we present a robust algorithm for locating interstitial spaces within a structure, without the need for a reference structure. This is accomplished by gradually placing "virtual spheres" on the places where they can fit without overlapping with real atoms (or other virtual spheres). Each virtual sphere wanders randomly through the system, following a Monte Carlo procedure which attempts to minimize the total overlap, below a certain fixed threshold.

## 2. Theoretical background

When two identical spheres of radius $R_0$ overlap with each other (see Fig. 1), each one loses a volume given by [9]

$$V_m = \pi\left(\frac{2}{3}R_0^3 - \frac{1}{3}R_0^2 r - \frac{1}{12}R_0 r^2 \frac{1}{24}r^3\right). \tag{1}$$

Dividing by the volume of one of the spheres, the fractional overlap is

$$f_{over}(r) = \frac{2V_m}{\frac{4}{3}\pi R_0^3} = 1 - \frac{1}{2}\frac{r}{R_0} - \frac{1}{8}\left(\frac{r}{R_0}\right)^2 + \frac{1}{16}\left(\frac{r}{R_0}\right)^3. \tag{2}$$

The equivalent expression for 2D is

$$F_{over}(r) = \frac{1}{\pi}\left[\cos^{-1}\left(\frac{r}{2R_0}\right) - \frac{r}{4R_0^2}\sqrt{4R_0^2 - r^2}\right]. \tag{3}$$

Using Eq. (2), we can calculate the total overlap of a sphere, centered at an arbitrary position $\vec{r}$, with its neighbors,

$$F_{over}(\vec{r}) = \sum_i f_{over}\left(|\vec{r} - \vec{r}_i|\right)\theta\left(2R_0 - |\vec{r} - \vec{r}_i|\right), \tag{4}$$

where $\theta$ is the step function, which makes the sum only over the non-zero terms coming from neighbors within a distance of $2R_0$.

* Corresponding author.
*E-mail address:* sdavis@gnm.cl (S.M. Davis).

0010-4655/$ – see front matter 
doi:10.1016/j.cpc.2010.12.009

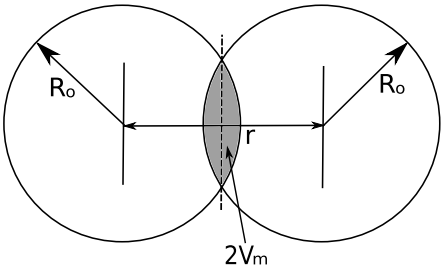


**Fig. 1.** Intersection of two identical spheres.

We can use the following operational definition: a point $\vec{r}$ in space will be labeled as a vacancy if the total overlap of a "virtual sphere", centered at $\vec{r}$ and having radius $R_0$ is smaller than a threshold $\Omega$, defined by

$$\Omega = \gamma \frac{1}{N} \sum_{i=1}^{N} F_{over}(\vec{R}_i), \tag{5}$$

where the $\vec{R}_i$ are the coordinates of the $N$ atoms in the system. Thus, $\Omega$ is proportional to the overlap "felt" by an occupied site on average.

By using this definition it is possible to devise an automatic procedure, which will first compute the threshold $\Omega$ for the set of atomic coordinates and the atomic radius $R_0$ (using $\gamma$ as a fixed external parameter), without any need for a regular reference structure. Then, iteratively proceed to identify points $\vec{V}_k$ in space with $F_{over}(\vec{V}_k) < \Omega$.

The parameter $\gamma$ controls the "greediness" of the threshold $\Omega$, a small value of $\gamma$ will direct the search towards really empty regions of space, improving the precision of the algorithm, while larger values will allow for certain overlapping between the vacancy and the surrounding atoms. A value of $\gamma = 1$ keeps the threshold at the same value of the typical overlap of an atom in the system, so for most uses a value of $\gamma$ slightly higher than 1 is recommended.

The radius $R_0$ will be usually taken as half the nearest neighbor distance, as determined from a previous calculation of the radial distribution function $g(r)$ [10], i.e., the first peak in $g(r)$ happens at $r = 2R_0$.

It is important to note that, for the computation of the total overlap $F_{over}$, all the previously identified vacancies must be considered as if they were real atoms, otherwise the procedure will never reach a stopping point. In this sense the algorithm will search for a vacancy candidate, fills it with an "occupied" virtual sphere, and goes for the next candidate, hence the name.

## 3. Algorithm description

1. Put a test vacancy site at a random coordinate $\vec{r}_0$ inside the simulation box.
2. If the current total overlap $\omega_i = F_{over}(\vec{r}_0)$ is less than the threshold $\Omega$, then a new vacancy was found. In this case, place an "empty sphere" centered at $\vec{r}_0$ and go back to step 1.
3. Otherwise generate a random displacement $\Delta\vec{r}$ for the test vacancy.
4. If the new total overlap $\omega = F_{over}(\vec{r}_0 + \Delta\vec{r}) < \omega_i$ then accept the move.
5. Otherwise reject the move with a probability $P_r = e^{(\beta/\omega)\Delta F} - 1.0$, where $\beta/\omega$ acts like the inverse temperature $(k_B T)^{-1}$ in the Metropolis formulation. This "temperature" is set to be proportional to the current overlap $\omega$ to improve convergence when the algorithm is getting close to a minimum, like in the steepest-descent family of optimization algorithms.
6. Go back to step 2 unless the maximum number of iterations `NMAX` was exceeded. In this case, the program ends. The coordinates of all the vacancies found are saved to a separate file.

It should be possible to use an alternate stopping criterion (at step 6) instead of completing NMAX iterations, but this "early exit" might leave some vacancies out at the end of the run. One possible heuristic criterion could be to stop if the best starting location for the next test vacancy exceeds a certain multiple of $\Omega$ (in that case, no coordinate seems "empty enough" to admit another vacancy).

## 4. Implementation details

The `SearchFill` code is implemented in `C++`, using a link cell algorithm [10] to speed up the computation of interatomic distances, needed for the sum in $F_{over}$. The default value of $\gamma$ is taken equal to 1.2, and $\beta$, the "inverse temperature" for the Metropolis algorithm is taken to be $\beta \approx 4$, which gives a probability $P_r \approx 1/2$ of accepting a move which increases the overlap by 10%.

## 5. Results

Fig. 2 (left side) shows an ideal face-centered cubic (FCC) structure with 108 atoms, in which 11 atomic vacancies have been created, by removing the same amount of atoms at random. Fig. 2 (right side) shows the same structure, after the vacancies have been located and filled using the search-and-fill algorithm. Due to the regularity of the structure, the search-and-fill procedure is able to find exactly the 11 vacancies in the correct positions. (See Table 1.)

Fig. 3 (left side) shows the same FCC structure with 108 atoms as Fig. 2, but this time equilibrated to $T = 1500$ K, by using molecular dynamics with an embedded-atom potential [11] describing the interatomic forces. To make some sense of the temperature scale, 8000 K for this particular potential and density corresponds to the critical superheating temperature.

From this structure again 11 atomic vacancies have been created, by removing the same amount of atoms at random. Fig. 3 (right side) shows the reconstructed structure, after the vacancies have been located and filled.

Table 2 shows the result of applying the search-and-fill procedure to a body-centered cubic (bcc) sample, consisting of 5488 atoms, at different temperatures, covering all the range from an ideal to a superheated solid. It is clear the performance of the algorithm decreases with temperature but still is quite close to finding the right number and coordinates of the placed vacancies. All the runs were performed using the default values for $\beta$ and $\gamma$. (See

**Table 1**
Results for 100 runs of the search-and-fill algorithm on a 108 atom, face-centered cubic crystal equilibrated via molecular dynamics at different temperatures and with different number of vacancies explicitly created.

| System | $T$ (K) | Vacancies | Vacancies found | Error in positions (Å) |
|---|---|---|---|---|
| FCC, 108 atoms | 0 | 1 | $1 \pm 0.0$ | $0.682 \pm 0.250$ |
| FCC, 108 atoms | 0 | 2 | $1.7 \pm 0.46$ | $0.626 \pm 0.186$ |
| FCC, 108 atoms | 0 | 5 | $4.2 \pm 1.10$ | $0.713 \pm 0.225$ |
| FCC, 108 atoms | 0 | 11 | $9.43 \pm 2.00$ | $0.635 \pm 0.107$ |
| FCC, 108 atoms | 1500 | 1 | $1 \pm 0.0$ | $2.480 \pm 0.339$ |
| FCC, 108 atoms | 1500 | 2 | $2.46 \pm 0.68$ | $1.442 \pm 0.274$ |
| FCC, 108 atoms | 1500 | 5 | $4.48 \pm 1.08$ | $0.907 \pm 0.239$ |
| FCC, 108 atoms | 1500 | 11 | $9.13 \pm 1.15$ | $0.849 \pm 0.156$ |
| FCC, 108 atoms | 2500 | 1 | $2.05 \pm 0.36$ | $1.300 \pm 0.371$ |
| FCC, 108 atoms | 2500 | 2 | $2.99 \pm 0.33$ | $1.028 \pm 0.195$ |
| FCC, 108 atoms | 2500 | 5 | $5.7 \pm 0.79$ | $0.999 \pm 0.172$ |
| FCC, 108 atoms | 2500 | 11 | $11.05 \pm 0.94$ | $1.022 \pm 0.122$ |

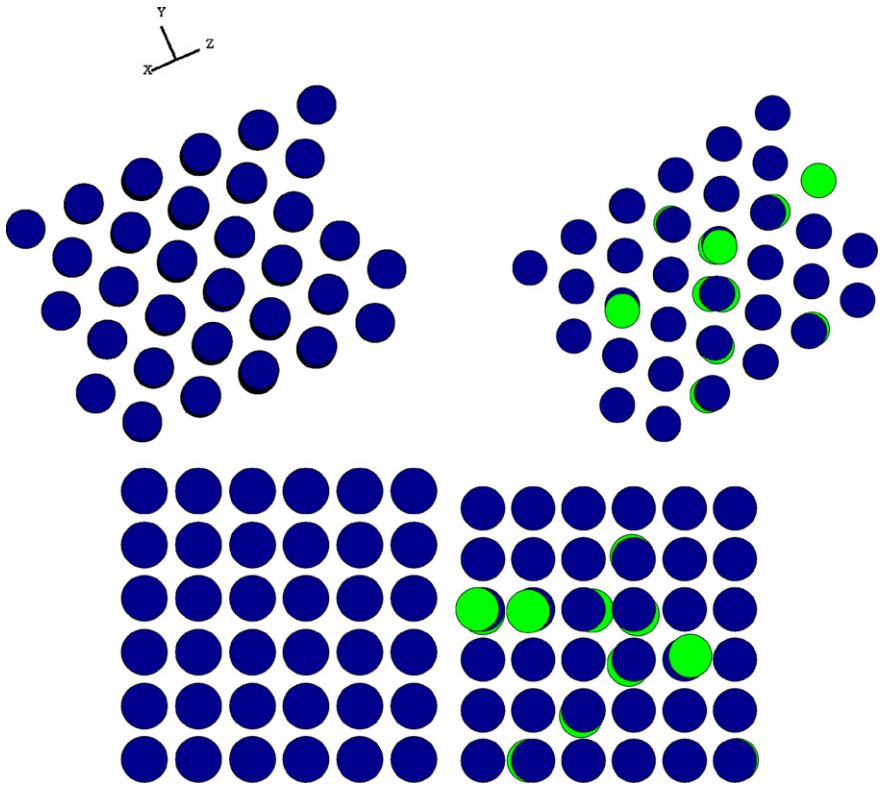


**Fig. 2.** Face-centered cubic, 108 atoms. Left, with 11 atoms removed at random. Right, reconstructed with the search-and-fill algorithm.

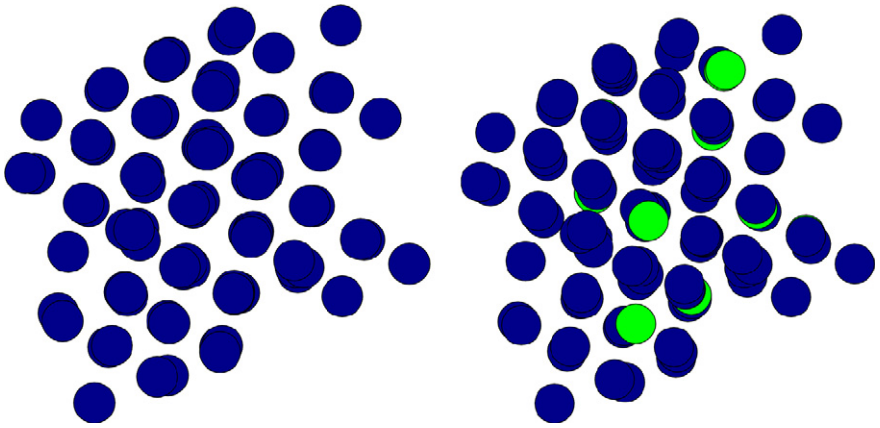

**Fig. 3.** Face-centered cubic, 108 atoms. Left, with 11 atoms removed at random. Right, reconstructed with the search-and-fill algorithm.

Figs. 4 and 5.) Table 3 shows a similar test, but using an amorphous sample with the same number of atoms.

## 6. Efficiency

Our implementation is usually fast enough (in a common desktop computer) for analysis of a single configuration of about 5000 atoms and 50 vacancies, taking around 130 seconds for a single run. By adjusting the maximum number of iterations `NMAX` below the default value, after performing a few tests, it is possible to reduce the total number of Monte Carlo steps, because the code will notice earlier the moment it is stuck in a local minimum.

Fig. 6 shows the performance of the algorithm for increasing system sizes (from 128 to 1548 atoms) but constant number of vacancies (namely, three). The scaling obeys a power law,

$$t(N) \propto N^{0.84}, \tag{6}$$

where $N$ is the number of atoms.

**Table 2**
Results for 100 runs of the search-and-fill algorithm on a 5488 atom, face-centered cubic crystal equilibrated via molecular dynamics at different temperatures and with different number of vacancies explicitly created.

| System | $T$ (K) | Vacancies | Vacancies found | Error in positions (Å) |
|---|---|---|---|---|
| BCC, 5488 atoms | 0 | 10 | $9.93 \pm 0.70$ | $0.185 \pm 0.016$ |
| BCC, 5488 atoms | 0 | 25 | $24.99 \pm 0.10$ | $0.185 \pm 0.012$ |
| BCC, 5488 atoms | 2000 | 10 | $7.70 \pm 2.50$ | $0.218 \pm 0.036$ |
| BCC, 5488 atoms | 2000 | 25 | $23.37 \pm 1.18$ | $0.212 \pm 0.016$ |
| BCC, 5488 atoms | 4000 | 10 | $9.84 \pm 0.99$ | $0.268 \pm 0.035$ |
| BCC, 5488 atoms | 4000 | 25 | $23.32 \pm 0.63$ | $0.351 \pm 0.025$ |
| BCC, 5488 atoms | 6000 | 10 | $7.65 \pm 0.94$ | $0.352 \pm 0.046$ |
| BCC, 5488 atoms | 6000 | 25 | $23.61 \pm 3.68$ | $1.829 \pm 0.018$ |
| BCC, 5488 atoms | 8000 | 10 | $8.47 \pm 1.38$ | $0.353 \pm 0.048$ |
| BCC, 5488 atoms | 8000 | 25 | $20.67 \pm 2.90$ | $1.779 \pm 0.038$ |

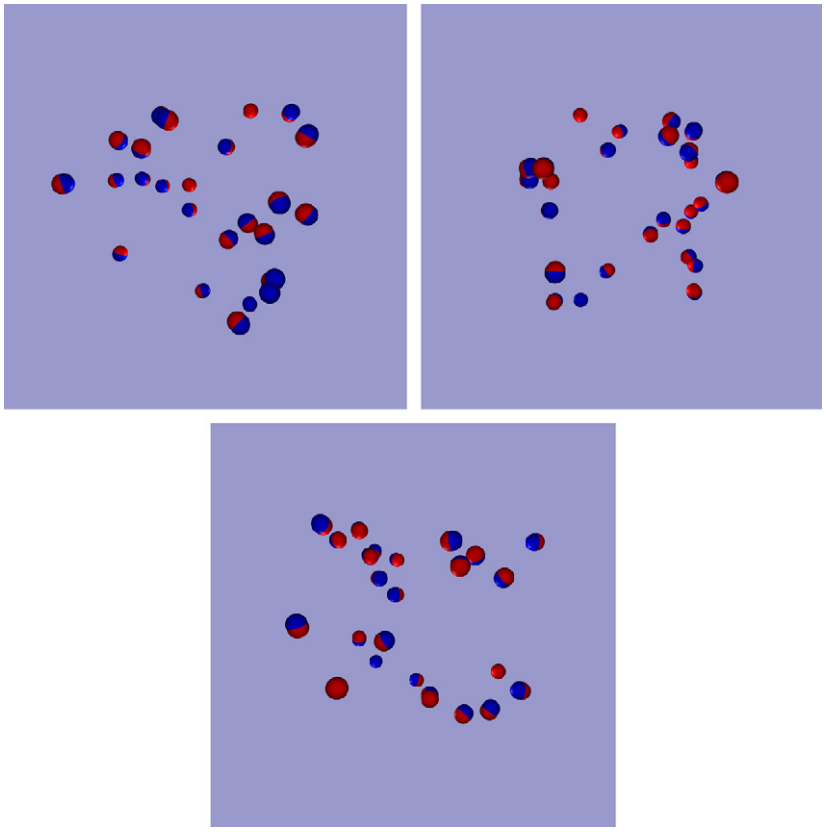

**Fig. 4.** (Color online.) Positions of the 25 vacancies placed in the 5488 atom fcc sample at $T = 4000$ K (light spheres), together with the positions found by the search-and-fill algorithm (dark spheres). The three panels are different views of the same set of coordinates, projected in the XY (upper left), YZ (upper right) and XZ (down) planes.

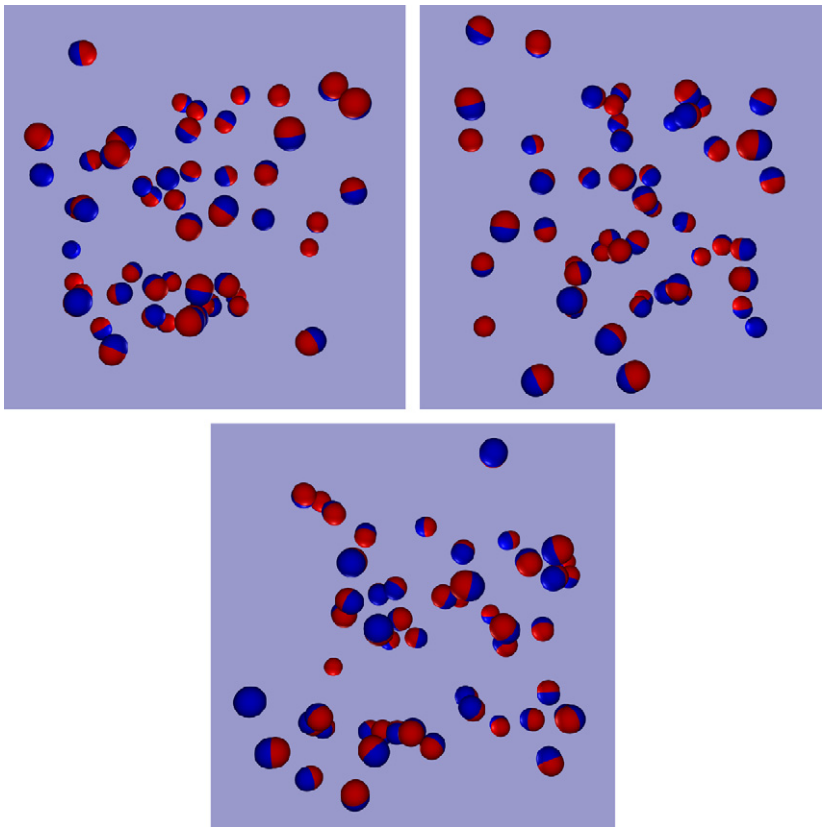

**Fig. 5.** (Color online.) Positions of the 50 vacancies placed in the 5488 atom amorphous sample (light spheres), together with the positions found by the search-and-fill algorithm (dark spheres). The three panels are different views of the same set of coordinates, projected in the XY (upper left), YZ (upper right) and XZ (down) planes.

**Table 3**
Results for 100 runs of the search-and-fill algorithm on a 5488 atom amorphous sample with different number of vacancies introduced.

| System | Vacancies | Vacancies found | Error in positions (Å) |
|---|---|---|---|
| Amorphous, 5488 atoms | 10 | $9.81 \pm 1.26$ | $0.305 \pm 0.030$ |
| Amorphous, 5488 atoms | 25 | $25 \pm 0.0$ | $0.383 \pm 0.033$ |
| Amorphous, 5488 atoms | 50 | $49.74 \pm 0.44$ | $0.376 \pm 0.025$ |

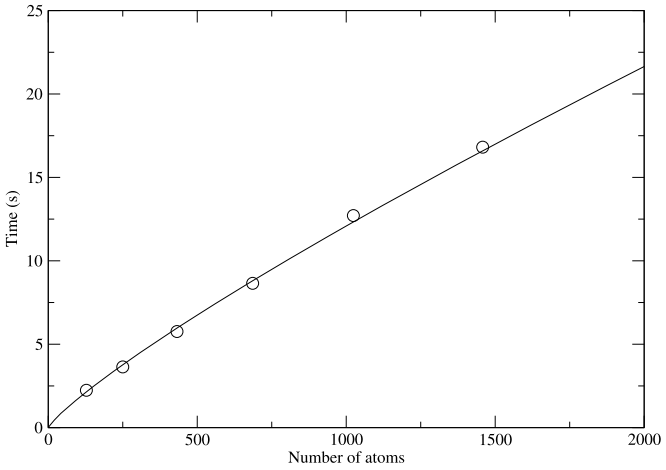


**Fig. 6.** Time needed to complete a search for three vacancies, under different system sizes (from 128 to 1458 atoms).

The performance of the algorithm can be fine-tuned by adjusting the free parameters $\beta$ and $\gamma$. The inverse "temperature" $\beta$ has an optimum value around 3 times $2\ln 2$ for the samples considered (Fig. 7). Small values of $\beta$ will give more freedom to explore the solutions but, at the same time, also impair the search. Larger values of $\beta$ will improve the "descent" when closing in to a solution but at the same time increase the likelihood of the search being stuck at a local minimum. Reducing $\gamma$ will always improve the precision of the coordinates obtained, as shown in Fig. 8 but at increasing computational cost.

The code has not been parallelized, but this and other optimizations are planned as a continuation of this work.

## 7. Concluding remarks

Our search-and-fill algorithm has proven useful in the task of counting and locating individual vacancies in crystals as well as amorphous solids. It seems to perform better when increasing the number of vacancies (the error in the positions is smaller). The method is, in principle, also applicable to detecting nanometer-sized pores in amorphous materials. It is just a matter of choosing a larger value of $R_0$.

## 8. Code compilation

The code is C++ according to the ISO C++98 standard, so it should compile without problems on every compiler supporting C++. It has been tested with the GNU compiler version 4.1 and 4.3, as well as the Intel C++ compiler, version 10.1. The code is self-contained, its only dependency is the math library, `libm` on UNIX.

## 9. Example run

An example run is shown in Fig. 9. Here we consider $R_0 = 1.01$ Å, and we fix the threshold $\Omega = 0.15$ (i.e., $\gamma \approx 3$, to facilitate the search). The configurations are from an amorphous Fe sample (5488 atoms) from which exactly 10 atoms have been removed at random. The procedure finds precisely those 10 atoms.

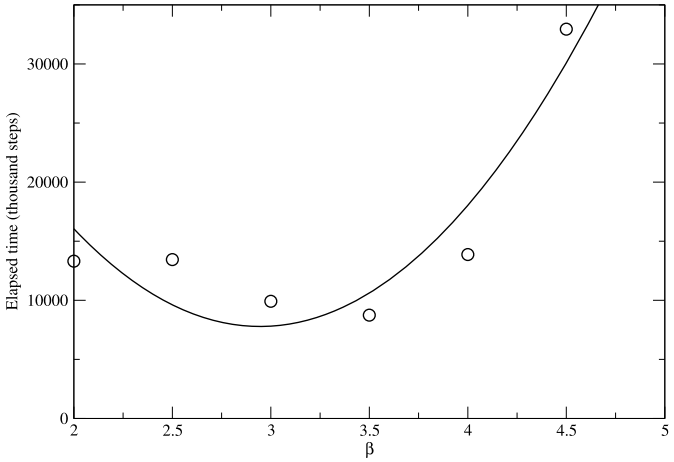


**Fig. 7.** Number of steps needed to complete the vacancy search, for different values of the inverse "temperature" $\beta$, in units of $2\ln 2$. The system was a 5488-atom bcc crystal at $T = 4000$ K with 10 vacancies introduced.

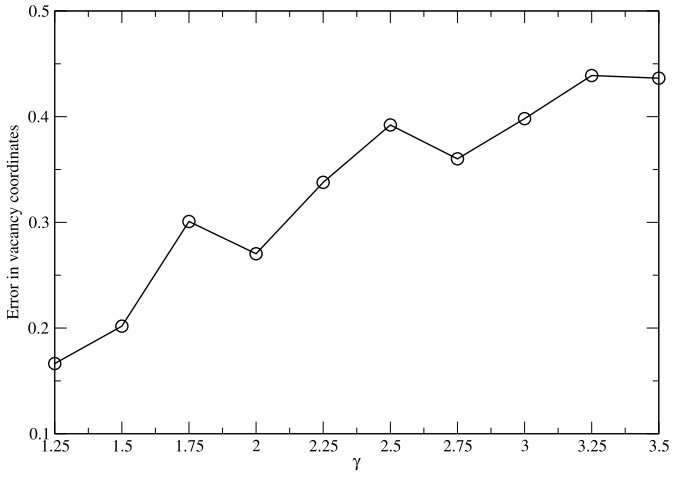


**Fig. 8.** Error in vacancy positions (Å) for different values of the "greediness" parameter $\gamma$. The system was a 5488-atom bcc crystal at $T = 4000$ K with 10 vacancies introduced.

The order of command-line parameters is as follows: first, the file containing the atomic coordinates (in `XYZ` format, described below), then the simulation cell lengths $L_x$, $L_y$ and $L_z$, followed by the value of $R_0$ and, optionally, the number of runs to perform and the fixed threshold $\Omega$. If no value of $\Omega$ is given, a suitable default is used by taking $\gamma = 1.2$. The `XYZ` format consists of a line containing the number of atoms, followed by a title line (which in this case is ignored by the program but must be present, even empty) and then one line for every atom, containing the atomic symbol and then the $x$, $y$ and $z$ coordinates.

## Acknowledgements

Computations were performed at the National Supercomputer Center (NSC) in Linköping and the Parallel Computer Center (PDC) in Stockholm. We also thank the Swedish Research Council VR for financial support.

```
$ ./searchfill amorphous-10holes.xyz 33.472 33.472 33.472 1.01 1 0.15
-> Loading input file amorphous-10holes.xyz
-> Read 5478 atoms, C of M = 16.8635, 16.9221, 16.8873
-> Constructing linkedcell lists...
-> Linkedcell using cell cutoff 4.94214: 4.89321 in units of r0
-> Sum of linked cell atoms : 5478
-> Linkedcell data is ready
-> Minimum overlap for the atomic positions: 0
-> Maximum overlap for the atomic positions: 0.248983
-> Average overlap for the atomic positions: 0.0494267
-> Setting target overlap to be: 0.15
-> Found one vacancy! (ov=0.136902, coord=4) at 17.3243, 19.2943, 10.2266
->   after 5533 iterations...
-> Total vacancies up to now: 1
-> Found one vacancy! (ov=0.118826, coord=5) at 20.2988, 13.8957, 10.0201
->   after 30842 iterations...
-> Total vacancies up to now: 2
-> Found one vacancy! (ov=0.129275, coord=3) at 30.8162, 21.8924, 3.88638
->   after 22061 iterations...
-> Total vacancies up to now: 3
-> Found one vacancy! (ov=0.102263, coord=3) at 20.4038, 9.01094, 9.00861
->   after 36457 iterations...
-> Total vacancies up to now: 4
-> Found one vacancy! (ov=0.120147, coord=4) at 18.7785, 7.47403, 33.3196
->   after 3343 iterations...
-> Total vacancies up to now: 5
-> Found one vacancy! (ov=0.140018, coord=4) at 30.3941, 24.088, 18.8013
->   after 56404 iterations...
-> Total vacancies up to now: 6
-> Found one vacancy! (ov=0.124132, coord=6) at 1.20943, 20.4139, 32.5379
->   after 231275 iterations...
-> Total vacancies up to now: 7
-> Found one vacancy! (ov=0.140664, coord=2) at 17.8331, 28.1947, 21.0979
->   after 88157 iterations...
-> Total vacancies up to now: 8
-> Found one vacancy! (ov=0.138601, coord=3) at 3.4709, 23.4409, 10.6167
->   after 85449 iterations...
-> Total vacancies up to now: 9
-> Found one vacancy! (ov=0.137826, coord=5) at 13.8325, 16.0959, 18.5118
->   after 538058 iterations...
-> Total vacancies up to now: 10
Stuck in a local minimum! iter = 8070871, lastvacancy_iter=538058
-> No more vacancies found. Aborting...
-> Minimum overlap found is 0.416393, at 23.1049, 1.39359, 24.0897
-> Percentage difference with the target overlap is 177.595
Total number of vacancies found: 10
```

**Fig. 9.** The output from an example run of the SearchFill code. 10 atoms were initially removed, and the program find precisely those 10 atoms and their approximate location.

## References


[1] R.W. Balluffi, S.M. Allen, W.C. Carter, Kinetics of Materials, John Wiley and Sons, 2005.
[2] M. Hillert, Phase Equilibria, Phase Diagrams and Phase Transformations, Cambridge University Press, 1998.
[3] L. Gómez, A. Dobry, C. Geuting, H.T. Diep, L. Burakovsky, Phys. Rev. Lett. 90 (2003) 095701.
[4] G. Voronoi, J. Reine Angew. Math. 134 (1908).
[5] M. Forsblom, G. Grimvall, Nature Materials 4 (2005) 388.
[6] J.D. Honeycutt, H.C. Andersen, J. Phys. Chem. 91 (1987) 4950.
[7] F. Delogu, J. Phys. Chem. B 110 (2006) 12645.
[8] A.B. Belonoshko, S. Davis, N.V. Skorodumova, P.H. Lundow, A. Rosengren, B. Johansson, Phys. Rev. B 76 (2007) 064121.
[9] E.W. Weisstein, Sphere–sphere intersection, http://mathworld.wolfram.com/Sphere-SphereIntersection.html.
[10] M. Allen, D. Tildesley, Computer Simulation of Liquids, Clarendon Press, Oxford, 1987.
[11] A.B. Belonoshko, R. Ahuja, B. Johansson, Phys. Rev. Lett. 84 (2000) 3638.