NanoCap: A Framework for Generating Capped Carbon Nanotubes and Fullerenes

M. Robinson*

Nanochemistry Research Institute, Department of Chemistry, Curtin University, Perth, WA 6845, Australia

N.A. Marks

Discipline of Physics & Astronomy, Curtin University, Perth, WA 6845, Australia

Abstract

`NanoCap` provides both libraries and a standalone application for the construction of capped nanotubes of arbitrarily chirality and fullerenes of any radius. Structures are generated by constructing a set of optimal dual graph topologies which are subsequently optimised using a carbon interatomic potential. Combining this approach with a GUI featuring 3D rendering capabilities allows for the rapid inspection of physically sensible structures which can be used as input for molecular simulation.

Keywords: nanotubes, fullerenes, modelling
PACS: 61.48.De, 61.48.-c, 07.05.Tp,33.15.-e

PROGRAM SUMMARY

Manuscript Title: NanoCap - A Generator for Capped Carbon Nanotubes and Fullerenes
Authors: Marc Robinson
Program Title: NanoCap
Journal Reference:
Catalogue identifier:
Licensing provisions: Creative Commons Attribution-NonCommercial 2.5 (CC BY-NC 2.5)
Programming language: Python, C
Computer: Any system with Python (with NumPy and SciPy) and a C compiler.
Operating system: Linux, OS-X, Windows
RAM: up to 4 GB
Keywords: Molecular Modelling, Atomistic Simulation, Fullerene, Nanotube
Classification: 16.1 Molecular Physics and Physical Chemistry - Structure and Properties
External routines/libraries: NumPy[1], SciPy [2], EDIP [3], (GUI version: Qt+PySide [4], VTK [5])
Nature of problem:
The ability to readily produce arbitrary sized, low-energy fullerene and capped nanotube structures for molecular simulation.
Solution method:
Structures are generated using the dual lattice representation, which are subsequently optimised using physical carbon interatomic potentials.
Running time:
Scales dependent on the number of carbon atoms in the structure. For the C_{196} molecule 10 structures can be found in around 30 seconds on a single CPU.

[1] T. E. Oliphant. Comp. Sci. Eng. 9(3):1020 (2007). `http://www.numpy.org`
[2] E. Jones, T. E. Oliphant, P. Peterson *et al.* SciPy: Open Source Scientific Tools for Python. `http://www.scipy.org`
[3] N. A. Marks. Phys. Rev. B 63(3):035401 (2000).
[4] Digia. Qt - Cross-platform application and UI framework. `http://www.qt-project.org/`
[5] W. Schroeder. K. Martin, B. Lorensen. The Visualisation Toolkit (1993-2008) `http://www.vtk.org/`

1. Introduction

An immediate problem facing computational scientists who wish to model the physical and chemical properties of fullerenes or non-periodic capped nanotubes is the generation of sensible initial atomic coordinates. On one hand, highly symmetric topologies are readily available, such as icosahedral fullerenes which can be modified to terminate nanotubes of certain chiralities. A common example is the capping of the (5,5) or (9,0) nanotubes with a section of the C_{60}

*Corresponding author. *Address:* Curtin University, GPO Box U1987, Perth, Western Australia. *Tel.:* +61 8 9266 3780.
Email address: `Marc.Robinson@curtin.edu.au` (M. Robinson*)

fullerene. The construction of capped nanotubes using this approach is severely limited as the choice of nanotube chirality is constrained to known fullerene structures. Alternatively, complete enumeration of fullerene and cap topologies is possible when treating the carbon network as a 3-regular graph [1, 2]. However, as the number of carbon atoms increases, the number of possible topological arrangements grows exponentially. As a result, additional optimisation and analysis is required to produce a set of physically sensible structures. An application that enabled a computer modeller to quickly and directly generate a set of energetically favourable fullerene or capped nanotube structures would undoubtedly be a useful tool.

At the time of writing, there are two applications available which explicitly enable the construction of fullerenes and capped nanotubes. `Nanotube Modeler` [3] is capable of constructing both capped nanotubes and fullerenes, however options are limited particularly with regards to nanotube caps where only five simple chiralities [(9,0), (5,5), (6,6), (10,0) and (10,10)] are available. A larger number of fullerene structures are available but it is not possible to produce fullerene of arbitrarily diameter. The capping of nanotubes using sections of fullerenes can also be carried out using generic visualisation and data manipulation tools such as `Virtual NanoLab` part of the `Atomistix ToolKit` [4].

`FullGen` [1, 2] part of the `CaGe` software package also enables the generation of fullerenes and capped nanotubes but adopts a purely mathematical approach. This application is useful for the enumeration of structures and identifies the associated symmetry groups. Unlike the application described in this paper, `FullGen` does not attempt to produce physically sensible, low energy structures nor does it produce structures containing heptagons which may be important for studies of defects.

The fundamental function of the `NanoCap` application is the construction of physically sensible fullerene and capped nanotube topologies for use in atomistic computer simulations. To achieve this goal, the algorithms outlined are highly generalised such that fullerenes of any radius and capped nanotubes of arbitrary chirality can be constructed. Such a level of generalisation is possible due to the underlying concept behind the approach: the carbon lattice itself is not constructed but instead construction involves its dual lattice equivalent. This idea is illustrated in Fig. 1 which shows the connection between the carbon network and its face dual lattice triangulation. Notably, the duality demonstrated is reversible, the ability to generate the triangular mesh allows extraction of the 3-fold coordinate carbon network and vice-versa. Therefore, construction of the dual lattice implicitly produces a carbon lattice and herein lies the advantage of this approach; generating the dual lattice is simpler and faster.

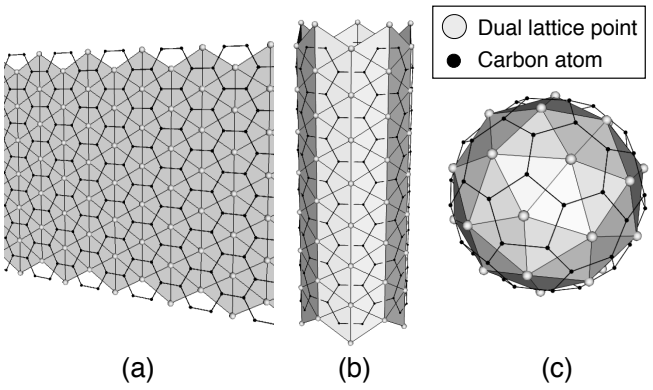

Figure 1: The connection between the carbon lattice (solid black spheres) and the face dual representation (larger grey spheres). (a) A graphene layer. (b) The (5,5) nanotube, which is constructed by rolling the graphene layer shown in (a). (c) The C_{60} fullerene and its dual lattice the truncated icosahedron, which is typically used to cap the (5,5) nanotube. Bonds are shown between neighbouring carbon atoms.

The reason for the relative ease in which the dual lattice is produced becomes apparent when considering the lattice as close packed arrangement of spheres. This arrangement can be constructed by maximising the distance between the centre point of each sphere. A way of achieving this is to assign a repulsive potential energy function to each point and minimise the net total energy. This approach is analogous to a well studied problem, *The Thomson Problem* [5, 6, 7, 8, 9, 10, 11, 12, 13], which concerns the minimum energy arrangement of point charges on the surface of a sphere.

The applicability of the proposed algorithm was recently demonstrated through the construction and examination of caps for nanotubes of various chiralities [14]. The reader is directed to this study for an overview of literature specific to the construction of nanotube caps. Using Density Functional Tight Binding (DFTB) to accurately model the carbon system, this work validated the ability of the approach to produce low energy cap topologies. This conclusion

emphasised the value of the approach and provided the motivation for the development of `NanoCap`.

2. Algorithm Flow Chart

An overview of the algorithms present in `NanoCap` is shown in Fig. 2. The core procedures (represented by rectangles with double-struck vertical edges) vary in complexity but fundamentally involve the creation, manipulation or analysis of point sets. Each routine is described in the proceeding sections following the order illustrated in Fig. 2.

3. Fullerene Dual Lattice Initialisation

The initial set of fullerene dual lattice points are produced by randomly distributing points on the sphere. This is achieved by drawing two random numbers, z_0=[−1,1] and t_0=[0,2π], which are then used to construct the cartesian coordinates:

$$\begin{aligned} x_i &= \sqrt{1-z_0}\cos(t_0) \\ y_i &= \sqrt{1-z_0}\sin(t_0) \\ z_i &= z_0 \end{aligned} \tag{1}$$

During the dual lattice optimisation process, a single point is fixed at the pole to constrain the rotational degrees of freedom of the system during the search for alternate structures. An additional point may also be held on the equator to further reduce the generation of symmetry equivalent structures.

4. Nanotube Dual Lattice Initialisation

Algorithms for the generation of uncapped nanotubes are well documented [15, 16] and typically concern the construction nanotubes that are periodic along the tube axis. However, producing finite capped nanotubes removes the requirement of such periodicity and a simpler approach to nanotube generation can be adopted. Fundamentally, the construction of a nanotube involves the transformation from a single 2D sheet of graphene. To correspond to the eventual construction of the nanotube, the 2D axis are labelled x and z, where z represents the nanotube axis. As illustrated in Fig. 3, this sheet is traditionally defined using two lattice vectors $\mathbf{a_1}$ and $\mathbf{a_2}$:

$$\mathbf{a_1} = \tfrac{\sqrt{3}a_c}{2}(\sqrt{3}, 1) \tag{2}$$

$$\mathbf{a_2} = \tfrac{\sqrt{3}a_c}{2}(\sqrt{3}, -1) \tag{3}$$

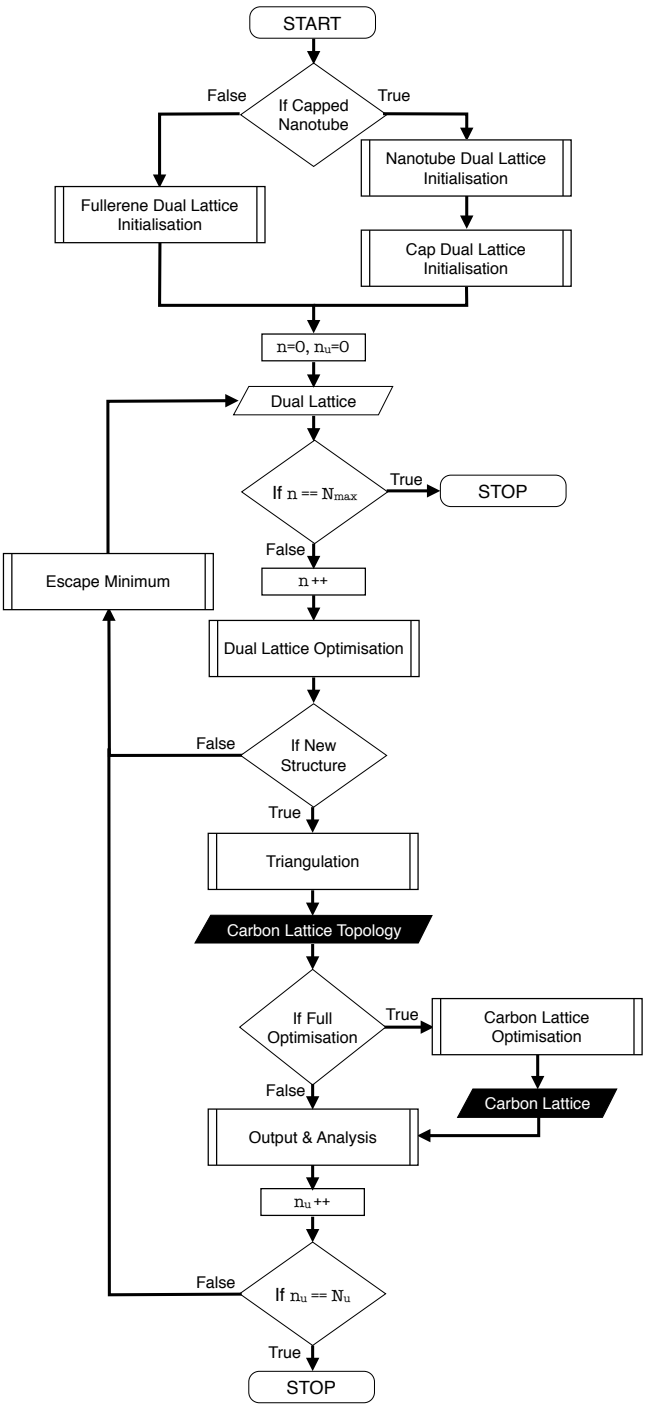

Figure 2: Flow diagram for the construction of a carbon fullerene or capped nanotube. Each core process (represented by rectangles with double-struck vertical edges) is explained in the text. In the above diagram, $\mathtt{N}_u$ indicates the number of unique structures to be found. $\mathtt{N}_{max}$ limits the minima searching procedure.

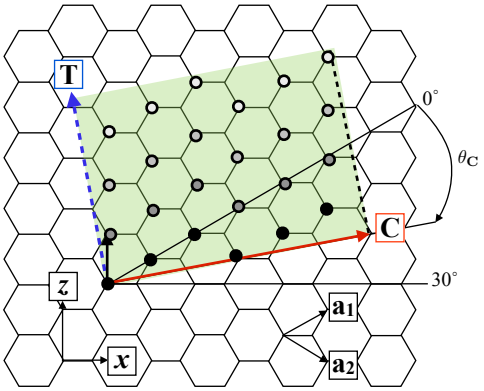

Figure 3: Definition of the 2D lattice points **P** used for nanotube construction. At each point, the basis shown in Fig. 4 is constructed. Points are constructed in rows along the chiral vector as indicated by the graduated colour scale. Example chiral vector shown is for the (4,2) nanotube.

where a_c is the carbon bond length of 1.421 Å.

The properties, symmetry and caps of a nanotube depend on the orientation of the underlying graphene sheet. To describe this orientation, a vector is introduced, namely the chiral vector **C** which is defined in terms of $\mathbf{a_1}$ and $\mathbf{a_2}$:

$$\mathbf{C} = (n, m) = n\mathbf{a_1} + m\mathbf{a_2} \tag{4}$$

The magnitude of **C** gives the circumference of the nanotube and as such the cylindrical radius r_t is given by:

$$r_t = \frac{|\mathbf{C}|}{2\pi} \tag{5}$$

which is used to determine the scaling factor γ to reduce the nanotube to a cylinder of unit radius. (γ=$1/r_t$).

The angle **C** subtends with the axis of the graphene sheet is labelled the chiral angle $\theta_\mathbf{C}$ as shown in Fig. 3. Two special cases of nanotube are constructed when $\theta_\mathbf{C}$ equals 0° [**C**=$(n, 0)$] and 30° [**C**=(n, n)]. These *achiral* nanotubes are known as the *zigzag* and *armchair* respectively and have additional mirror plane symmetry when compared to *chiral* [**C**=(n, m)] nanotubes.

Perpendicular to **C** is the vector **T** that defines the axial direction of the nanotube:

$$\mathbf{T} = (-\mathrm{C}_z, \mathrm{C}_x) \tag{6}$$

The vectors $\widehat{\mathbf{C}}$ and $\widehat{\mathbf{T}}$ form a new orthonormal coordinate system that can be used to align the nanotube with the cartesian axis.

The construction of each point **P** in the 2D graphene lattice is carried out row by row along **C**. These rows are indicated in Fig. 3 and contain n+m points. The points in each row are produced iteratively, with the point at iteration i given by:

$$\mathbf{P}_i = n_i\mathbf{a_1} + m_i\mathbf{a_2} \tag{7}$$

with (n_i, m_i) ranging from (0,0) to (n, m). The incrementation of m_i or n_i is dependent on:

$$\begin{array}{rl} n_i\texttt{++} & \text{if } \; m_i/(2n_i + m_i) > m/(2n + m) \\ m_i\texttt{++} & \text{if } \; m_i/(2n_i + m_i) \leq m/(2n + m) \end{array} \tag{8}$$

Each new row is translated in the z direction by $\sqrt{3}a_c$. Rows of points are continually added until the current length of the nanotube surpasses a user defined length. The minimum length is governed by the force cutoff described in Section 6.1 which is proportional to the density of dual lattice points in the cap. Performing the capping procedure with the shortest nanotube possible significantly reduces computational expensive, particularly in triangulation and rendering routines.

At each point in the 2D nanotube lattice basis points are required as shown in Fig. 4. The carbon atoms although not needed by `NanoCap` are produced by default. For each point **P** at position (p_x, p_z), the position of the **A** and **B** carbon atoms are given by:

$$\mathbf{A} = (p_x, p_z) \tag{9}$$

$$\mathbf{B} = (p_x + a_c, p_z) \tag{10}$$

The dual lattice point **D** which is fundamental to the operation of `NanoCap` is simply offset along the x direction to the centre of a hexagon:

$$\mathbf{D} = (p_x + 2a_c, p_z) \tag{11}$$

The final three points labelled I_{1-3} represent a potential C_2' axis of rotational symmetry once the 3D nanotube is constructed. The position of these points on the 2D lattice are given by:

$$\mathbf{I}_1 = (p_x + a_c/2, p_z) \tag{12}$$

$$\mathbf{I}_2 = (p_x + 5a_c/4, p_z + \sqrt{3}a_c/4) \tag{13}$$

$$\mathbf{I}_3 = (p_x + 5a_c/4, p_z - \sqrt{3}a_c/4) \tag{14}$$

These points are used to align the 2D lattice such that once in 3D, there exists a simple 180° rotation to map one end of the nanotube to the other. This

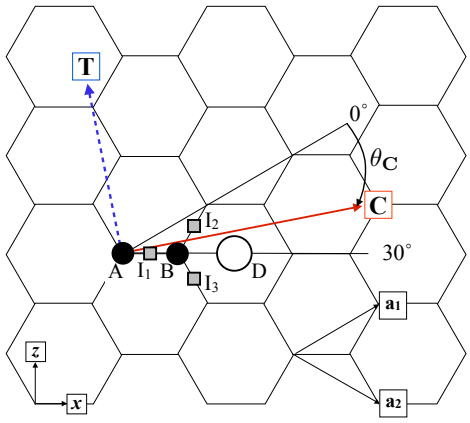

Figure 4: Basis positions of the atoms and points associated with the 2D nanotube sheet. A and B represent the carbon atoms, D indicates the position of the dual lattice point and I_{1-3} give the three intersections of carbon-carbon bonds which represent $C_2^{'}$ rotational axis of symmetry once the 3D tube is constructed. Repeating this basis for each 2D lattice point generates all atoms and points required for the nanotube cap construction.

is achieved by determining the point which intersects with the perpendicular bisector to $\mathbf{T}$. This point is used to shift the 2D lattice such that 3D x axis of the nanotube is inline with the $C_2^{'}$ axis of rotation.

Once constructed the 2D lattice is reorientated such that $\widehat{\mathbf{C}}$ and $\widehat{\mathbf{T}}$ align with the x and z cartesian axis. The dual lattice points are then shifted in the z direction such that the boundary is at z=0.

The transformation of the 2D lattice points into the 3D nanotube is straightforward. As the z position remains the same in 3D as in 2D, the only coordinate to consider is along the x axis. The position of each point i along the 2D x-axis gives the angle around the circumference of the nanotube:

$$\theta_i = 2\pi \frac{x_i^{2D}}{|\mathbf{C}|} \tag{15}$$

This angle is used to determine the cartesian coordinates of each point in the 3D xy plane:

$$\begin{aligned} x_i &= r_t \cos(\theta_i) \\ y_i &= r_t \sin(\theta_i) \end{aligned} \tag{16}$$

Once created, the dual lattice is scaled to a unit cylinder using γ defined previously. The dual lattice points belonging to the nanotube are held fixed during the minimisation process.

5. Cap Dual Lattice Initialisation

Upon construction of the nanotube of the required chirality, the number of dual lattice points in the tube are used to estimate the number of points that should be used in the generation of the caps. This estimate is extrapolated from the density of dual lattice points in the nanotube. It should be noted that for nanotubes that have a small radius, this estimate will be lower than the ideal number of points. For all cases of cap construction it is recommended to sample a range of dual lattice points, either side of the estimated number.

Generation of the initial cap points is carried out in a similar manner to that described for the initialisation of the fullerene dual lattice. The points are produced along the negative z coordinate, attaching to the nanotube at z=0. This is achieved by modifying equation 1 such that z_0=[−1,0]. These dual lattice points belonging to the cap are free to move during the minimisation process.

Currently, a single cap is constructed at one end of the nanotube which is then rotated 180° about the x axis to cap the alternate end. This is possible due to the alignment of the $C_2^{'}$ axis with the x axis during the construction of the nanotube. Producing identical caps at either end of the nanotube is important when determining cap energies as in our previous work [14]. As an alternative, different caps could be produced by repeating the initialisation, optimisation and triangulation processes for each end of the nanotube. Currently, the `NanoCap` software is constrained to produce identical cap structures.

6. Dual Lattice Optimisation

As introduced previously, optimising the dual lattice corresponds to finding a solution to the Thomson problem. For a given set of N_D points, this involves minimising the total potential energy:

$$\phi = \sum_{i=1}^{N_D} \sum_{j=i+1}^{N_D} \frac{1}{|\mathbf{r_i} - \mathbf{r_j}|} \tag{17}$$

which is analogous to the total electrostatic energy of a system of point charges. The algorithm outlined in the present work centres around minimising this potential energy for the dual lattice points constructed for either the fullerene or capped nanotube topologies. The process of minimisation of the fullerene

dual lattice points is directly in line with traditional approaches to solving the Thomson problem. At each iteration of the minimisation algorithm, the points are normalised back to the surface of the sphere. In addition, the radial component of force is removed from each atom, ensuring all forces act tangentially to the sphere's surface.

The algorithm adopted for the minimisation of the dual lattice is an implementation of the limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) method [17, 18, 19] as included in the `SciPy` optimisation libraries [21]. The method circumvents direct calculation of the Hessian matrix by continual updates and corrections during optimisation.

6.1. Modifications for Capping Nanotubes

In contrast to the fullerene dual lattice minimisation which is a direct analogue to solving the Thomson problem, the optimisation procedure associated with the capped nanotube dual lattice requires important modifications. Firstly, only the cap dual lattice points are free to move during minimisation, with the points generated on the tube held fixed. The implementation of this is trivial and involves the multiplication of the force by a binary mask of length $3N_D$, which contains zeros for the fixed points. A further adaption of the minimisation process specific to capping nanotubes is the surface to which the points are normalised. Dependent on which side of the tube-cap boundary a point lies, it is either scaled to the unit hemisphere or cylinder. This is an important feature as it allows cap points to move into the nanotube region, which is a necessity for highly chiral tubes where the tube boundary may be coarse.

A final and vital modification specific to capping nanotubes is associated with the range of the potential energy function. The introduction of the tube points which are held fixed generates a significant net repulsion towards the cap points. For long nanotubes, this can cause clustering of the cap points at the apex of cap. There are two solutions to this problem; truncate the potential at a given pair separation or restrict the number of tube points included in the force evaluation. Using the first option, cap points were found to penetrate the tube resulting in the need for different truncation criteria for the cap and tube points. The second option was found to be much more reliable and involved fewer modifications to the generalised optimisation routines implemented for the fullerene construction. To determine the tube points involved in the energy minimisation, a cutoff Z_c is introduced along the tube axis which depends on the number of cap points N_D^{cap}:

$$Z_c(\rho) = A \exp\left(\mu \rho^{\alpha}\right), \ \rho = \frac{\mathrm{N}_D^{cap}}{2\pi} \tag{18}$$

where A= 0.790378, μ = 1.62583 and α= 0.251753. Although this cutoff is calculated automatically by default, it can be manually overridden in situations where the cap density is significantly greater than the tube density.

7. Escaping Minima

As discussed in the literature regarding finding solutions to the Thomson problem, as the number of points increases the number of local minima grows exponentially [22]. For this reason, the code includes a tool for minima searching. This involves looping through the procedures for dual lattice minimisation, triangulation and if selected, the minimisation of the resulting carbon lattice using a physical carbon potential. The user defines the extent of the search by inputting the number of required unique minima and the maximum number of minima to search. After each successful minimisation, there are several options to escape the local potential energy basin. The two simplest methods are to either randomly offset the dual lattice points from the minima or to reset them completely. Resetting the structures can be very effective for systems with a large number of well separated local minima and can be quick to produce a multitude of structures. Randomly offsetting the dual lattice points is also effective and the minimisation process is typically quicker than a complete reset due to the closer proximity to a minimum. Importantly the perturbation is constrained to be tangential to the surface of either the sphere or cylinder. Also, in the case of nanotube cap generation, care is taken as to ensure no cap points are offset into the tube as points may become too close to the fixed points belonging to the tube. This ensures sensible initial forces during optimisation. The magnitude of the random tangential perturbation is determined by the nearest neighbour separation of the current set of points.

Although fast, resetting or offsetting the system in the hope to find a new basin is not reliable and can result in revisiting the same minima, particularly if

the system is in a wide basin. Therefore as a final option, the system with position vector $\mathbf{R}$ can escape the current minimum by iteratively climbing out of the basin following the gradient of the potential or the negative direction of force $\mathbf{F}$:

$$\mathbf{R}_{i+1} = \mathbf{R}_i - \alpha \mathbf{F}_i \tag{19}$$

where i denotes the iteration number and α is a variable step size dependent on the change in the magnitude of force per iteration. When the system leaves the vicinity of the current basin, the projection of force onto the separation vector changes sign:

$$(\mathbf{R}_{i+1} - \mathbf{R}_i) \cdot \mathbf{F}_i \tag{20}$$

This ensures the system visits a new basin, although does not guarantee the resultant structure will be symmetrically unique. As the algorithm only requires a method of escaping a minimum, advanced saddle point finding methods are not required.

Currently, structures are determined to be unique by comparison of the total dual lattice energy. Future revisions will introduce the detection of symmetry equivalent structures.

To encourage the sampling of different minima during the structure search, an option to add $3N_D$ dimensional Gaussians to the potential energy function has been implemented. If enabled, after each dual lattice optimisation stage a Gaussian is placed at the location of the new minima. In future steps of the structure search the total potential energy used during the dual lattice optimisation then has an additional contribution from previously placed Gaussians. This additional potential for each Gaussian g takes the form:

$$\begin{aligned} \phi_g(\mathbf{R}_\Delta) &= H \exp\left(-\mathbf{R}_\Delta^{\mathrm{T}} \mathbf{A} \mathbf{R}_\Delta\right) \\ \mathbf{R}_\Delta &= \mathbf{R} - \mathbf{R}_g \end{aligned} \tag{21}$$

where $\mathbf{R}$ and $\mathbf{R}_g$ are the $3N_D$ position vectors of the the dual lattice points and the Gaussian respectively. The matrix $\mathbf{A}$ contains $1/2\sigma^2$ on the diagonal. The height H and width σ of the Gaussian are user defined parameters and should ideally be significantly smaller than the widths and heights of potential wells on the dual lattice energy surface. Typical parameters used during development of `NanoCap` range from 0.1 to 0.5 for systems containing up to a few hundred dual lattice points. If these parameters are too small the structure search will take a long time to locate a new minima, yet clearly if they are too large then the potential energy surface may be modified to such an extent that the structure search fails completely.

Gaussian functions were chosen due the simple form of the derivatives with respect a component of the position vector r_α:

$$\frac{d\phi_g(\mathbf{R}_\Delta)}{dr_\alpha} = \phi_g(\mathbf{R}_\Delta) \cdot \frac{d\left(-\mathbf{R}_\Delta^{\mathrm{T}} \mathbf{A} \mathbf{R}_\Delta\right)}{dr_\alpha} \tag{22}$$

which allows for fast force evaluation.

8. Triangulation

Upon successful termination of the minimisation process, the dual lattice points represent a minimum energy structure with regards to the potential in Eq. 17. To construct the carbon lattice, the face dual of the dual lattice network is generated. This is achieved by reversing the dual lattice transformation shown in Figure 1 which requires triangulation of the dual lattice points. There is a significant literature and knowledge base regarding triangulation algorithms. Many of these algorithms concern much more complex topologies where the neighbourhood of each point varies greatly. For this reason, a simple procedure is adopted based upon 3D Delauny triangulation. This involves constructing triangles such that no point lies within the circumcircle created by each triangles vertices. As the routine progresses, the normals of each triangle are stored and are used in a secondary algorithm which ensures no triangles are constructed inside the closed surface. This is accomplished by checking the intersections of the forward and backward facing normals with other triangles. Any normal that intersects multiple triangles is flagged and the corresponding triangle is removed. This is an important check for nanotubes which have a diameter that is of a similar magnitude as the dual lattice point separation, for example the (5,0) nanotube.

Due to the various definitions of the centre of a triangle, the carbon lattice may be constructed in a number of different ways. Currently `NanoCap` places a carbon atom at the centroid of each triangle as this has been found to give the best spacing between atoms. This choice is only significant for the fullerene and cap regions as the dual lattice points belonging to the nanotube produce an equilateral triangulated mesh, for which each centre is coincident.

The triangulation routine is the first generalised module, working on fullerene and capped nanotube

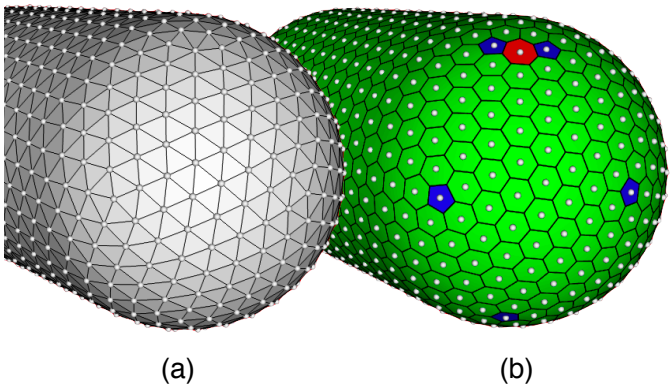

Figure 5: Minimised dual lattice of the capped (23,17) nanotube with 444 carbon atoms in the cap. (a) Triangulation of the dual lattice is shown in grey with the cap dual lattice points shown in white. (b) Rendering of carbon rings with pentagons in blue, hexagons in green and heptagons in red.

dual lattice points alike. After triangulation, a single set of carbon atoms are produced which are used in later analysis and minimisation routines.

9. Carbon Lattice Minimisation

As the fullerenes and capped nanotubes constructed from the dual lattice only represent sensible 3-fold topologies, an accurate physical model of carbon is required to produce the final atomic coordinates. Currently, the Environmental Dependent Interatomic Potential (EDIP) [23] is implemented within the code. EDIP provides a suitably physical model of carbon without the computational expensive of first principles methods. Future releases of `NanoCap` will incorporate other empirical carbon potentials, in particular those accessible through The Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) [24] using the already developed `Python` interface.

The carbon potential is used in a two stage process to produce the fully relaxed carbon structures. Firstly, the structures are scaled from the unit radius topologies outputted from the dual lattice minimisation procedure. For fullerenes, an initial estimate of the scale factor is determined using the known C_{60} radius. For capped nanotubes, an estimate is already known as the radius of the uncapped nanotube has been defined (Eq. 5). To determine the precise scale factor, constrained optimisation is performed using the Nelder-Mead simplex algorithm [25] as implemented in the `SciPy` libraries [21]. Once at a sensible scale, the structure is fully optimised using the same L-BFGS algorithm described previously [17, 18, 19, 21]. Both the scaled and fully optimised structures are stored and the associated atomic coordinates can be output to file. The scaled structures are useful for the construction of concentric carbon fullerenes (*carbon onions*) or capped multi-walled carbon nanotubes, as they allow for a uniform interlayer separation within the spherical regions.

10. Output and Analysis

At any time during a `NanoCap` session, structures can be saved to disk. The dual lattice, constrained carbon lattice and fully optimised carbon lattice can be saved in the common *.xyz* file format along with information relating to the structure, such as the ring statistics. In addition, if the GUI version of `NanoCap` is used, the Schlegel diagram can be saved as an image along with a isometric view of the structure. Outputted images can be encoded with software such as `FFMpeg` to produce animations of the structure.

10.1. Ring Counting

Ring counting is carried out using the approach outlined by Franzblau [26]. The method is centred around identifying all shortest path (SP) rings in a network where the connectivity is known. For each atom i in the constructed fullerene or capped nanotube, a neighbour list is generated and all rings involving atom i are identified. For each ring, a distance is computed between each vertex in terms of the number of intermediate vertices. If these distances do not correspond to the true distances in the full network, then the ring is not an SP ring and is discarded. Importantly, after all SP rings have been determined for atom i, it is removed from the full network of atoms. This removes all rings associated with atom i and eliminate the possibility for double counting. The application of this method is made possible using recursive algorithms, for both the generation of all rings containing vertex i and for the determination of SP rings.

10.2. Schlegel Transformation

In additional to the 3D rendering of the final structures `NanoCap` also includes the option to render the 2D Schlegel diagram. This represents the projection of the 3D network onto a plane and is particularly useful for the examination of the topology of capped

nanotubes. By default projection is carried out into the xy plane, inline with the nanotube orientation along the z axis. The projected 2D position $\mathbf{r}'_i$ of each atom or point i is then given by:

$$\begin{aligned} \mathbf{r}_i &= (x_i, y_i, 0) \\ \mathbf{r}'_i &= \mathbf{r}_i + \gamma_s \hat{\mathbf{r}}_i \end{aligned} \tag{23}$$

where γ_s controls the extent of the projection in the plane with a value of γ_s=0 resulting in a direct projection. A cutoff along the z axis, r_s is also incorporated which determines which points or atoms are used in the Schlegel construction. This is useful when only the capped region of a nanotube is required. Examples of a Schlegel diagrams for a range of capped nanotubes are shown in Fig 6.

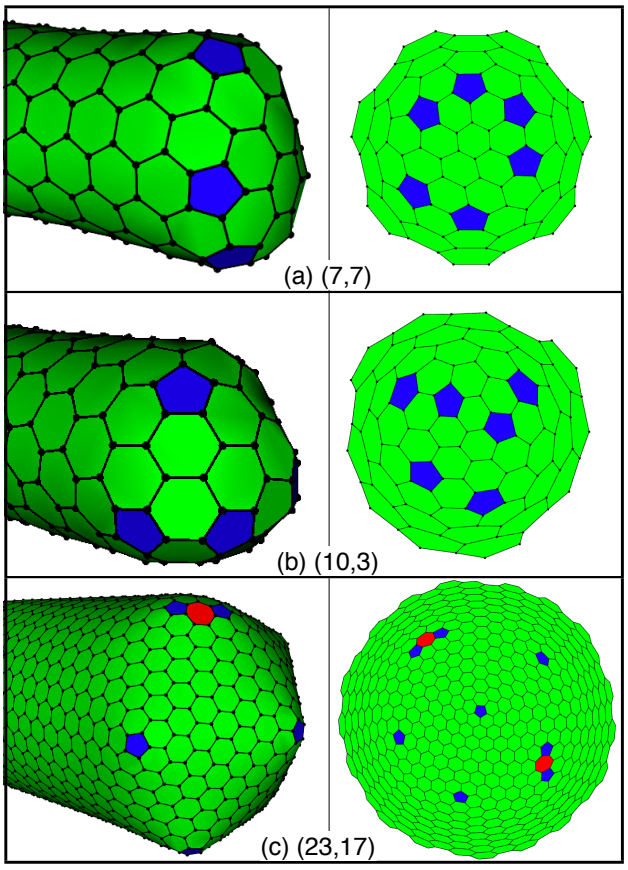

Figure 6: Fully optimised 3D isometric and 2D Schlegel views of a range of capped nanotubes. (a) The (7,7) nanotube with a 44 atom cap. (b) A capped (10,3) nanotube containing 50 carbon atoms. (c) The (23,17) nanotube with a cap containing 444 carbon atoms which includes heptagon ring structures.

11. Implementation and Source Code

The `NanoCap` framework is implemented in `Python`, with computationally expensive algorithms written in `C` and compiled as shared libraries. `NanoCap` is designed to be used as both a standalone application complete with a GUI and with 3D rendering capabilities and also as libraries to be used in `Python` scripts. To achieve this, the code takes advantage of the versatile object orientated nature of `Python` with core classes and routines separated from GUI and rendering elements. In addition, as `Python` is an interpreted language with relatively simple syntax, users with little programming experience can quickly produce custom scripts that access the complete functionality of the `NanoCap` libraries.

11.1. Core classes

The core classes and routines in `NanoCap` (`nanocap.core`, `nanocap.structures`, `nanocap.ext`) contain the main functionality for fullerene or capped nanotube construction. These modules can be used as libraries in `Python` scripts to produce structures quickly or in bulk. Only the `NumPy` [20] and `SciPy` [21] packages are required when using `NanoCap` as a library. An example script that uses the libraries to produce capped (10,10) nanotubes can be found in Figure 7. This script could readily be modified to run in parallel either locally on multiple threads or distributed to separate cpu nodes using `MPI` (or a `Python` wrapper such as `MPI4Py` [27]).

11.2. GUI and rendering

The `NanoCap` graphic user interface (GUI) is implemented using the `Qt` framework [28], using `PySide` bindings. Rendering is carried out using the Visualisation Toolkit (`VTK`) [29] which again is accessed through `Python` wrappers. The main advantage of using the GUI enabled version of `NanoCap` is the ability to visual inspect each new structure during generation. As a new structure is found, the user is presented with a 3D visualisation which can be interactive manipulated whilst the structure search continues. Each structure is dynamically entered into a structure table, which the user can view and use to load previously found structures.

```
import os
from nanocap.core.minimisation import DualLatticeMinimiser
from nanocap.core.minimisation import CarbonLatticeMinimiser
from nanocap.core.minimasearch import MinimaSearch
from nanocap.structures.cappednanotube import CappedNanotube
from nanocap.core.output import write_points

n,m = 10,10
l = 20.0
cap_estimate = True

N_nanotubes = 5
N_max_structures = 20
basin_climb = True
calc_rings = True

dual_lattice_minimiser = "Thomson"
carbon_lattice_minimiser = "EDIP"
dual_lattice_mintol=1e-10
dual_lattice_minsteps=100
carbon_lattice_mintol=1e-10
carbon_lattice_minsteps=100
optimiser="LBFGS"
seed = 12345

my_nanotube = CappedNanotube()
my_nanotube.setup_nanotube(n,m,l=l)

if(cap_estimate):
    N_cap_dual = my_nanotube.get_cap_dual_lattice_estimate(n,m)

my_nanotube.construct_dual_lattice(N_cap_dual=N_cap_dual,seed=seed)
my_nanotube.set_Z_cutoff(N_cap_dual=N_cap_dual)

Dminimiser = DualLatticeMinimiser(FFID=dual_lattice_minimiser,
                                  structure = my_nanotube,
                                  min_type= optimiser,
                                  ftol = dual_lattice_mintol,
                                  min_steps = dual_lattice_minsteps)

Cminimiser = CarbonLatticeMinimiser(FFID=carbon_lattice_minimiser,
                                  structure = my_nanotube,
                                  min_type= optimiser,
                                  ftol = carbon_lattice_mintol,
                                  min_steps = carbon_lattice_minsteps)

Searcher = MinimaSearch(Dminimiser,
                        carbon_lattice_minimiser= Cminimiser,
                        basin_climb=basin_climb,
                        calc_rings=calc_rings)

Searcher.start_search(my_nanotube.dual_lattice,
                      N_nanotubes,
                      N_max_structures)

Searcher.structure_log.write_log(os.getcwd(),"myStructures.out")

for i,structure in enumerate(Searcher.structure_log.structures):
    carbon_lattice = structure.carbon_lattice
    filename = "C{}_carbon_atoms_{}".format(carbon_lattice.npoints,i)
    write_points(filename,carbon_lattice,format="xyz")
```

Figure 7: Example `Python` script that uses the `NanoCap` libraries to create and save 5 capped (10,10) nanotubes of around 10 Å in length.

11.3. External libraries

Currently, the only external library included in `NanoCap` is EDIP. Other libraries such as the LAMMPS `Python` [24] module will be implemented in future releases. The `Fortran` source code for EDIP is compiled as a shared object and imported into `Python` using the `ctypes` library. To access the internal force and position arrays in EDIP, additional `Fortran` modules were written to serve as an interface. These access points are used by a `Python` interface that contains routines for EDIP initialisation, passing atomic coordinates and retrieving forces and energies. All external libraries are stored in the `ext` directory of the source.

12. Summary

`NanoCap` is a simple and generic application for the construction of low energy fullerene and capped nanotube structures. It provides an ideal tool to accompany the study of finite carbon molecules using atomistic computer simulation. The implementation involves a standalone application which includes a GUI and allows for dynamic visual inspection through 3D rendering. In addition, the `NanoCap` libraries can be used in custom `Python` scripts that enabled the user to produce structures in bulk or to include the structure generation routines into pre-existing code.

13. References

[1] G. Brinkmann and A. W. Dress, J. Algorithms **23**, 345 (1997).
[2] G. Brinkmann, U. Nathusius, and A. Palser, Discret. Appl. Math. **116**, 55 (2002).
[3] Nanotube Modeler - `http://www.jcrystal.com/`
[4] Atomistix ToolKit - `http://www.quantumwise.com/`
[5] D. J. Wales, H. McKay, and E. L. Altschuler, Phys. Rev. B **79**, 224115 (2009).
[6] D. J. Wales and S. Ulker, Phys. Rev. B **74**, 212101 (2006).
[7] E. L. Altschuler and A. Pérez-Garrido, Phys. Rev. E **71**, 1 (2005).
[8] A. Pérez-Garrido, Phys. Rev. B **62**, 6979 (2000).
[9] E. L. Altschuler and A. Pérez-Garrido, Phys. Rev. E **73**, 36108 (2006).
[10] A. Pérez-Garrido and M. A. Moore, Phys. Rev. B **60**, 15628 (1999).
[11] A. Pérez-Garrido, M. Ortuño, E. Cuevas, and J. Ruiz, J. Phys. A. Math. Gen. **29**, 1973 (1996).
[12] J. R. Edmundson, Acta Crystallogr. Sect. A Found. Crystallogr. **48**, 60 (1992).
[13] J. R. Edmundson, Acta Crystallogr. Sect. A Found. Crystallogr. **49**, 648 (1993).
[14] M. Robinson, I. Suarez-Martinez, and N. A. Marks, Phys. Rev. B **87**, 155430 (2013).
[15] R. Saito, G. Dresselhaus, and M. S. Dresselhaus, *Physical Properties of Carbon Nanotubes* (1998).
[16] E. B. Barros, A. Jorio, G. G. Samsonidze, R. B. Capaz, A. G. Souza Filho, J. Mendes Filho, G. Dresselhaus, and M. S. Dresselhaus, Phys. Rep. **431**, 261 (2006).
[17] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, ACM Trans. Math. Softw. **23**, 550 (1997).
[18] R. Byrd, P. Lu, J. Nocedal, and C. Zhu, SIAM J. Sci. Comput. **16**, 1190 (1995).

[19] J. L. Morales and J. Nocedal, ACM Trans. Math. Softw. **38**, 1 (2011).
[20] T. E. Oliphant. Comp. Sci. Eng. 9(3):1020 (2007). `http://www.numpy.org`
[21] E. Jones, T. E. Oliphant, and P. Peterson, *SciPy: Open Source Scientific Tools for Python* `http://www.scipy.org/`.
[22] T. Erber and G. M. Hockney, Phys. Rev. Lett. **74**, 1482 (1995).
[23] N. A. Marks, Phys. Rev. B **63**, 035401 (2000).
[24] S. Plimpton, J. Comput. Phys. **117**, 1 (1995).
[25] J. A. Nelder and R. Mead, Comput. J. **7**, 308 (1965).
[26] D. Franzblau, Phys. Rev. B **44**, 4925 (1991).
[27] MPI4Py - MPI for Python. `http://mpi4py.scipy.org/`
[28] Digia, *Qt - Cross-platform application and UI framework* `http://www.qt-project.org/`.
[29] W. Schroeder, K. Martin, and B. Lorensen, *The Visualisation Toolkit (VTK) - Copyright (c) 1993-2008* `http://www.vtk.org/`.