AVP — Another Void Program

Andrew C.R. Martin

Institute of Structural and Molecular Biology,
Division of Biosciences,
University College London,
Gower Street,
London WC1E 6BT

14th September, 2004
Updated 11th December, 2020

Abstract

AVP is a new method for the analysis of voids in proteins (defined as empty cavities not accessible to solvent). AVP combines analysis of individual discrete voids with analysis of packing quality. While these are different aspects of the same effect, they have traditionally been analysed using different approaches.

1 Introduction

Protein cavities are holes in the interior of a protein that are not accessible to bulk solvent. They may be large enough to accommodate other atoms or molecules such as water, but may be empty. Empty cavities are termed 'voids'.

Traditionally, a distinction has been made between analysis of individual voids (normally considered to be those cavities which, while empty, are large enough to contain a water molecule) and general packing quality. In reality, identifying discrete voids and analysing packing quality are simply extremes of the same phenomenon — non-optimal packing of atoms in a protein.

Methods for assessing individual voids include VOIDOO by Kleywegt and Jones[1], a grid-based program, which maps a protein onto a 3D grid and then assigns grid points as protein, bulk solvent, or cavity depending on their location in relation to the protein structure. In comparison, VOLBL[2] is an analytical method that utilizes 'alpha-shapes' to calculate internal cavities for both the solvent accessible and molecular surface models. Both VOIDOO and VOLBL use probes of the same size to detect the solvent accessible surface of a protein and its internal cavities. Such methods do not allow discrete voids smaller than the size of a water molecule to be examined — if one uses too small a probe, the cavities will 'leak' to the bulk solvent.

In the extreme case of a zero-sized probe, one is making an assessment of packing quality, but if a single zero-sized probe is used for assessment of voids and of solvent, all cavities in the protein would be contiguous with the bulk solvent. Traditionally, therefore, different methods have been used for packing assessment. A number of groups have used Voronoi polyhedra[3, 4, 5, 6, 7]. Other methods are the 'occluded surface algorithm'[8] and the rapid algorithm QPack[9]. Other methods are listed by Fleming and Richards[10].

AVP adapts the grid-based void-finding methods to allow detection of voids of any size and may therefore also be used to assess packing quality. This is achieved by using two probes: one to delimit the solvent accessible regions, the second to identify voids. Uniquely, this method allows for very small or zero sized void probes without 'leakage' of the void into the bulk solvent. We are thus able to assess both packing quality and individual voids, previously treated as separate problems, using a single method.

2 Using AVP

While AVP allows you to use a zero-sized probe for detecting voids, in practice this is not particularly useful. If you do this to assess the size of individual voids, then you are most likely to end up with a single void which 'leaks' throughout the protein and is therefore of similar size to the total void volume. After extensive testing, we recommend a void probe size of radius 0.5Å which is in general too large to pass though gaps between closely packed protein atoms, but is small enough to detect most important small voids.

While orientation effects are minimal and measures have been taken in the program to allow 'off-grid' refinements, probe sizes greater than 0.0Å but less than 0.5Å are not recommended since the evaluation of individual voids is much more sensitive to orientation of the protein on the grid at these sizes.

Using a zero-sized probe to assess packing is more useful, but, of course, even optimally packed atoms will show a void volume if this is done. For example, channels can be seen running down the centre of an alpha-helix! Once again, extensive testing has shown that a probe radius of 0.5Å is a sensible choice and has the added benefit of not having to run the program twice if you are interested in both individual voids and overall packing.

If one is assessing the effects of mutations on void sizes, then the fact that a single orientation is used means that relative values can be compared safely.

Void size refinement must be switched on explicitly with -R and the void probe size must be specified with -p.

3 Usage

Recommended use is:

```
avp -R -p 0.5 file.pdb file.out
```

```
avp V1.5 (c) 2001-20, Prof. Andrew C.R. Martin, University of Reading,
                      UCL

Usage: avp [-q] [-g gridspacing] [-p probesize] [-s solventsize]
           [-r] [-e] [-c] [-o[a][s] file]
           [-f file] [-l] [-n file] [-O(xyz) value] [-w] [-S]
           [file.pdb [file.out]]
   -q Quiet - do not report progress
   -g Specify the grid spacing (Default: 1.000000)
   -p Specify the probe size (Default: 0.000000)
   -s Specify the solvent size (Default: 1.400000)
   -r Refine void sizes
   -R Redefine void points and refine void sizes
```

```
   -e Assign points to voids with edge connections
   -c Assign points to voids with corner connections (implies -e)
   -o Output the void grid points to file. With 'a', also output atom
      grid points; with 's', also output solvent grid points
   -f Output the refined void points to a file. Used with -r
   -l Include protein voxels next to void and solvent in volume
      refinement
   -n Output atom records for atoms nearest to each void to file.
   -Ox -Oy -Oz Specify an offset for the grid
   -w Print a list of waters that neighbour voids
   -S Reassign surface protein to solvent if can fit a solvent near

avp (Another Void Program) is a program to calculate void volumes in
proteins. It uses a simple grid-based method but separates the probe
size used to define void points as being voids (-p) from the probe
size used to define channels to the surface (-s). Thus one can find
very small voids without these being connected via very small
diameter passages to the surface.
```

-q The quiet flag simply stops progress information being printed. Useful for automated runs.

-g Specify the grid spacing. Using a finer grid will give more accurate evaluations of void sizes, but will dramatically increase memory usage. It is rarely necessary to change the default. (Default: 1.000000)

-p Specify the probe size. This is the most useful parameter that you are likely to need to change. The default is only recommended for use in packing evaluation; a value of 0.5 is recommended for individual void analysis and is also suitable for packing evaluation. (Default: 0.000000)

-s Specify the solvent size. Rarely necessary to change the default value which is the size of a water molecule. (Default: 1.400000)

-r Refine void sizes This (or `-R`) is absolutely required. Without it you will only get a rough approximation of the void size. Refinement of the void sizes splits each voxel into 1000 sub-voxels to fine-tune the void size.

-R This does a `-r`, but also provides off-grid refinements to minimize the effect of grid orientation. This is the recommended default.

-e Void clustering, by default, only clusters voxels if they are face connected. This adds edge-connected voxels. Whether you use this depends on your definition of a distinct void!

-c Assign points to voids with corner connections. As `-e`, but also adds corner connected voxels.

-o Output the void grid points to a PDB file. If you use `-oa`, then it will also output grid points assigned as atoms; if you use `-os`, then points assigned as solvent will also be output. (You can also do -oas). The resulting output file is described in Section 3.1.

-f Output the refined void points to a file (i.e. the 1000 per original voxel). Used with `-r` or `-R`. This is really just for debugging the void refinement process since there are too many points to use effectively.

-l Include protein voxels next to void and solvent in volume refinement. The voxel refinement normally includes only the voxels initially defined as void. Including this option also looks at the adjacent protein voxels (which may be part void). This is useful with `-r`, but the additional off-grid refinement provided by `-R` means it is less useful.

-n Output atom records for atoms nearest to each void to file. This is only partially implemented, but prints the atoms which are adjacent to the voids.

-Ox -Oy -Oz Specify an offset for the grid. This is only really used for debugging and analysis of how the program works. It allows the grid to be offset on the protein to assess grid orientation effects.

-w Print a list of waters that neighbour voids. This is used only for debugging.

-S Reassign surface protein to solvent if can fit a solvent near. This is the second phase of off-grid refinement. However, testing has shown that it does not add much to the lack of grid-orientation sensitivity compared with `-R`.

Recommended use is:

```
avp -R -p 0.5 file.pdb file.out
```

3.1 Void point output file

If you use the `-o` option, you will obtain a PDB file containing the void points. Typically you would merge this with the original PDB file (`cat original.pdb voids.pdb > both.pdb`) and view the combined file with something like RasMol or PyMol, rendering the voids as spacefilled spheres.

The following chain labels are used:

X	'Surface voids' — these are void points next to the water accessible surface, but not forming a channel to water.
Y	Fully buried void points.
Z	When using `-oa`, Atom grid points. When using `-os`, Solvent grid points.

4 Installation

Unpack the distribution file:

```
gzip -d AVP1.5.tar.gz
tar xvf AVP1.5.tar
```

If you are using GNU tar, you can combine these two steps:

```
tar zxvf AVP1.5.tar.gz
```

This will create a directory, AVP1.5

Now enter the src sub-directory and type make to build the software:

```
cd AVP1.5/src
make
```

With recent versions of the GCC C compiler, you can also compile to use multiple cores with OpenMP:

```
cd AVP1.5/src
make -f Makefile_gcc.omp
```

Finally copy the executable to somewhere appropriate:

```
cp avp /usr/local/bin
```

To get help on using AVP, type

```
avp -h
```

5 Algorithm

The AVP algorithm proceeds as follows:

Grid construction A grid is constructed around the protein; this grid is large enough that at least one plane of water probes can be placed on all sides. Initially each point on the grid is assigned as being of type void. By default, the spacing of this grid is 1Å.

Protein assignment The grid is searched and any grid point that is within an atom sphere is changed to type protein. In order to speed this process, the list of atoms is first sorted along the x-axis. When grid points are checked, a maximum and minimum possible x-coordinate are calculated from the current x grid position plus or minus the maximum radius of a protein atom (1.9Å). A binary search of the sorted atom list is performed to find only those atoms within a yz slice of the protein neighbouring the required x-coordinate. Simple maximum distance checks are made on the y- and z-coordinates before calculation of actual distances. Also, while performing this 'walk' across the grid, a list of atoms within 2 atom radii of each grid point is associated with that point. This is used later during void volume refinement (see below).

Solvent assignment The 6 surfaces of the grid are all assigned as type water. By moving along the positive and negative x-, y- and z-axes in turn, each point currently assigned as void is converted to solvent if a solvent probe can be placed at that point without clashing with protein and at least one of the 26 neighbouring points is already solvent (i.e. 6 face-connected, 12 edge-connected and 8 corner-connected neighbours). The same optimization of using an atom list sorted on x-coordinates with a binary search, described for protein assignment, is used during this stage. As well as setting each primary grid point to type solvent, any other points within the solvent radius (optionally multiplied by a 'solvent expansion factor') are also converted to type solvent. This process then iterates until no new points are converted from void to solvent. Points within a solvent radius must be converted to solvent as, if this is not done, a 'shell' of void points is formed all around the protein surface. In the strategy adopted by Kelywegt and Jones[1], these points were instead assigned as protein as they grew the protein atoms by the probe radius. An alternative to the iterative strategy adopted here would be to use a flood fill in 3 dimensions. However, the standard flood-fill algorithm is recursive and is computationally impractical for problems of this size.

Void clustering The preceeding steps have flagged grid points classified as protein or solvent; remaining points, still assigned as type void, are now true voids. These are clustered to determine the distinct void regions in the protein. This is done by walking along the grid to find void points. Once a void point is found, a standard three-dimensional flood fill is started to cluster all connected adjacent points into the same void. The walk across the grid then continues until another void point is found that is not yet assigned to a void cluster.

Void volume refinement At this stage an estimate of the void volumes may be made by assuming each void point represents a voxel of volume equal to the grid spacing cubed. At small or zero void probe sizes this will generally be a large over-estimate although some grid points may have been assigned as protein whereas the voxel they represent may be partially void. To improve accuracy, each void voxel and all neighbouring protein voxels are therefore split into 1000 sub-voxels and the total number of void sub-voxels is then counted. Only those atoms in the list associated with each original grid point are examined when making this assignment to speed up the search.

The following adaptations are used to minimize the grid orientation effects, As described above, voxels are initially assigned as protein by checking whether the centre of the voxel is within the van der Waals radius of a protein atom. We then check each voxel, initially assigned as protein, to see whether it contains any off-centre points at which a probe of minimum radius 0.1Å could fit. To do this, we build a list of neighbouring atoms and look at each pair of atoms in turn. If two atoms are separated by more than the sum of their radii plus the diameter of the probe, then we walk along the vector between the two atoms in 0.05Å steps and check whether any point is at least a probe radius away from all other atoms — if so, the voxel is reassigned as void. A voxel can thus be assigned as void even if the grid spacing assigns it as protein. In the last step of the procedure described above, every voxel assigned as void is refined by splitting it into 1000 sub-voxels each of which is individually assigned as protein or void.

Secondly we examine all surface protein voxels and, in a similar way, reassign them as solvent if a solvent probe can be placed at any point along the vector between the centre of this voxel and the centre of an adjacent protein voxel. Again, this allows voxels whose centres are within the protein to be treated as solvent and, after voxel refinement (which reassigns sub-voxels as solvent or protein) produces a more accurate evaluation of void volume reducing leakage of voids to the protein surface.

6 Citing AVP

If you use AVP in your research, please cite the following paper:

Cuff, A. L. & Martin, A. C. R. (2004) Analysis of void volumes in proteins and application to stability of the p53 tumour suppressor protein. *J. Mol. Biol.* **344**, 1199–1209.

References

[1] Kleywegt, G. & Jones, T. A. (1994). Detection, delineation, measurement and display of cavities in macromolecular structures. *Acta Cryst.* **D50**, 178–185.

[2] Edelsbrunner, H., Facello, M., Fu, P. & Liang, J. (1995). Measuring proteins and voids in proteins. *Proc. 28th Ann. Hawaii Internat. Conf. System Sciences* **5**, 256–264.

[3] Lesk, A. M. & Chothia, C. (1980). Solvent accessibility, protein surfaces, and protein folding. *Biophys. J.* **32**, 35–47.

[4] Ptitsyn, O. B. & Volkenstein, M. V. (1986). Protein structure and neutral theory of evolution. *J. Biomol. Struct. Dyn.* **4**, 137–156.

[5] Gerstein, M., Sonnhammer, E. L. & Chothia, C. (1994). Volume changes in protein evolution. *J. Mol. Biol.* **236**, 1067–1078.

[6] Tsai, J., Taylor, R., Chothia, C. & Gerstein, M. (1999). The packing density in proteins: Standard radii and volumes. *J. Mol. Biol.* **290**, 253–266.

[7] Richards, F. M. (1974). The interpretation of protein structures: total volume, group volume distributions and packing density. *J. Mol. Biol.* **82**, 1–14.

[8] Pattabiraman, N., Ward, K. B. & Fleming, P. J. (1995). Occluded molecular surface: Analysis of protein packing. *J. Molec. Recog.* **8**, 334–344.

[9] Gregoret, L. M. & Cohen, F. E. (1990). Novel method for the rapid evaluation of packing in protein structures. *J. Mol. Biol.* **211**, 959–974.

[10] Fleming, P. J. & Richards, F. M. (2000). Protein packing: Dependence on protein size, secondary structure and amino acid composition. *J. Mol. Biol.* **299**, 487–498.