Computer Physics Communications 180 (2009) 455–458

Contents lists available at ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc

g_permute: Permutation-reduced phase space density compaction ☆

F. Reinhard [a], O.F. Lange [b], J.S. Hub [d,e], J. Haas [c,e,*], H. Grubmüller [c,e]

[a] Laboratoire Kastler Brossel, ENS, UPMC-Paris 6, CNRS, 24 rue Lhomond, 75005 Paris, France
[b] Department of Biochemistry, University of Washington, Seattle, WA 98195, USA
[c] Theoretical and Computational Biophysics Department, Am Fassberg 11, 37077 Göttingen, Germany
[d] Computational Biomolecular Dynamics Group, Am Fassberg 11, 37077 Göttingen, Germany
[e] Max-Planck-Institute for Biophysical Chemistry, Am Fassberg 11, 37077 Göttingen, Germany

ARTICLE INFO

Article history:
Received 11 July 2008
Received in revised form 22 October 2008
Accepted 29 October 2008
Available online 8 November 2008

PACS:
02.60.Pn
02.70.-c
02.70.Ns
05.10.-a
87.10.-e
87.10.Tf
05.70.-a
65.40.gd

Keywords:
Permutation reduction
Solvent entropy
All-atom (fully atomistic) molecular dynamics simulations
GROMACS
Compacted configuration space density

ABSTRACT

Biomolecular processes are governed by free energy changes and thus depend on a fine-tuned interplay between entropy and enthalpy. To calculate accurate values for entropies from simulations is particularly challenging for the solvation shell of proteins, which contributes crucially to the total entropy of solvated proteins, due to the diffusive motion of the solvent molecules. Accordingly, for each frame of a Molecular dynamics (MD) trajectory, our software relabels the solvent molecules, such that the resulting configuration space volume is reduced by a factor of $N!$ with N being the number of solvent molecules. The combinatorial explosion of a naive implementation is here overcome by transforming the task into a linear assignment problem, for which algorithms with complexity $\mathcal{O}(N^3)$ exist. We have shown in previous research that the solvent entropy can be estimated from such a compacted trajectory by established entropy estimation methods. In this paper, we describe the software implementation which also allows applications beyond entropy estimation, such as the permutation of lipids in membrane bilayers.

Program summary

Program title: g_permute
Catalogue identifier: AECJ_v1_0
Program summary URL: http://cpc.cs.qub.ac.uk/summaries/AECJ_v1_0.html
Program obtainable from: CPC Program Library, Queen's University, Belfast, N. Ireland
Licensing provisions: GPL
No. of lines in distributed program, including test data, etc.: 45 173
No. of bytes in distributed program, including test data, etc.: 2 730 678
Distribution format: tar.gz
Programming language: C
Computer: PC-compatible running Linux
Operating system: Linux
RAM: Dependent on the number of solvent molecules, min 12 582 912 bytes
Classification: 3, 4.8, 4.9
External routines: liblap (included); From GROMACS-3.3.1: libgmx (not included)
Nature of problem: Estimating the entropy of solvent molecules from a molecular dynamics simulation trajectory cannot be performed on ordinary trajectories.
Solution method: Compacting the configuration space of molecules by exploiting their permutation symmetry. Applies to trajectories either compatible to those obtained with the GROMACS simulation package [1] or multi-model pdb (Protein Data Bank) files.
Restrictions: In rare cases the time to find a solution for the linear assignment problem can be very long.
Running time: Dependent on trajectory length and number of molecules to be permuted.

☆ This paper and its associated computer program are available via the Computer Physics Communications homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author at: Max-Planck-Institute for Biophysical Chemistry, Am Fassberg 11, 37077 Göttingen, Germany.
E-mail address: jhaas@gwdg.de (J. Haas).

0010-4655/$ – see front matter
doi:10.1016/j.cpc.2008.10.018

References:
[1] D. van der Spoel, et al., J. Comput. Chem. 26 (2005) 1701.

1. Introduction

Solvation is known to directly effect not only the structure of proteins and other macromolecular complexes [1], but also their functions in biological processes. These are governed by free energy changes and thus depend on a fine interplay of enthalpy and entropy. Paradoxically, although all entropic effects and driving forces are fully described by atomistic MD simulations, it turns out to be rather challenging to actually extract accurate values for entropic contributions from an MD trajectory. This is particularly true for the biologically relevant solvation shell, which, due to the diffusive motion of the solvent molecules is inaccessible to the straightforward application of established entropy estimation methods [2,3]. The main obstacle is the huge configuration space, which is inadequately sampled. We have shown in previous research, that the permutation symmetry of the solvent can be exploited by transforming the trajectory with the permutation reduced (PR) approach [4] such that established estimation methods are applicable, e.g., the quasiharmonic approximation or principal component analysis (PCA). Here, we focus on the implementation of the PR approach within the GROMACS framework. The method involves a combinatorial problem, which is solved through its equivalence with the linear assignment problem, for which $\mathcal{O}(N^3)$ methods exist [5]. A further advantage is that our implementation makes diffusive systems accessible to improved fit functions.

2. Theoretical background

Our program g_permute reads a trajectory and transforms it by permuting the indices of the solvent molecules in each frame. We refer to this process as "permutation reduction" or "relabeling" of the solvent molecules. This transformation leaves all properties of the system unchanged, such that the resulting trajectory will still provide accurate thermodynamic quantities, albeit with the potential to significantly improve sampling. Apparently, optimal sampling is achieved for maximally reduced configuration space volume. As will be detailed below, the combinatorial problem of selecting for each MD frame the one, which maps the configuration vector into a maximally compact volume out of the $N!$ possible permutations, is overcome through the equivalence with the linear assignment problem. This renders the resulting trajectory accessible to established density estimation methods such as PCA and solves the sampling problem.

To describe the algorithm in detail, let us consider a pure solvent trajectory, containing N solvent molecules consisting of m atoms each. Each configuration ("frame") of this trajectory will thus be described by a $3Nm$-dimensional configuration vector $\mathbf{x}$. To uniquely label the molecules and atoms, we will assume this vector to consist of Nm 3-dimensional vectors $\mathbf{x}_{i,j}$, $i \in 1 \ldots N$, $j \in 1 \ldots m$, denoting the position of atom j of the solvent molecule i. The full input trajectory is thus given by a time series of T frames of these vectors

$$\mathbf{x}_{i,j}(t_n), \quad i = 1 \ldots N, \ j = 1 \ldots m, \ n = 1 \ldots T.$$

Let furthermore $\mathbf{x}^0$ be a fixed reference configuration, e.g., the first frame of the trajectory.

We aim at transforming the trajectory frame by frame by relabeling the molecules. Its output will thus be a trajectory $\mathbf{x}_{\pi_n(i),j}(t_n)$ with $\pi_n \in \mathrm{Sym}_N$ being a permutation of the N solvent molecules, which in general will be different for each frame.

To obtain a trajectory, that is maximally compact in configuration space, each permutation π_n is chosen such as to map the respective configuration of each frame $\mathbf{x}(t_n)$ to a new configuration which is as close as possible to the reference structure $\mathbf{x}^0$ according to $3Nm$-dimensional Euclidean metrics (see Fig. 1),

$$d_n = \sum_{i=1}^{N} \left| \mathbf{x}_{\pi_n(i),1}(t_n) - \mathbf{x}^0_{i,1} \right|^2.$$

Finding π_n is equivalent to the "linear assignment problem" (LAP), for which $\mathcal{O}(N^3)$ methods exist [5]. g_permute is based on a LAP solving algorithm developed by Jonker [6], an implementation of this algorithm is available electronically [7] and distributed along with g_permute.

Two different ways of computing the distance d_n are implemented in g_permute. By default, only the distances between the first atoms ($j = 1$) are considered. This procedure is appropriate for permuting water molecules, where the first atom corresponds to the oxygen atom. Alternatively, g_permute can operate in a center of mass (COM) mode. Then, d_n is the distance between the COMs of the solvent molecules, extending greatly the range of molecules, which can be permuted (e.g., to lipids), and hence the scope of possible applications. For both modes one can either supply a reference frame containing the reference structure or points (COMs) for the group of molecules to be permuted. Alternatively, the reference structure or COMs are calculated from the first frame.

From a configuration space perspective, the resulting trajectory is mapped into a compact configuration space region centered around the reference configuration $\mathbf{x}_0$. It is instructive, however, to consider this permutation in real (3D) space (Fig. 2).

Recall that, a molecule A will be swapped (relabeled) with another molecule B, whenever A approaches the reference position of B. As a result, each molecule will remain close to its reference position (Fig. 2B), rather than exploring the full available as in the original trajectory (Fig. 2A).

3. Overview of the software structure

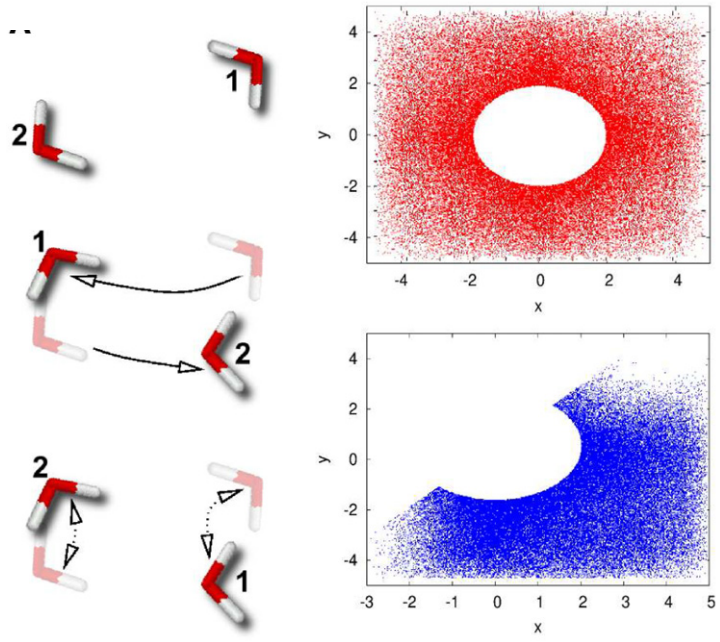
g_permute is written in C and linked to the lap and GROMACS libraries. While liblap is supplied with the distribution, libgmx has to be installed separately (script supplied). Sample makefiles are also provided for the GNU gcc compilers. Liblap is the library for the linear assignment solving algorithm [6], while libgmx contains the GROMACS routines needed for fitting and in- and output of the trajectory as well as all structure files.

4. Description of the individual software components

The distribution consists of g_permute.c itself (main program) and further lap.c (code for linear assignment problem) and memory.c (memory handling of matrices and vectors), the latter two are to be compiled into the shared library liblap.

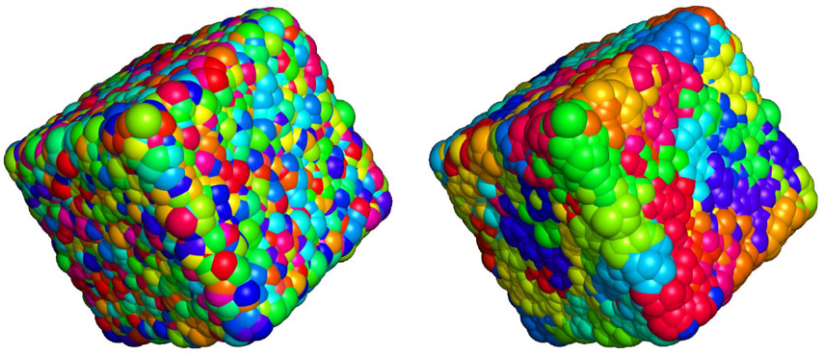
5. Installation instructions

(1) install gromacs-3.3.1 (shell script "instGMX331.sh" supplied).
(2) cd into 'liblap' and edit the Makefile.



(A) (B)

Fig. 1. Panel A shows the relabeling step for one pair of water molecules. In panel B the compacting of the configuration space density is illustrated for the 2D case.



(A) (B)

Fig. 2. Overlay of 200 frames of a MD trajectory containing 216 water molecules in a cubic box. Shown are only the oxygen atoms. Each atom is assigned one unique color for all frames. A: Original trajectory, B: relabeled trajectory. The relabeling algorithm constrains each atom to a region near its reference position, leading to the clustering of colors in B. Figure prepared and rendered with PYMOL [8].

- set the environment variable "PREFIX" to the directory, where you want the final library to be installed (the default PREFIX=../../g_permute-1.1 will create a directory g_permute-1.1 in the same directory where the tar archive was decompressed to and install liblap.so to../../g_permute-1.1/lib.
  Make sure the path set here is in your "LD_LIBRARY_PATH" when running g_permute.
- specify "GMXDIR", "GMXLIB" and "GMXINC" to point to your GROMACS installation. Furthermore, make sure you have compiled GROMACS with shared libraries enabled (i.e. with 'configure-enable-shared' or use the supplied script).
- type 'make', then 'make install'.

(3) cd into 'src' and edit the Makefile as in../liblap.
type 'make' followed by 'make install'.
'make install' with the default settings will install g_permute to../../g_permute-1.1/bin.

6. Test run description

6.1. Data input

g_permute takes a maximum of four input arguments, three filenames and one numerical argument.

-f the input trajectory.
-n an index file.
-r the reference structure (use this to override the default of using the first frame as a reference. If -com is set the default is to use COM coordinates calculated from first frame in trajectory).
-m the number of atoms per molecule to be permuted.

6.2. Optional switches

g_permute takes up to five optional switches, where the switches -b and -e are to be used for frames selection, while -com

and -rm_pbc affect the distance calculation itself. The switch -dt reduces the number of frames to include only those where time t modulo the timestep of the trajectory dt is equal the time indicated.

-com center of mass (COM) mode instead of default picking of first atom (oxygen in case of water).
-rm_pbc calculate distance using periodic boundary conditions, default is off.
-b Begin at the specified frame in ps.
-e End at the specified frame in ps.
-dt Only consider frames when time t MOD dt == first time (ps), otherwise disregard frame.

6.3. Data output

Output is written to a file specified by -o, there are two optional output files containing COM coordinates, either permuted or in original order.

-o the output trajectory.
-oref trajectory containing COM coordinates in original order (optional).
-orefp trajectory containing the permuted COM coordinates (optional).

6.4. *Index file* [*index.ndx*] *for permuting TIP4P water trajectories*

In an index file the indices of the atoms in one group are listed below line with the [ ] tags, the first group here are the oxygen atoms, the second group includes all water atoms.

```
[ oxygen_atoms ]
1 5 9 13 etc.
[ water_atoms ]
1 2 3 4 5 6 etc.
```

6.5. *Example run*

The simplest command to permute TIP4P water (see Fig. 2) in a GROMACS-based trajectory is:

g_permute -m 4 -s start_struc.pdb -f traj.xtc
-o permute.xtc -n index.ndx

g_permute then reads in the topologies of the system from *start_struc.pdb* and the trajectory from *traj.xtc*, referring to the groups listed in *index.ndx* as described above. The permuted trajectory will be written to *permute.xtc*. Note that the output trajectory can also be written directly to a multi-model pdb file.

Similarly, when permuting lipids containing 46 atoms per molecule:

g_permute -m 46 -s start_struc.pdb -f traj.xtc
-o permute.xtc -n index.ndx

The index file in this case would contain two groups, one, e.g., resembling the atoms in the head group of the lipid and a second listing all atoms of the lipids.

Acknowledgements

We thank Stephanus Fengler for valuable discussions. This work was supported by the Human Frontier Science Program (Grant RGP 53/2004) and the Volkswagen foundation (Grant numbers: I/80436, I/80585).

References

[1] P. Ball, Chem. Rev. 108 (2008) 74.
[2] J. Schlitter, Chem. Phys. Lett. 215 (1993) 617.
[3] R.M. Levy, A.R. Srinivasan, W.K. Olson, J.A. McCammon, Biopolymers 23 (1984) 1099.
[4] F. Reinhard, H. Grubmüller, J. Chem. Phys. 126 (2007) 014102.
[5] R.E. Burkard, Discrete Appl. Math. 123 (2002) 257.
[6] R. Jonker, A. Volgenant, Computing 38 (1987) 325.
[7] MagicLogic, http://www.magiclogic.com/assignment.html.
[8] W. DeLano, DeLano Scientific LLC, Palo Alto, CA, USA, http://www.pymol.org, 2008.