RMSD computation for clusters of identical particles

EUN-JONG HONG, KYU-HWAN LEE, WOLFGANG WENZEL

Abstract:- The root mean square deviation (RMSD) is nowadays universally used to assess the similarity or difference of different conformations of clusters, molecules, nanostructures and macromolecular assemblies. However, surprisingly, after decades of use of this method there is no available implementation to compute the RMSD of sets of identical particles, which are not covalently bonded. Since the computational effort of enumerative techniques grows exponentially with the number of particles, straightforward generalizations of established alignment procedures cannot be applied to this problem. Here we developed a computational strategy that employs branch-and-bound algorithms for the solution of this problem and demonstrate the feasibility for clusters of up to 60 particles. In test calculations our algorithm succeeds to find the global solution of the RMSD problem by sampling (on average) just 120 of 10^{80} possible permutations.

Key-Words:- molecular similarity, RMSD, branch-and-bound, cluster similarity

I. INTRODUCTION

Computation of the root mean square deviation, as a measure of geometrical similarity, is probably the most widely used and universally accepted criterion for nanostructures, including clusters, molecules (ranging from small organic compounds to large macromolecular assemblies). The root mean square deviation (RMSD) measures the average deviation between two sets of atoms, after they are aligned with one another as perfectly as possible. In a recent application of cluster comparisons for bio-macromolecules, we were confronted with the problem of computing the RMSD of sets of coordinates that are not covalently bonded. Obviously such sets of coordinates have no inherent order and an optimal alignment must sample the different permutations of the particles before constructing the optimal translation-rotation for geometrical alignment. The number of such possible permutations of coordinates for identical particles grows exponentially with the number of particles, so that a straightforward generalization of existing methods for RMSD computation by enumerating all possible permutations is unfeasible for all but the smallest number of particles.

Much to our surprise we found no published solution to this problem in the literature, in addition we learned that simple modifications of the enumerative scheme, such as a downhill search of pairwise exchanges of particles, do not -- in general -- converge to the correct solution.

Manuscript received May 9, 2007: Revised received September 4, 2007

Eun-Jong Hong is with the Department of Electrical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
Kyu-Hwan Lee is with the Computational Science Center, Korean Institute of Science and Technology, 39-1 Hawolgok-don, Seong-buk Gu, Seoul 136-791, KOREA
Wolfgang Wenzel is with the Institute for Nanotechnology, Research Center Karlsruhe, PO Box 3640, D-76021 Karlsruhe, GERMANY, wolfgang.wenzel@int.fzk.de, http://www.fzk.de/biostruct

We have therefore developed an algorithm, that combines the traditional geometrical alignment with a branch-and-bound search over permutations that permits the solution of the RMSD problem for identical particles for much larger system size.

II. PROBLEM FORMULATION

The RMSD (root mean square distance) measures distance of two sequences of 3D points. That is, the RMSD for two sequences $X = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ and $Y = (\mathbf{y}_1, \ldots, \mathbf{y}_n)$ is defined as

$$RMSD(X,Y) = \sqrt{\frac{\sum_{i=1}^{N} \| \mathbf{x}_i - \mathbf{y}_i \|^2}{N}}. \quad (1)$$

Meanwhile, when we want to measure geometric difference of X and Y, the two point sets are aligned to each other as closely as possible before computing the RMSD. We call such RMSD as $RMSD_M$. Assuming $\mathbf{c}_X$ and $\mathbf{c}_Y$ are centers of mass for X and Y, respectively, i.e.

$$\mathbf{c}_X = \sum_{i=1}^{N} \mathbf{x}_i / N, \quad (2)$$

and

$$\mathbf{c}_Y = \sum_{i=1}^{N} \mathbf{y}_i / N, \quad (3)$$

we let $\mathbf{p}_i = \mathbf{x}_i - \mathbf{c}_X$ and $\mathbf{q}_i = \mathbf{y}_i - \mathbf{c}_Y$. If we let $R(\theta)$ be the rotation matrix of angle θ, then $RMSD_M$ is defined as follows:

$$RMSD_M(X,Y) = \min_{\theta} \sqrt{\frac{\sum_{i=1}^{N} \| R(\theta)\mathbf{p}_i - \mathbf{q}_i \|^2}{N}}.$$
(4)

That is, $RMSD_M$ measures the geometric distance of X and Y when they are translated so that their centers of mass coincide and are optimally rotated around the center of mass.

$RMSD_M$ assumes that $\mathbf{x}_i$ matches $\mathbf{y}_i$. That is, we have a prior knowledge of point-to-point correspondences between X and Y. However, when X and Y represent sets of identical particles as in atom clusters, we need to consider every possible mapping between X and Y to compute geometric difference. Let F be the set of all possible one-to-one correspondences between X and Y. Then, we define $RMSD_C$ as follows:

$$RMSD_C(X,Y) = \min_{f\in F}\min_{\theta}\sqrt{\frac{\sum_{i=1}^{N}\|R(\theta)\mathbf{p}_i - f(\mathbf{p}_i)\|^2}{N}}. \qquad (5)$$

III. PROBLEM SOLUTION

Computation of $RMSD_C$ requires exploration of different one-to-one correspondences between X and Y. Due to its combinatorial property, the problem is conveniently formulated as a search problem in the branch-and-bound (BnB) framework.

A. Branch-and-bound

There are two main ideas in branch-and-bound. First, it uses branching (or splitting) to tackle a large problem effectively. Instead of solving one huge problem, BnB solves many smaller subproblems. Second, it uses bounding. Unlike naive enumerative approaches that explores every feasible solution, BnB attempts to save computation by only (lower) bounding the optimal solution of a subproblem and compare the (lower) bound to a known feasible solution (upper bound) [2].

1. Branching

Branching is done by picking one unassigned point of X, then generating child subproblems, as many as the number of possible matches for the selected point, i.e. the number of unassigned points in Y. Therefore, the root node (depth = 1) will have N children, nodes with depth 2 will have $N-1$ children, and so on.

Current scheme of selecting the point to split is as follows: for each point in X, compute the maximum pairwise distance from every other point in X. Pick the point whose maximum distance is the largest among the unassigned points. The idea is that we first try to match the pair of points that are most far apart. If the there are only a small number of points that are far apart, the scheme will work as expected.

2. Upper bounds

In $RMSD_C$ computation, an upper bound can be obtained from an arbitrary one-to-one correspondence of X and Y. In our method, we generate a random permutation to match unassigned points of X to unassigned points of Y, and compute $RMSD_M$ for the resulting one-to-one correspondence. In each subproblem, we repeat this for a fixed number of time.

3. Lower bounds

For each non-root subproblem S, we can assume points $\mathbf{x}_1,\ldots,\mathbf{x}_m$ $(m \le N)$ are already matched to some m points in Y without loss of generality. Let F' be the set of one-to-one correspondences that are compatible with such m matches of points. Then, the optimal value of the subproblem ($OPT(S)$) is

$$OPT(S) = \min_{f\in F'}\min_{\theta}\sqrt{\frac{\sum_{i=1}^{N}\|R(\theta)\mathbf{p}_i - f(\mathbf{p}_i)\|^2}{N}}. \qquad (6)$$

This can be lower-bounded as

$$OPT(S) \ge \min_{f\in F'}\min_{\theta}\sqrt{\sum_{i=1}^{m}\|R(\theta)\mathbf{p}_i - f(\mathbf{p}_i)\|^2 /N}$$
$$= \min_{\theta}\sqrt{\sum_{i=1}^{m}\|R(\theta)\mathbf{p}_i - f(\mathbf{p}_i)\|^2 /N}. \qquad (7)$$

of

That is, the lower bound is obtained by finding the optimal rotation only for matched points.

4. Subproblem selection

In our recommended approach, the search tree is explored in a depth-first manner. In addition, for each branching, the child subproblems are ordered so that the subproblem with smallest lower bound is expanded first. This is similar to what best-first search does. In Section 4, we will show depth-first search is on average better than best-first search.

To be able to select a child subproblem based on its lower bound, the lower bound computation for a child subproblem is performed before adding the subproblem to the search tree. Therefore, during expansion, only upper bounds are updated.

B. Implementation

The BnB method was implemented in the C++ programming language using two public libraries:

- PICO BnB library [1]: C++ framework for implementing parallel BnB methods. It provides various skeleton procedures such as node creation/deletion, splitting, and bounding. In this work, BnB was implemented as a sequential method. Therefore, parallel functionalities were not used.
- NEWRAN03: C++ library for generating sequences of pseudo-random numbers from a wide variety of distributions. The library was particularly used for generating random permutations as random feasible solutions, therefore providing upper-bounds.

We implemented the generation of upper and lower bounds as discussed above. Each subproblem is expanded (roughly) by first computing the bounds and selecting the state of the node to be considered (bounded). We then compute a lower-bound for each candidate child subproblem and sort the candidate child subproblems in the ascending order of their lower-bounds. We keep only those candidates whose lower-bounds are less than the current global upper-bound, and report the number of such candidates as the number of children to be spawned from the current subproblem.

C. Numerical Experiments

1. Generating test cases

We wrote a python program to generate random test cases. for N particles by reading an arbitrary starting conformation, which is then subjected to an arbitrary permutation, after which the particles are displaced in each Cartesian direction by a random deviation drawn from an uniform random in the range of [-NOISE NOISE]. In principle this procedure generates an arbitrary new conformation. If NOISE is smaller than the distances between particles, the generated permutation is it the optimal permutation for the solution of the RMSD problem, however, this need not be the case for large values of the NOISE parameter. However even in the latter case, the geometric alignment based on the permutation used to generate the new conformation is an upper bound on the solution and can thus be used to evaluate the method.

2. Numerical Tests

According to our (still limited) computational experiments, rmsdCluster works extremely well when $RMSD_C \approx 0$. On a workstation with an AMD Opteron processor and 4 Gbytes of memory, and using a binary compiled with -O1 option by GNU C/C++ compiler, solving a test case with 300 points and $RMSD_C = 0$ took only 15 seconds. More results are summarized in Table 1.

N	N_{perm}	N_{bnb}	std
10	10^{6}	15.8	3.0
20	10^{17}	36.5	7.1
30	10^{31}	56.8	12.7
40	10^{46}	79.7	18.2
50	10^{63}	97.5	14.2
60	10^{80}	120.4	22.1

Table 1: Number of particles, total number of permutations, average number of nodes explored in the branch and bound method and standard deviation.

The data demonstrates that the fraction of nodes explored by the branch-and-bound algorithm to determine the optimal solution is a completely insignificant fraction of the total number of permutations. Even for N=10 exact enumeration of all possible permutations strains the presently available computational power, for the larger numbers of N investigated here, exact enumeration is completely impossible.

When large random noise was added to the moving coordinates, rmsdCluster often performed much worse. Particularly, when the number of points is 16, RADIUS = 50, and NOISE = 2, we obtained a case with $RMSD_C = 1.57$, and observed it took 324 seconds (exploring 114,944 nodes) to solve the case. We note good upper-bounds were not found effectively for such cases until almost the end of the run. After good upper-bounds were found, the BnB was completed very rapidly, i.e. pruning based on lower-bounds weas relatively efficient when a suboptimal upper-bound is available.

IV. CONCLUSIONS

We introduced and formulated the RMSD computation problem for cluster of identical points (particles) and suggested a BnB method to solve the problem exactly. The method was implemented as a C++ code (rmsdCluster) using the PICO branch-and-bound library and NEWRAN03 library. We have investigated the numerical performance of this implementation and found that only a minuscule fraction of the total number of permutations must be explored by the branch-end-bound method to find the optimal solution of the problem. To our knowledge this implementation represents the first exact solution of the RMSD problem for sets of identical particles.

For very dissimilar structures we believe that the present algorithm is still not completely optimal. In addition to optimization of the code, future work will therefore focus on the formulation of better heuristic upper bounds to further reduce its computational time.

REFERENCES

[1] J. Eckstein, C. A. Phillips, and W. E. Hart. Pico: an object oriented framework form parallel branch and bound. Technical report, RUTCOR, 2001. download available: http://www.cs.sandia.gov/ web1400

[2] G. L. Nemhauser and L. A. Wolsey. Integer anOn the otherd Combinatorial Optimization. Wiley, N. Y., 1988. download: http://www.robertnz.net/nr03doc.htm